

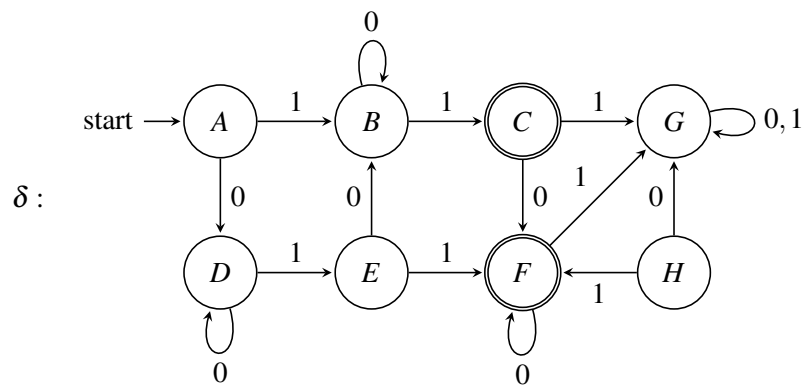
Beispiel für die Minimierung von DEAs

nach dem Verfahren aus [1] (S. 170ff)

Sebastian Wild und Markus E. Nebel

13. März 2012

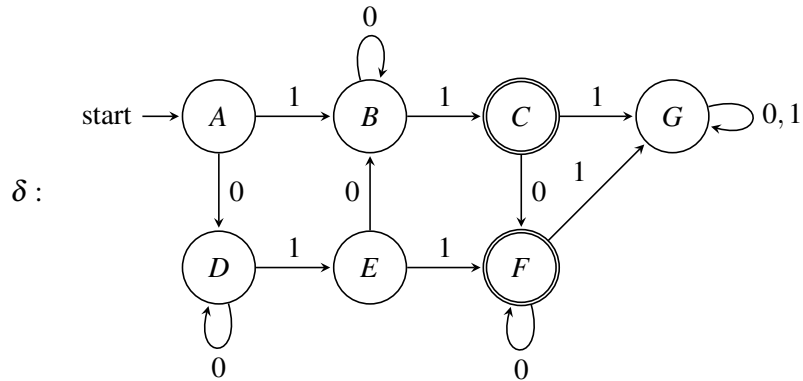
Gegeben sei folgender DEA $A = (\{A, B, C, D, E, F, G, H\}, \{0, 1\}, \delta, A, \{C, F\})$ mit



(1) Nicht erreichbare Zustände verwerfen

Wie im Graph leicht zu erkennen ist, ist der Zustand H von A aus nicht erreichbar [in diesem Fall ganz einfach, weil es überhaupt keine Kante zu H gibt].

Solche nicht erreichbaren Zustände können in einer akzeptierenden Berechnung nicht vorkommen, und sind damit unnütz. Löschen wir sie also einfach aus dem Automaten, so ist der neue Automat $A' = (\{A, B, C, D, E, F, G\}, \{0, 1\}, \delta', A, \{C, F\})$ mit



äquivalent zum alten Automaten A.

(2) \sim bestimmen

Das machen wir hier (zur Präsentation einer Alternative zum Vorgehen aus dem Buch) mittels des *table filling* Algorithmus, der in [1] (S. 170ff) beschrieben ist. Dabei tragen wir statt einem Kreuz¹ jeweils ein Paar $n/v \in \{n/v \mid n \in \mathbb{N} \wedge v \in \mathbb{N} \cup \{-\}\}$ in die Tabelle ein, wenn zwei Zustände als *unterscheidbar* erkannt wurden. Dabei ist n eine fortlaufende Nummer, die angibt, um das ‚wievielte Kreuz‘ es sich hier handelt, $v \in \mathbb{N}$ bezeichnet das ‚Kreuz‘, aus dem gefolgert wurde, dass diese Zustände unterscheidbar sind [siehe dazu das Verfahren unten]. War keine Voraussetzung nötig wählen wir für v das Symbol $-$.

Da der Automat vollständig spezifiziert ist, genügt es mit Paaren aus $E \times (Z \setminus E)$ bzw. $(Z \setminus E) \times E$ zu beginnen. Diese sind offensichtlich unter der Eingabe ε verschieden, denn je ein Zustand akzeptiert diese Eingabe, der andere nicht.

Analog zum Buch nennen wir die Relation, die Zustände [nur] mit Wörtern der Länge $\leq \ell$ unterscheidet $\overset{\ell}{\sim}$.

$\overset{0}{\sim} :$

B						
C	1/-	3/-				
D			5/-			
E			6/-			
F	2/-	4/-		8/-	9/-	
G			7/-			10/-
	A	B	C	D	E	F

Dabei haben wir die Tabelle spaltenweise von links nach rechts ausgefüllt, weshalb die Einträge in ihrer Komponente vor dem / in dieser Reihenfolge fortlaufend nummeriert sind. Da wir in diesem Initialisierungsschritt auf keine Voraussetzungen zurückgreifen, sind die zweiten Komponenten alle $-$.

¹Im Original werden die Felder der Tabelle jeweils mit einem Kreuz versehen, wenn die Zustände getrennt werden. Da wir hier jedoch Zusatzinformationen in der Tabelle vermerken, ist „Kreuz“ als Synonym für „Eintrag“ zu verstehen.

