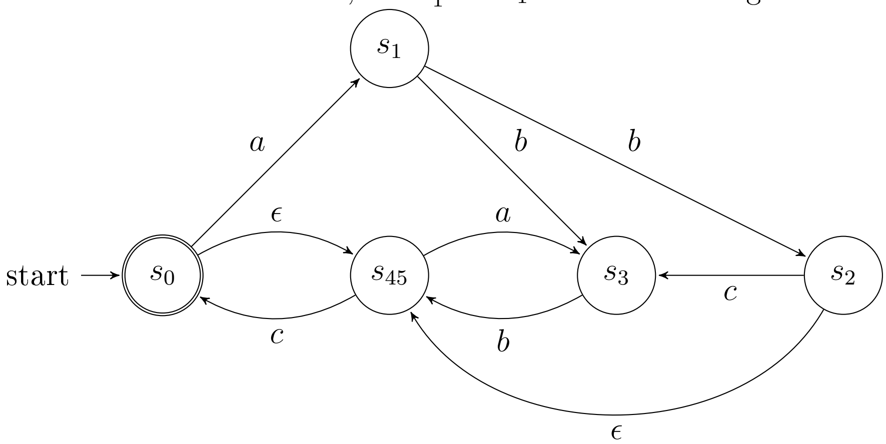
Lehrstuhl für Softwaretechnik und Programmiersprachen
Professor Dr. Michael Leuschel

Compilerbau - Wintersemester 2021/22

Übungsblatt 2 - Musterlösung

Aufgabe 2.1
In Aufgabe 2.1 ist auch eine direkte Umwandlung in einen DFA eine ausreichende Lösung. Der Umgang mit ϵ-Kanten ist hier nur eine mögliche Lösung. In der Klausur erhalten korrekte DFAs alle Punkte

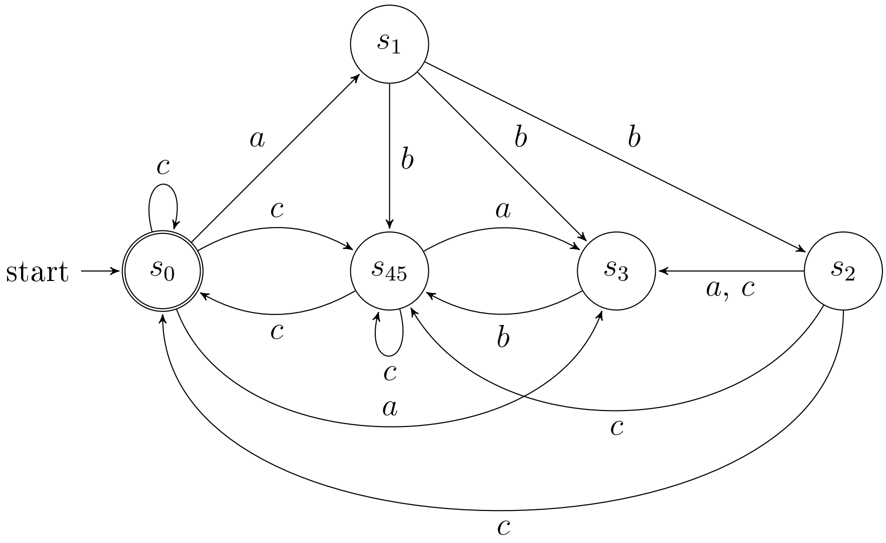
(a) (i) Die Konstruktion dieser NFAs erfolgt nach einem Algorithmus. Es gibt durchaus einfachere Lösungen.
Wir konstruieren zuerst den ϵ-freien NFA $M_1' := (Z_1', \Sigma_1, \delta_1', s_0, E_1')$ zu $M_1 = (Z_1, \Sigma_1, \delta_1, s_0, E_1)$:
Als erstes verschmelzen wir Zustände, die einen ϵ-Zykel bilden, und entfernen die daran beteiligten ϵ-Übergänge. Sollte ein Zustand der zu verschmelzenden Zustände ein Endzustand sein, so ist der neue Zustand auch ein Endzustand. Da dies hier nicht der Fall ist, ist $E_1' := E_1$. Es entsteht folgender Automat:



