Automatic Generation of DFA Minimization Problems

Gregor Sönnichsen

Universität Bayreuth

16. Juli 2020

2020-07-16

1. stelle BA vor
2. trägt Namen
3. warum die Beschäftigung
4. kurz in kleiner Motivation erläutern

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:

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

Bekannt, AT klassischer Teil von inf-bezogenen Lehrplänen
DFA-Minimierung typische Aufgabe, einfache Gründe: Alg. ist...

-

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

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

Bekannt, AT klassischer Teil von inf-bezogenen Lehrplänen
DFA-Minimierung typische Aufgabe, einfache Gründe: Alg. ist...

-

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 automated 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 automated generation of DFA minimization problems is interesting because it could. . .

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

Bekannt, AT klassischer Teil von inf-bezogenen Lehrplänen
DFA-Minimierung typische Aufgabe, einfache Gründe: Alg. ist...

-

So ergibt sich dann auch...

Introduction

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

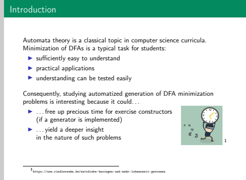
- sufficiently easy to understand
- practical applications
- understanding can be tested easily

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



Bekannt, AT klassischer Teil von inf-bezogenen Lehrplänen
DFA-Minimierung typische Aufgabe, einfache Gründe: Alg. ist...

-

So ergibt sich dann auch...

1. gibt Aufgabenersteller Zeit für andere Dinge
2. lässt auf interessante Einsichten hoffen
   -> Schwierigkeit

Outline

Outline

Introduction
Problem definition and approach
Generating Minimal DFAs
Extending Minimal DFAs
Live Demonstration and Conclusion

1. mit intro fertig
2. Den Ansatz in zwei Abschnitten im Detail erklären
3. im Anschluß wird es noch eine... und abschließend Resume ziehen

Problem definition

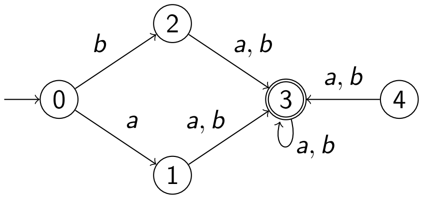
Preliminaries

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

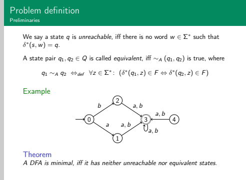
$$q_1 \sim_A q_2 \;\Leftrightarrow_{def}\; \forall z \in \Sigma^*: \; (\delta^*(q_1, z) \in F \Leftrightarrow \delta^*(q_2, z) \in F)$$

Example



Theorem

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



gehe davon aus, dass Def. DFA, Überführungsfkt. usw. bekannt ist

-

w, gelesen von ... aus, in ... führt

-

die folgende Relation für die beiden wahr ist

-

q1,q2 haben also das selbe Akzeptanzverhalten, egal welches Wort von den beiden Zuständen aus gelesen wird

-

weil sie schon für die selben Symbole in den jeweils selben Zustand führen

-

ein DFA ist minimal; dieses T. ist unweigerlich mit Hopcrofts Alg. verbunden

Problem definition

Hopcroft's Minimization Algorithm

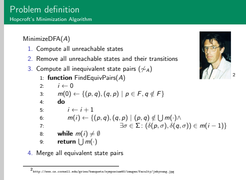
MinimizeDFA(A)

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)
   $i \leftarrow 0$
   $m(0) \leftarrow \{(p,q),(q,p) \mid p \in F, q \notin F\}$
   do
   $i \leftarrow i+1$
   $m(i) \leftarrow \{(p,q),(q,p) \mid (p,q) \notin \bigcup m(\cdot) \wedge$
   $\exists \sigma \in \Sigma \colon (\delta(p,\sigma),\delta(q,\sigma)) \in m(i-1)\}$
   while $m(i) \neq \emptyset$
   return $\bigcup m(\cdot)$
4. Merge all equivalent state pairs


[2]

