Automatic Generation of DFA Minimization Problems

Gregor Sönnichsen

Universität Bayreuth

17. Juli 2020

Introduction

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

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

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

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

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...

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

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

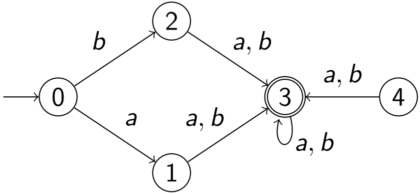
Outline

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

MinimizeDFA(A)

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

4. Merge all equivalent state pairs

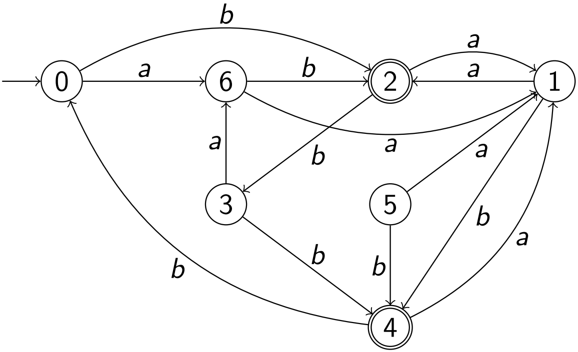

[2]

[2] http://www.cs.cornell.edu/gries/banquets/symposium40/images/faculty/jehyoung.jpg

Problem definition

A sample DFA minimization task. . .

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

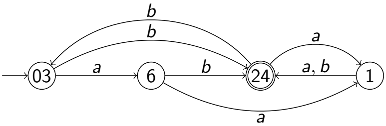
<u>*Solution:*</u>

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	■	■	■	■	■	■



Problem Definition and Approach

DFAMinimization

> Given : *A DFA* A_{task}.
> Task : *Compute* $A_{sol} = \text{MinimizeDFA}(A_{task})$.

Main question: How to generate instances of DFAMinimization?

Problem Definition and Approach

DFAMinimization

Given : *A DFA* A_{task}.
Task : *Compute* $A_{sol} = \text{MinimizeDFA}(A_{task})$.

Main question: How to generate instances of DFAMinimization?

Idea: First generate A_{sol}, then add equivalent, then unreachable states.

Parameters

↓

Generate DFA Minimization Problem

- Generate Minimal DFA
  (using a rejection algorithm and randomization/enumeration)
- Extend Minimal DFA
  - Add Equivalent States
  - Add Unreachable States

↓

A_{task}

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 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 DB_{found}
        return A_{test}

[3]

[3] `https://moviewriternyu.files.wordpress.com/2015/10/rejection.jpg`

We will restrict ourselves to $Q = [0, n_s - 1],\ \Sigma = [0, k - 1],\ s = 0.$

We will restrict ourselves to $Q = [0, n_s - 1],\ \Sigma = [0, k - 1],\ s = 0$.

Generation...

(a) by randomization:

$$F = random\_subset(Q)$$
$$\delta(q, \sigma) = choose\_one(Q) \qquad \forall q \in Q, \sigma \in \Sigma$$

We will restrict ourselves to $Q = [0, n_s - 1],\ \Sigma = [0, k - 1],\ s = 0$.

Generation...

(a) by randomization:

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

(b) by enumeration: An *enumeration state* $s_{n_s,k,n_F} = (F_F, F_\delta)$ has the following semantics:

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

Example state: $s_{4,2,2} = (0110)_2 \ \ (10\ 13\ 22\ 03)_4$

Outline

Extending Minimal DFAs
Adding Equivalent States (1)

We now want to add states $r_1, \ldots, r_{n_e}$ to the solution DFA, such that every r_i is equivalent to a state e in the solution DFA:

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

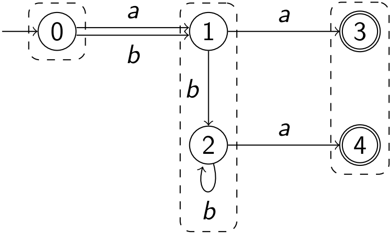
Whenever we add a state r_i, we will first choose an $e \in Q_{sol}$, then we add the transitions of r_i.