Nun bilden wir von jedem Zustand $z \in Z_1' := \{s_0, s_1, s_2, s_3, s_{45}\}$ und jedem $s \in \Sigma_1$ den Wert

$$\delta_1'(z, s) := \epsilon\text{-closure}(\delta_1(\epsilon\text{-closure}(z), s)),$$

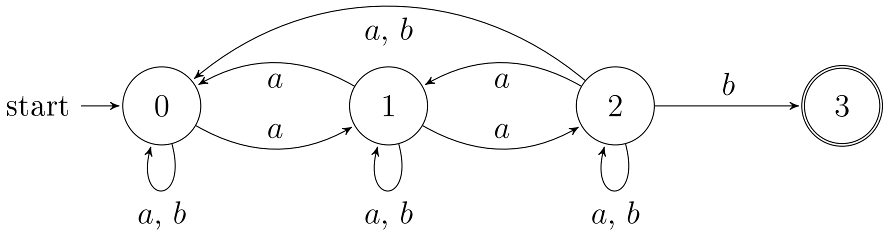
sodass der endgültige NFA M_1' entsteht:



Wir konstruieren jetzt den ϵ-freien NFA $M_2' := (Z_2', \Sigma_2, \delta_2', 0, E_2)$ zu $M_2 = (Z_2, \Sigma_2, \delta_2, 0, E_2)$:
Da es keine ϵ-Zykel gibt, ist $E_2' := E_2$ und wir bilden direkt von jedem Zustand $z \in Z_2' := \{0, 1, 2, 3\}$ und jedem $s \in \Sigma_2$ den Wert

$$\delta_2' (z, s) := \epsilon\text{-closure} (\delta_2 (\epsilon\text{-closure} (z) , s)) ,$$

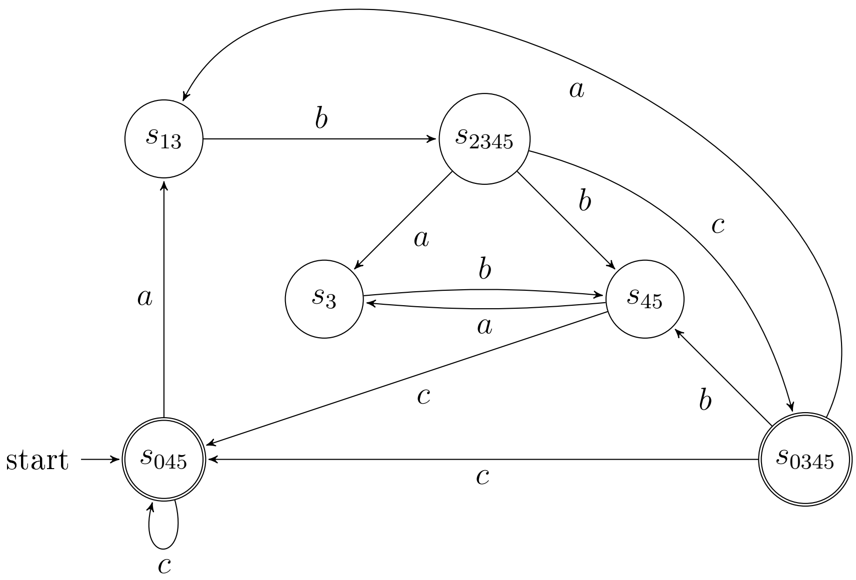
sodass der endgültige NFA M_2' entsteht:



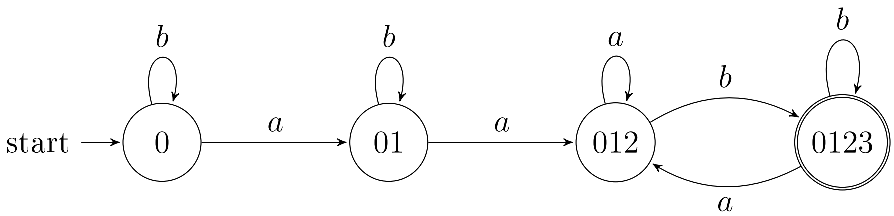
(ii) Der Algorithmus, um aus einem beliebigen NFA $N := (Z, \Sigma, \delta, z_0, E)$ einen DFA $N' := (Z', \Sigma, \delta', z_0', E')$ zu konstruieren, lautet folgendermaßen:

1) Bilde $z_0' := \epsilon\text{-closure} (z_0)$.
2) Definiere die Menge $Z' := \{z_0'\}$ und $E' := \{\}$.
3) Für alle $T \in Z'$ und alle $a \in \Sigma$ führe folgende Schritte aus:
   3.1) Bilde $U := \epsilon\text{-closure} (\text{moveTo} (T, a))$.
   3.2) Füge U der Menge Z' hinzu.
   3.3) Definiere $\delta' (T, a) := U$.
4) Wiederhole Schritt 3 solange, bis sich die Menge Z' nicht mehr ändert.
5) Für alle $T \in Z'$ mache Folgendes:
   5.1) Ist $T \cap E \neq \emptyset$, füge T der Menge E' hinzu.
6) Füge ggf. Müllzüstände hinzu, um eine totale Überführungsfunktion zu gewährleisten.

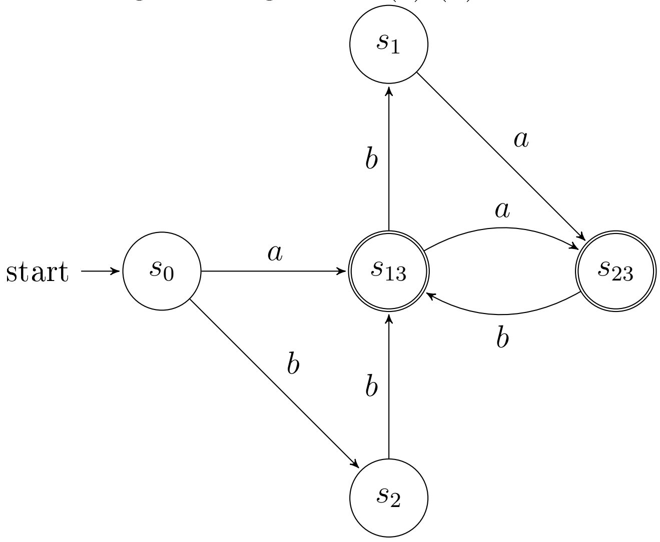
Zuerst bilden wir den DFA D_1 aus M_1:



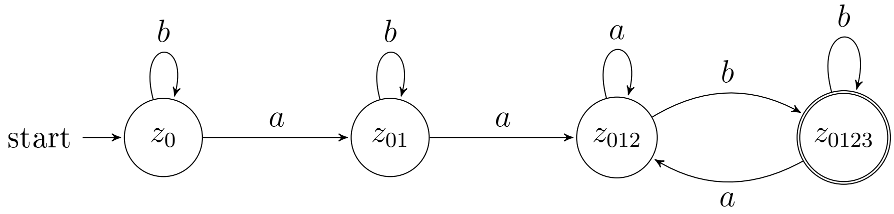
Jetzt konstruieren wir den DFA D_2 aus M_2:



(b) Wir konstruieren den zu NFA M äquivalenten DFA N nach dem Algorithmus aus der Lösung zur Aufgabe 2.1 (a) (ii), sodass dieser folgendermaßen aussieht:



Aufgabe 2.2
Der aus dem NFA M nach obigem Algorithmus konstruierte DFA M' sieht folgendermaßen aus:



Ein schneller Algorithmus zur Minimierung eines DFA N ist der Folgende: Sei der DFA $N = (\Sigma, Z, \delta, z_0, F)$ gegeben. Für jedes ungeordnete Paar Zustände $\{z, z'\}$, wobei beide Zustände entweder in F oder nicht in F sein müssen, wird z und z' mit jedem Buchstaben $s \in \Sigma$ verknüpft. Die Zustände sind genau dann äquivalent, wenn für jedes $s \in \Sigma$ eine der vier folgenden Bedingungen zutrifft:

- Die Zustände zeigen aufeinander.
- Die Zustände zeigen jeweils auf sich selbst.
- Die Zustände zeigen auf den gleichen Zustand.
- Die Zustände zeigen auf 2 äquivalente Zustände (kann meistens nicht sofort gezeigt werden, deswegen mehrmalige Überprüfung).

Hinweis 1: Es muss ggf. ein Müllzustand hinzugefügt werden, damit der Minimalautomat eine totale Überführungsfunktion besitzt.
Hinweis 2: Es können **nie** $z \in F$ **und** $z' \notin F$ **äquivalent** sein.
Nachdem dieser Algorithmus angewandt wurde, erkennt man, dass der DFA M' schon minimal ist.

Aufgabe 2.3
Das C-Programm gibt Folgendes aus:
3
2
Ein Compiler würde folgende Tokenliste erstellen (entsprechend leicht modifiziertem Programm Tokenizer aus Aufgabe 1.1):
LINE 1
(KEYWORD , 'int')
(IDENTIFIER , 'a')
(SIGN , '(')
(IDENTIFIER , 'n')
(SIGN , ')')
(SIGN , '{')
(IDENTIFIER , 'return')
(SIGN , '(')
(IDENTIFIER , 'n')
(OP , '+')

(CONSTANT , '1')
(SIGN , ')')
(SIGN , ';')
(SIGN , '}')
LINE 2
(KEYWORD , 'int')
(IDENTIFIER , 'x')
(OP , '=')
(CONSTANT , '2')
(SIGN , ';')
LINE 3
(KEYWORD , 'void')
(IDENTIFIER , 'b')
(SIGN , '(')
(SIGN , ')')
(SIGN , '{')
(IDENTIFIER , 'x')
(OP , '=')
(IDENTIFIER , 'a')
(SIGN , '(')
(IDENTIFIER , 'x')
(SIGN , ')')
(SIGN , ';')
(IDENTIFIER , 'printf')
(SIGN , '(')
(CONSTANT , '"%d\n"')
(SIGN , ',')
(IDENTIFIER , 'x')
(SIGN , ')')
(SIGN , ';')
(SIGN , '}')
LINE 4
(KEYWORD , 'void')
(IDENTIFIER , 'c')
(SIGN , '(')
(SIGN , ')')
(SIGN , '{')
(KEYWORD , 'int')
(IDENTIFIER , 'x')
(OP , '=')
(CONSTANT , '1')
(SIGN , ';')
(IDENTIFIER , 'printf')

(SIGN , '(')
(CONSTANT , '"%d\n"')
(SIGN , ',')
(IDENTIFIER , 'a')
(SIGN , '(')
(IDENTIFIER , 'x')
(SIGN , ')')
(SIGN , ')')
(SIGN , ';')
(SIGN , '}')
LINE 5
(KEYWORD , 'int')
(IDENTIFIER , 'main')
(SIGN , '(')
(SIGN , ')')
(SIGN , '{')
(IDENTIFIER , 'b')
(SIGN , '(')
(SIGN , ')')
(SIGN , ';')
(IDENTIFIER , 'c')
(SIGN , '(')
(SIGN , ')')
(SIGN , ';')
(SIGN , '}')

Aufgabe 2.4
Der NFA M_5 akzeptiert die Sprache

$$L(\gamma) = \{uavc \mid u \in \{a\}^*, v \in \{a, b\}^*\}\,,$$

wobei

$$\gamma := a^* a(a|b)^* c.$$

Aufgabe 2.5
Der NFA M_6 ist bereits ein minimaler DFA.