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

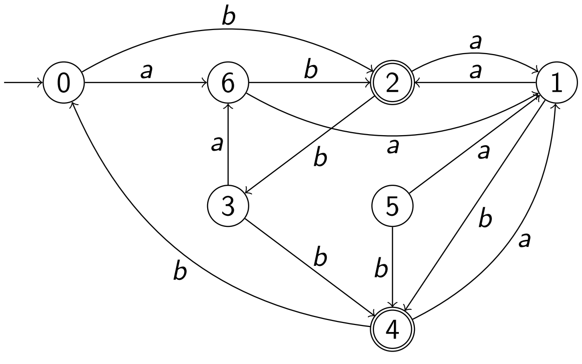


gehe auch hier davon aus
zur Erinnerung
-
wir init. eine Menge m(0) mit Zustandspaaren, von denen je genau ein Zustand akzeptierend ist
-
neue Menge m(i); Zustandspaare (p,q), (p,q) in keiner vorher ber. Menge, und Paar delta p sigma, delta q sigma ist in Vorgängermenge

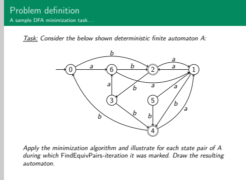
Problem definition

A sample DFA minimization task...

<u>Task:</u> 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.*



auf dieser Folie... Bsp DFA-Minimierungsaufgabe

Problem definition

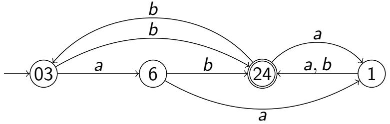
. . . and its solution

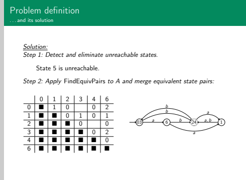
<u>*Solution:*</u>
Step 1: Detect and eliminate unreachable states.

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	■	■	■	■	■	■





Lsg, die dadurch zustandekommt, dass man Hop. Min.alg. anwendet

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?

Eine Min.aufgabe können wir wie folgt formalisieren

-

in Anlehnung an H.Min.alg.

-

in meiner Arbeit... Reihe an Parametern, mit denen man versch. Eig. des gen. Problems beeinfl. kann

-

Trial-and-Error-Methode

Problem Definition and Approach

DFAMinimization

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

Main question: How to generate instances of DFAMinimization?

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

Parameters

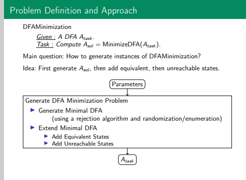
↓

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}



Eine Min.aufgabe können wir wie folgt formalisieren

-

in Anlehnung an H.Min.alg.

-

in meiner Arbeit... Reihe an Parametern, mit denen man versch. Eig. des gen. Problems beeinfl. kann

-

Trial-and-Error-Methode

Outline

Outline

Introduction

Problem definition and approach

Generating Minimal DFAs

Extending Minimal DFAs

Live Demonstration and Conclusion

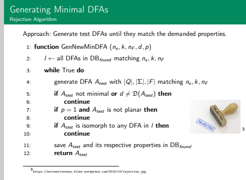
Generating Minimal DFAs
Rejection Algorithm

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

1: **function** GenNewMinDFA (n_s, k, n_F, d, p)
2: $I \leftarrow$ all DFAs in DB_{found} matching n_s, k, n_F
3: **while** True **do**
4: generate DFA A_{test} with $|Q|, |\Sigma|, |F|$ matching n_s, k, n_F
5: **if** A_{test} not minimal **or** $d \neq \mathfrak{D}(A_{test})$ **then**
6: **continue**
7: **if** $p = 1$ **and** A_{test} is not planar **then**
8: **continue**
9: **if** A_{test} is isomorph to any DFA in I **then**
10: **continue**
11: save A_{test} and its respective properties in DB_{found}
12: **return** A_{test}

[3]

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



1. while-Schleife
2. test DFA generieren, für die schon drei Parameter garantiert stimmen, nämlich im Enzelnen
3. restl. Eig. werden in z. 5-10, bsp. 5 minimalität
4. wenn es Fragen zu den and. get. Eig. gibt, dann kann ich hier gerne im Anschluss genauere Erläuterungen geben
5. Hier möchte.. nur auf test DFA Generierung in z.4 näher eingehen

Generating Minimal DFAs
Test DFA Generation

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

└Generating Minimal DFAs

Generating Minimal DFAs
Test DFA Generation
We will restrict ourselves to $Q = [0, n_s - 1],\ \Sigma = [0, k - 1],\ s = 0.$

Im Rahmen dieser Arbeit

Generating Minimal DFAs

Test DFA Generation

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

Generation...