We now want to add states $r_1, \ldots, r_{n_e}$ to the solution DFA, such that every r_i is equivalent to a state e in the solution DFA:

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

Whenever we add a state r_i, we will first choose an $e \in Q_{sol}$, then we add the transitions of r_i.

Example

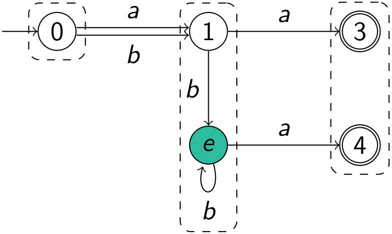


We now want to add states $r_1, \ldots, r_{n_e}$ to the solution DFA, such that every r_i is equivalent to a state e in the solution DFA:

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

Whenever we add a state r_i, we will first choose an $e \in Q_{sol}$, then we add the transitions of r_i.

Example



Extending Minimal DFAs

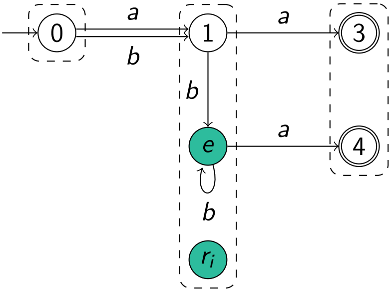
Adding Equivalent States (1)

We now want to add states $r_1, \ldots, r_{n_e}$ to the solution DFA, such that every r_i is equivalent to a state e in the solution DFA:

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

Whenever we add a state r_i, we will first choose an $e \in Q_{sol}$, then we add the transitions of r_i.

Example



Extending Minimal DFAs

Adding Equivalent States (2) - Outgoing Transitions

Observation:

$$r_i \sim_A e \implies \forall \sigma \in \Sigma \colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

Consequently:

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

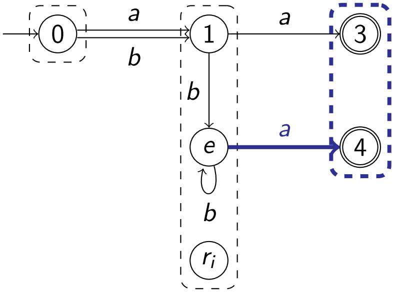
Observation:

$$r_i \sim_A e \implies \forall \sigma \in \Sigma \colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

Consequently:

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

Example



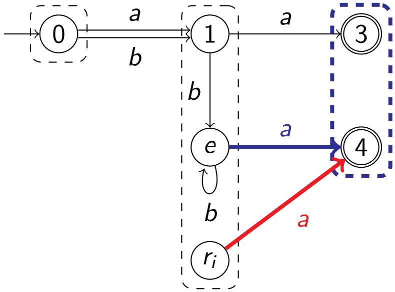
Observation:

$$r_i \sim_A e \implies \forall \sigma \in \Sigma\colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

Consequently:

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

Example



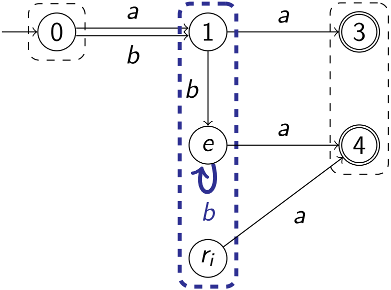
Observation:

$$r_i \sim_A e \implies \forall \sigma \in \Sigma\colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

Consequently:

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

Example



Extending Minimal DFAs

Adding Equivalent States (2) - Outgoing Transitions

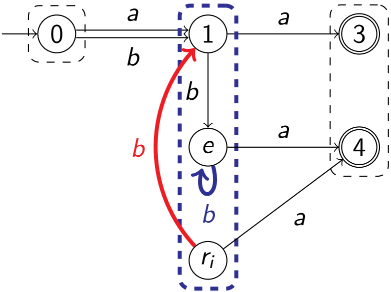
Observation:

$$r_i \sim_A e \implies \forall \sigma \in \Sigma\colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