Mit dieser Startrelation/-tabelle führt man nun folgendes Verfahren durch.

Setze m auf die Anzahl Einträge in der Initialtabelle $\overset{0}{\sim}$ und j auf 1.² Wiederhole dann bis alle Einträge $,n/v\text{'}$ abgehakt sind ($,n/v\check{\text{'}}$)

- Setze (X, Y) auf das Zustandspaar, das von $,j/v_j\text{'}$ unterschieden wird.
- Finde alle Zustandspaare U und V , die bei der Verarbeitung *eines einzigen* Symbols *,synchron‘* in Zustand X bzw. Y übergehen und deren zugehörige Zelle noch leer ist. Für diese Paare
 - erhöhe m um 1, und
 - fülle die Zelle für (U, V) mit $,m/j\text{'}$.
- Hake $,j/v_j\text{'}$ ab (setze die Zelle auf $,j/v_j\check{\text{'}}$)
- Erhöhe j um 1.

Indem man sich merkt, bis zu welchen Wert von m Wörter der Länge ℓ betrachtet wurden, kann man die einzelnen $\overset{\ell}{\sim}$ separieren.

Wiederum funktioniert das Verfahren nur dank der vollständigen Spezifiziertheit! Denn sonst können zwei Zustände auch unterscheidbar sein, weil vom einen mit Symbol a eine Kante existiert und vom anderen nicht.

Bis zu $m = 10$ lief bei uns ‚Phase 0‘, in der wir Wörter der Länge 0 betrachtet haben.

$\overset{1}{\sim}$:

B	11/3					
C	1/-✓	3/-✓				
D		13/6	5/-✓			
E	12/4		6/-✓	15/9		
F	2/-✓	4/-✓		8/-✓	9/-✓	
G		14/7	7/-✓		16/10	10/-✓
	A	B	C	D	E	F

Damit lassen sich also 16 Zustandspaare durch Wörter der Länge ≤ 1 unterscheiden.

²Die Variable m ist die oben erwähnte fortlaufende Nummer; sie wird immer die aktuelle Anzahl an eingetragenen ‚Kreuzen‘ angeben. j bezeichnet das Kreuz, aus dem im laufenden Schritt Folgerungen gezogen werden.

\sim^2 :

<i>B</i>	11/3✓					
<i>C</i>	1/-✓	3/-✓				
<i>D</i>		13/6✓	5/-✓			
<i>E</i>	12/4✓		6/-✓	15/9✓		
<i>F</i>	2/-✓	4/-✓		8/-✓	9/-✓	
<i>G</i>	17/14	14/7✓	7/-✓	18/16	16/10✓	10/-✓
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>

Bis einschließlich ‚Kreuz‘ 18 haben wir Wörter der Länge ≤ 2 gebraucht.

 \sim^3 :

<i>B</i>	11/3✓					
<i>C</i>	1/-✓	3/-✓				
<i>D</i>		13/6✓	5/-✓			
<i>E</i>	12/4✓		6/-✓	15/9✓		
<i>F</i>	2/-✓	4/-✓		8/-✓	9/-✓	
<i>G</i>	17/14✓	14/7✓	7/-✓	18/16✓	16/10✓	10/-✓
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>

Die beiden letzten ‚Kreuze‘ namens ‚17/14‘ und ‚18/16‘ erlaubten keine neuen Unterscheidungen und das *table filling* terminiert. Nach dem Argument im Beweis zu Satz 2.3 im Buch ist damit $\sim = \sim^3 = \sim^2$.

Wir haben damit die folgenden Äquivalenzklassen für Z modulo \sim gefunden:

$$\{A, D\}, \{B, E\}, \{C, F\}, \{G\}$$

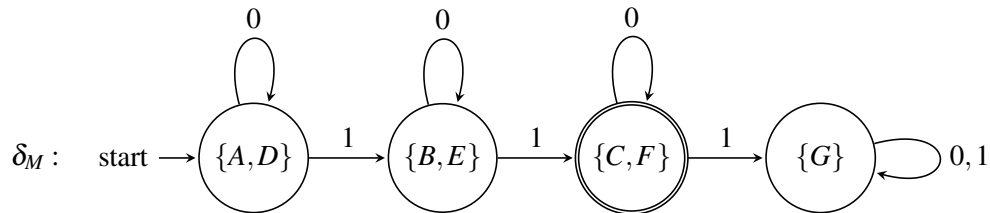
(3) Minimalen Graph zeichnen

Wie in Definition 2.14 beschrieben, verwenden wir nun die Äquivalenzklassen als Zustände.

In unserem Beispiel sieht das dann wie folgt aus

$$A_M = \left(\{ \{A,D\}, \{B,E\}, \{C,F\}, \{G\} \} , \{0,1\} , \delta_M , \{A,D\} , \{ \{C,F\} \} \right)$$

mit



Literatur

- [1] Hopcroft, Motwani, Ullman – *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, 2. Auflage, Addison Wesley Longman 2002



<http://www.springer.com/978-3-8348-1889-8>

Formale Grundlagen der Programmierung

Nebel, M.

2012, VIII, 194 S., Softcover

ISBN: 978-3-8348-1889-8