(a) by randomization:

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

Generating Minimal DFAs
Test DFA Generation
We will restrict ourselves to $Q = [0, n_s - 1],\ \Sigma = [0, k-1],\ s = 0$.
Generation...
(a) by randomization:
$$F = random\_subset(Q)$$
$$\delta(q,\sigma) = choose\_one(Q) \qquad \forall q \in Q, \sigma \in \Sigma$$

Ich werde hier zwei Methoden zur Gen vorstellen. Die Gen. per Randomisierung funkt. hier wie folgt

-

Für die Tr.fkt. wird für jede Komb. eines Zustands und eines Symbols ein zufälliger Zustand ausgewählt

Generating Minimal DFAs
Test DFA Generation

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

Generation...

(a) by randomization:

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

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

Generating Minimal DFAs
Test DFA Generation
We will restrict ourselves to $Q = [0, n_s - 1],\ \Sigma = [0, k-1],\ s = 0$.
Generation...
(a) by randomization:
$F = random\_subset(Q)$
$\delta(q,\sigma) = choose\_one(Q) \qquad \forall q \in Q, \sigma \in \Sigma$
(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: $s_{4,2,2} = (0110)_2\ \ (10\ 13\ 22\ 03)_4$

verwalten wir für eine gegebene Enumeration einen

-

parametrisiert mit

-

wenn i nicht akz., dann FF[i] = 0; für alle Zustände definiert

-

2. Feld speichert für alle Komb. eines Zustands und eines Symbols zu welchem Zustand man gelangt

-

Unten angedeutet; Felder als Zahlen interpretieren; durch gewisse Inkrementierungsfkt. von einem DFA zum nächsten gelangen; nicht weiter ausführen

Outline

Outline

Introduction

Problem definition and approach

Generating Minimal DFAs

Extending Minimal DFAs

Live Demonstration and Conclusion

Kapitel der Lös.DFA Gen. abgeschlossen; widmen uns Erweiterung dieser DFAs hin zu Aufgaben-DFAs

-

Beschäftigung zunächst, gemäß vorhin vorg. Reihenfolge, mit Hinz. äqu. Z.

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.

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.

Sei nE die gewünschte Anzahl an paarweise versch. äqu. Zustandspaaren.

-

In diesem Verfahren werden die rI einzeln nacheinander hinzugefügt

Extending Minimal DFAs

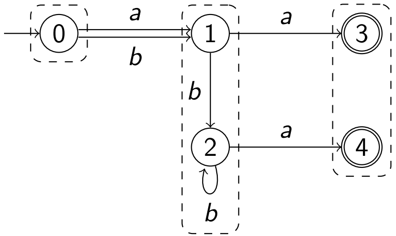
Adding Equivalent States (1)

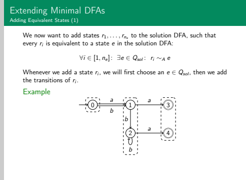
We now want to add states $r_1, \ldots, r_{n_e}$ to the solution DFA, such that every r_i is equivalent to a state e in the solution DFA:

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

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





im Bsp könnten wir zunächst Zst. 2 als orig. aussuchen, und dann rI gedanklich zu dessen Äqu. hinzufügen

-

dann würden wir jetzt die Tr. hinzufügen, dabei beginnen wir mit den ausgehenden Tr.

Extending Minimal DFAs

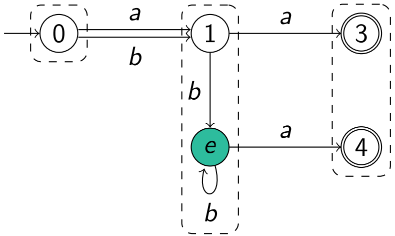
Adding Equivalent States (1)

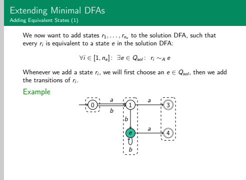
We now want to add states $r_1, \dots, r_{n_e}$ to the solution DFA, such that every r_i is equivalent to a state e in the solution DFA:

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

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

Example





im Bsp könnten wir zunächst Zst. 2 als orig. aussuchen, und dann rI gedanklich zu dessen Äqu. hinzufügen

-

dann würden wir jetzt die Tr. hinzufügen, dabei beginnen wir mit den ausgehenden Tr.