Consequently:

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

Example



Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

Let q be a state s.t. $\delta(q, \sigma) = p$ and we want $\delta(q, \sigma) = r_i$.

q must remain in its equivalence class

$\Rightarrow$ p must be in $[r_i]_{\sim_A} = [e]_{\sim_A}$

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

Let q be a state s.t. $\delta(q, \sigma) = p$ and we want $\delta(q, \sigma) = r_i$.

q must remain in its equivalence class

$\Rightarrow$ p must be in $[r_i]_{\sim_A} = [e]_{\sim_A}$

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

Observation: Since r_i must be reachable, we require that there is at least one such transition for which $q \neq r_i$ is true.

Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

Let q be a state s.t. $\delta(q, \sigma) = p$ and we want $\delta(q, \sigma) = r_i$.

q must remain in its equivalence class

$\Rightarrow$ p must be in $[r_i]_{\sim_A} = [e]_{\sim_A}$

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

Observation: Since r_i must be reachable, we require that there is at least one such transition for which $q \neq r_i$ is true.

Furthermore, we see that p must have at least 2 *ingoing elements*.

$$in(q) = |d^{-}(q)| + \begin{cases} 1 & \text{if } s = q \\ 0 & \text{else} \end{cases}$$

Let q be a state s.t. $\delta(q,\sigma) = p$ and we want $\delta(q,\sigma) = r_i$.

q must remain in its equivalence class

$\Rightarrow$ p must be in $[r_i]_{\sim_A} = [e]_{\sim_A}$

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

Observation: Since r_i must be reachable, we require that there is at least one such transition for which $q \neq r_i$ is true.

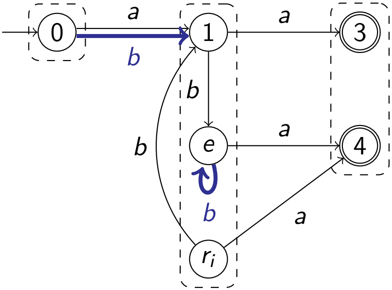
Furthermore, we see that p must have at least 2 *ingoing elements*.

$$in(q) = |d^{-}(q)| + \begin{cases} 1 & \text{if } s = q \\ 0 & \text{else} \end{cases}$$

R2: Choose at least one $((q,\sigma), p) \in \delta$ with $[p] = [e]$ and $in(p) \geq 2$. For at least one chosen transition $q \neq r_i$ must be true. Remove each $((q,\sigma), p)$ from δ and add $((q,\sigma), r_i)$.

R2: Choose at least one $((q,\sigma),p) \in \delta$ with $[p] = [e]$ and $in(p) \geq 2$. For at least one chosen transition $q \neq r_i$ must be true. Remove each $((q,\sigma),p)$ from δ and add $((q,\sigma),r_i)$.

Example

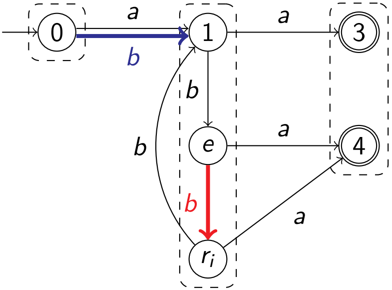


Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

R2: Choose at least one $((q, \sigma), p) \in \delta$ with $[p] = [e]$ and $in(p) \geq 2$. For at least one chosen transition $q \neq r_i$ must be true. Remove each $((q, \sigma), p)$ from δ and add $((q, \sigma), r_i)$.

Example

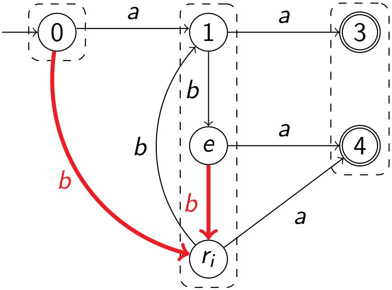


Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

R2: Choose at least one $((q,\sigma),p) \in \delta$ with $[p] = [e]$ and $in(p) \geq 2$. For at least one chosen transition $q \neq r_i$ must be true. Remove each $((q,\sigma),p)$ from δ and add $((q,\sigma),r_i)$.

Example



Extending Minimal DFAs

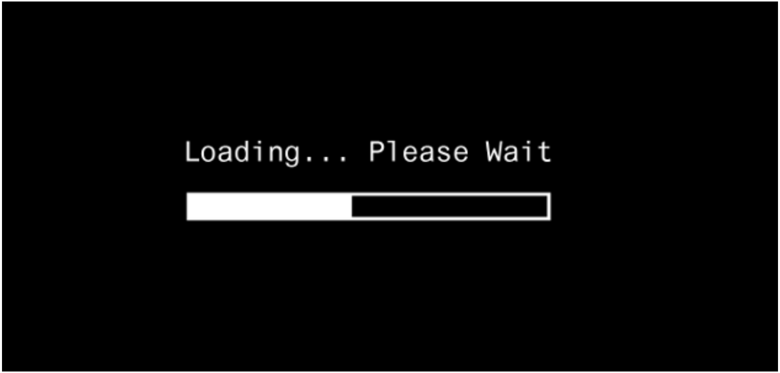
Adding Unreachable States

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 ← ∅
    for n_u times do
        let q be the new state
        steal ingoing tr. from a random subset of U × Σ
        add outgoing tr. to |Σ| random states
        add q to U
    return A
```

Outline

Live Demonstration



[4]

[4] https://sagamer.co.za/wp-content/uploads/2015/03/loading-please-wait.png

Conclusion

This presentation has...

- introduced the problem of DFA Minimization Problem Generation
- given an overview over a possible solution
- shown that the theoretic results might be useful in praxis

Conclusion

This presentation has...

- introduced the problem of DFA Minimization Problem Generation
- given an overview over a possible solution
- shown that the theoretic results might be useful in praxis

Lookout:

- more parameters, ranged parameters
  The *degree* of a state q is defined as $deg(q) = |d^-(q)| + |d^+(q)|$.
  $\Rightarrow$ capping the max. degree?
- investigate planarity and drawing algorithms
- complexity analysis

Conclusion

This presentation has...

- introduced the problem of DFA Minimization Problem Generation
- given an overview over a possible solution
- shown that the theoretic results might be useful in praxis

Lookout:

- more parameters, ranged parameters
  The *degree* of a state q is defined as $deg(q) = |d^-(q)| + |d^+(q)|$.
  $\Rightarrow$ capping the max. degree?
- investigate planarity and drawing algorithms
- complexity analysis

Thanks for listening!

Conclusion

This presentation has...

- introduced the problem of DFA Minimization Problem Generation
- given an overview over a possible solution
- shown that the theoretic results might be useful in praxis

Lookout:

- more parameters, ranged parameters
  The *degree* of a state q is defined as $deg(q) = |d^-(q)| + |d^+(q)|$.
  $\Rightarrow$ capping the max. degree?
- investigate planarity and drawing algorithms
- complexity analysis

Thanks for listening even longer!

Bonus-Slide

$\mathfrak{D}$-Proof

Definition

We will call a word w *distinguishing word of* p, q, iff

$$\delta^*(p, w) \in F \Leftrightarrow \delta^*(q, w) \notin F$$

Lemma

Iff $(p, q) \in m(n)$*, the shortest distinguishing word of* p, q *has length* n*.*

Lemma

If FindEquivPairs *has done* n *iterations and terminated (so* $\mathfrak{D}(A) = n$*), then the longest word* w*, that is a shortest distinguishing word for any state pair, has length* $\mathfrak{D}(A) - 1$*.*

Theorem

Given two DFAs A*,* A'*. If both are accessible and* $L(A) = L(A')$*, then* FindEquivPairs *runs with the same number of iterations on them:* $\mathfrak{D}(A) = \mathfrak{D}(A')$*.*