Extending Minimal DFAs

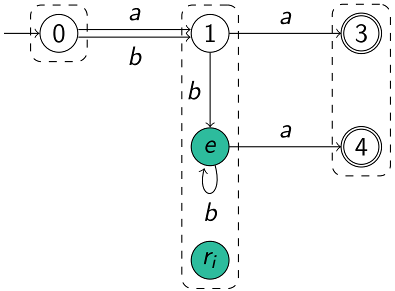
Adding Equivalent States (1)

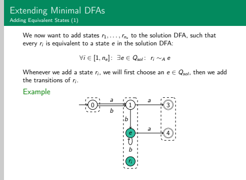
We now want to add states $r_1, \ldots, r_{n_e}$ to the solution DFA, such that every r_i is equivalent to a state e in the solution DFA:

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

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





im Bsp könnten wir zunächst Zst. 2 als orig. aussuchen, und dann rI gedanklich zu dessen Äqu. hinzufügen

-

dann würden wir jetzt die Tr. hinzufügen, dabei beginnen wir mit den ausgehenden Tr.

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 $\delta(r_i, \sigma) = q$.

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 $\delta(r_i, \sigma) = q$.

..dann sind folglich auch diejenigen Zust. äqu., wenn wir ein bel. Symbol von rI und e aus lesen

-

und setze diesen Zustand als Endzustand für delta(rI, sigma)

-

So konnten wir mit dieser Regel erreichen, dass jede Transition von rI in dieselbe Äqu.kl. führt, die wir err. würden, wenn wir der koresp. Tr. von e folgten

Extending Minimal DFAs

Adding Equivalent States (2) - Outgoing Transitions

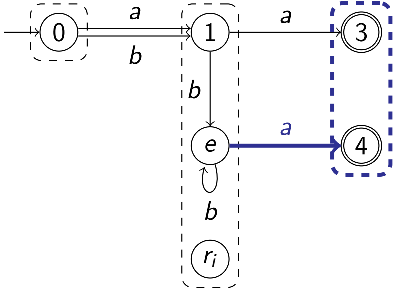
Observation:

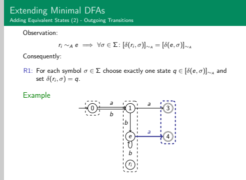
$$r_i \sim_A e \implies \forall \sigma \in \Sigma\colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

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





..dann sind folglich auch diejenigen Zust. äqu., wenn wir ein bel. Symbol von rI und e aus lesen

-

und setze diesen Zustand als Endzustand für delta(rI, sigma)

-

So konnten wir mit dieser Regel erreichen, dass jede Transition von rI in dieselbe Äqu.kl. führt, die wir err. würden, wenn wir der korresp. Tr. von e folgten

Extending Minimal DFAs

Adding Equivalent States (2) - Outgoing Transitions

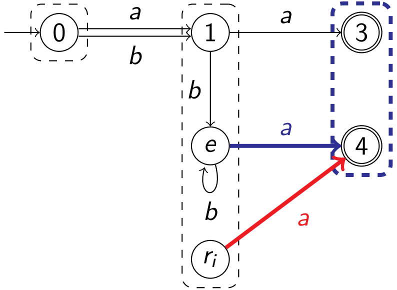
Observation:

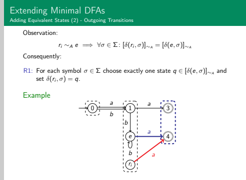
$$r_i \sim_A e \implies \forall \sigma \in \Sigma \colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

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





..dann sind folglich auch diejenigen Zust. äqu., wenn wir ein bel. Symbol von rI und e aus lesen

-

und setze diesen Zustand als Endzustand für delta(rI, sigma)

-

So konnten wir mit dieser Regel erreichen, dass jede Transition von rI in dieselbe Äqu.kl. führt, die wir err. würden, wenn wir der korresp. Tr. von e folgten

Extending Minimal DFAs

Adding Equivalent States (2) - Outgoing Transitions

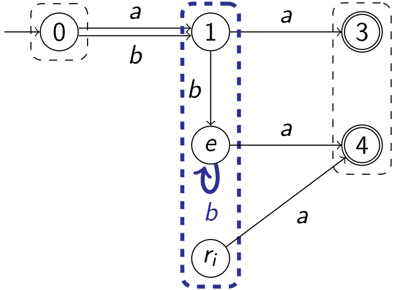
Observation:

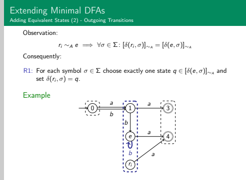
$$r_i \sim_A e \implies \forall \sigma \in \Sigma \colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}$$

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





..dann sind folglich auch diejenigen Zust. äqu., wenn wir ein bel. Symbol von rI und e aus lesen

-

und setze diesen Zustand als Endzustand für delta(rI, sigma)

-

So konnten wir mit dieser Regel erreichen, dass jede Transition von rI in dieselbe Äqu.kl. führt, die wir err. würden, wenn wir der koresp. Tr. von e folgten

Extending Minimal DFAs

Adding Equivalent States (2) - Outgoing Transitions

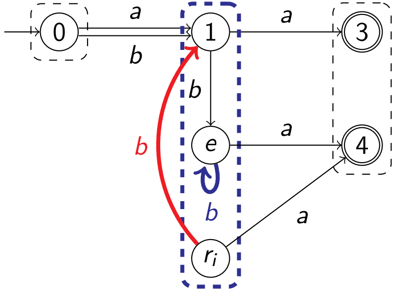
Observation:

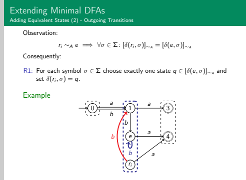
$$r_i \sim_A e \implies \forall \sigma \in \Sigma \colon [\delta(r_i,\sigma)]_{\sim_A} = [\delta(e,\sigma)]_{\sim_A}$$

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





..dann sind folglich auch diejenigen Zust. äqu., wenn wir ein bel. Symbol von rI und e aus lesen

-

und setze diesen Zustand als Endzustand für delta(rI, sigma)

-

So konnten wir mit dieser Regel erreichen, dass jede Transition von rI in dieselbe Äqu.kl. führt, die wir err. würden, wenn wir der korresp. Tr. von e folgten

Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.

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

Extending Minimal DFAs
Adding Equivalent States (3) - Ingoing Transitions
First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.
Let q be a state s.t. $\delta(q, \sigma) = p$ and we want $\delta(q, \sigma) = r_i$.

rI kann nicht Startzustand sein, daher nicht in Erreichbarkeits-Bed. aufgenommen

-

wir müssen Tr. klauen, weil es keine Tr. mit 'losen' Enden gibt- arbeiten mit kompletten DFAs

Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.

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 $[r_i]_{\sim_A} = [e]_{\sim_A}$

Extending Minimal DFAs
Adding Equivalent States (3) - Ingoing Transitions

First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.

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 $[r_i]_{\sim_A} = [e]_{\sim_A}$

Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.

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 $[r_i]_{\sim_A} = [e]_{\sim_A}$

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

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

Extending Minimal DFAs
Adding Equivalent States (3) - Ingoing Transitions
First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.
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 $[r_i]_{\sim_A} = [e]_{\sim_A}$
Furthermore, we see that p must have at least 2 *ingoing elements*.
$in(q) = |d^-(q)| + \begin{cases} 1 & \text{if } s = q \\ 0 & \text{else} \end{cases}$

im Wesentlichen: p muss freie eingehende Tr. haben

Extending Minimal DFAs

Adding Equivalent States (3) - Ingoing Transitions

First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.

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 $[r_i]_{\sim_A} = [e]_{\sim_A}$

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 $in(p) \geq 2$. Remove $((q, \sigma), p)$ from δ and add $((q, \sigma), r_i)$.

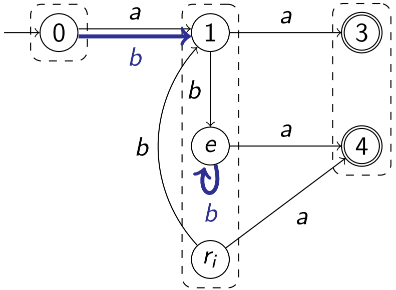
Extending Minimal DFAs
Adding Equivalent States (3) - Ingoing Transitions
First observation: Since r_i must be reachable, we require $|d^-(r_i)| >= 1$.
Let q be a state s.t. $\delta(q, \sigma) = p$ and we want $\delta(q, \sigma) = r_i$.
q must remain in its equivalence class
$\Rightarrow$ p must be in $[r_i]_{\sim_A} = [e]_{\sim_A}$
$\Rightarrow$ q has to have a transition to some state in $[r_i]_{\sim_A} = [e]_{\sim_A}$
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 $in(p) \geq 2$. Remove $((q, \sigma), p)$ from δ and add $((q, \sigma), r_i)$.

Extending Minimal DFAs

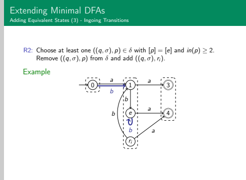
Adding Equivalent States (3) - Ingoing Transitions

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

Example

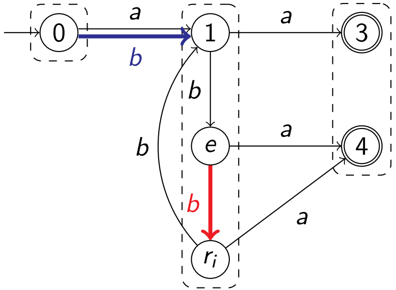


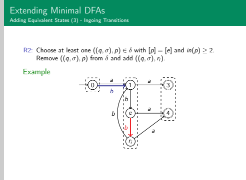
2020-07-16



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

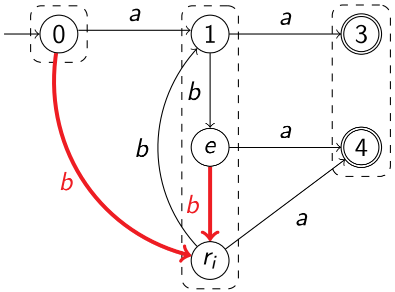
Example





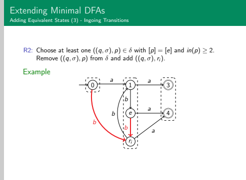
R2: Choose at least one $((q,\sigma),p) \in \delta$ with $[p] = [e]$ and $in(p) \geq 2$.
Remove $((q,\sigma),p)$ from δ and add $((q,\sigma),r_i)$.

Example



2020-07-16

└─Extending Minimal DFAs



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
        steal ingoing tr. from a random subset of $U \times \Sigma$
        add outgoing tr. to $|\Sigma|$ random states
        add q to U
    return A

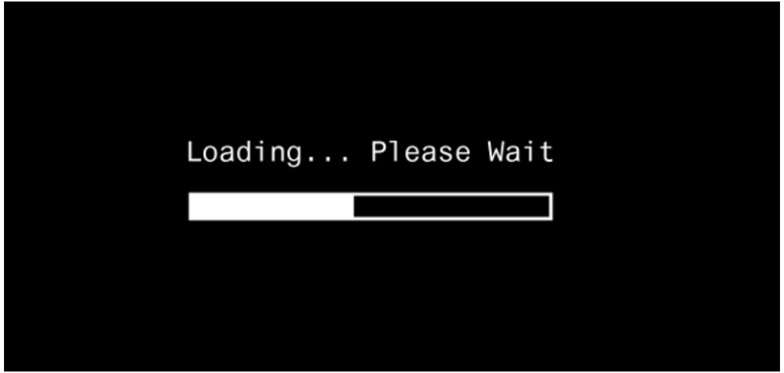
Extending Minimal DFAs
Adding Unreachable States
Reminder: We say a state q is *unreachable*, iff there is no word $w \in \Sigma^*$ such that $\delta^*(s, w) = q$.
function AddUnrStates (A, n_u, c)
$U \leftarrow \emptyset$
for n_u **times do**
let q be the new state
steal ingoing tr. from a random subset of $U \times \Sigma$
add outgoing tr. to $|\Sigma|$ random states
add q to U
return A

Outline

Outline

Introduction
Problem definition and approach
Generating Minimal DFAs
Extending Minimal DFAs
Live Demonstration and Conclusion

Live Demonstration

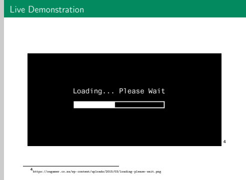


[4]

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

2020-07-16

Automatic Generation of DFA Minimization Problems
└Live Demonstration and Conclusion
└Live Demonstration



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

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)|.
⇒ 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!

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!

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:* $\mathfrak{D}(A) = \mathfrak{D}(A')$.

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: $\mathfrak{D}(A) = \mathfrak{D}(A')$.