Prof. Dr. Andreas Podelski
Dr. Matthias Heizmann
Christian Schilling

Abgabe: 24. November 2017

5. Übungsblatt zur Vorlesung
Informatik III

Aufgabe 1: Reguläre Ausdrücke 2,5 Punkte

Geben Sie reguläre Ausdrücke an, welche die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b\}$ beschreiben.

(a) $L_1 = \{w \in \Sigma^* \mid$ auf jedes a in w folgt direkt ein b $\}$

(b) $L_2 = \{w \in \Sigma^* \mid$ w enthält das Teilwort $bb\}$

(c) $L_3 = \{w \in \Sigma^* \mid$ w enthält das Teilwort bb nicht$\}$

(d) $L_4 = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ enthält genau zweimal das Symbol } a \text{ oder } w \text{ enthält} \\ \text{genau einmal das Symbol } b \end{array} \right\}$

(e) Die Sprache der Wörter mit einer geraden Anzahl b's am Ende:

$L_5 = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} \text{die Länge des längsten Suffixes von } w, \text{ welches nur aus} \\ b\text{'s besteht, ist gerade} \end{array} \right\}$

Aufgabe 2: Regulärer Ausdruck $\rightsquigarrow$ endlicher Automat 2 Punkte

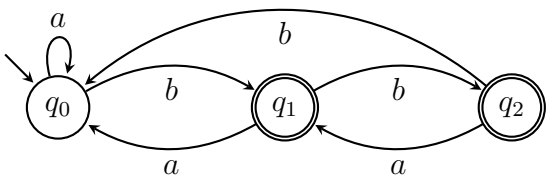
Betrachten Sie den folgenden regulären Ausdruck über $\Sigma = \{a, b\}$.

$$(b \cdot (a \cdot b))^*$$

Konstruieren Sie zu diesem Ausdruck einen äquivalenten endlichen Automaten. Verwenden Sie jeweils die im Skript vorgestellten Konstruktionen (Satz 2.12 „$\Leftarrow$“). Zustände dürfen Sie zur Vereinfachung umbenennen.
Es genügt, den resultierenden Automaten anzugeben. Wie immer gilt: Je gründlicher sie die einzelnen Schritte dokumentieren, desto eher bekommen Sie im Falle eines Fehlers noch Teilpunkte.

Aufgabe 3: Endlicher Automat $\rightsquigarrow$ regulärer Ausdruck 3 Punkte

Betrachten Sie den folgenden DEA $\mathcal{A}$ über $\Sigma = \{a, b\}$.



Bestimmen Sie für $\mathcal{A}$ das Gleichungssystem analog zur Vorlesung (Skript vor Bsp. 2.23). Berechnen Sie anschließend einen äquivalenten regulären Ausdruck, indem Sie das Gleichungssystem nach r_0 auflösen (Beweis zu Satz 2.12 „⇒“).

Sie dürfen reguläre Ausdrücke α, β, γ folgendermaßen vereinfachen. Für die Operationen „Konkatenation“ und „Oder“ gelten die folgenden Regeln:

$$\begin{aligned}
\text{Assoziativität:} &\quad \alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma, \;\; \alpha(\beta\gamma) = (\alpha\beta)\gamma \\
\text{Kommutativität:} &\quad \alpha + \beta = \beta + \alpha \\
\text{Neutrale Elemente:} &\quad \emptyset + \alpha = \alpha, \;\; \varepsilon\alpha = \alpha, \;\; \alpha\varepsilon = \alpha \\
\text{Distributivität:} &\quad \alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma, \;\; (\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma \\
\text{Absorption:} &\quad \emptyset\alpha = \emptyset, \;\; \alpha\emptyset = \emptyset
\end{aligned}$$

Für den Sternoperator gelten die folgenden Regeln:

$$\varepsilon^* = \varepsilon, \;\; (\varepsilon + \alpha)^* = \alpha^*, \;\; (\varepsilon + \alpha)\alpha^* = \alpha^*, \;\; \alpha^*(\varepsilon + \alpha) = \alpha^*$$

Aufgabe 4: Ableitung in einer Grammatik 1 Punkt

Gegeben sei die folgende Grammatik $G = (\Sigma, N, P, S)$ mit der Menge der Terminalsymbole $\Sigma = \{a\}$, der Menge der Nichtterminalsymbole $N = \{S, T, A, B, C, D, E, F, G\}$, dem Startsymbol S und den folgenden Regeln in P:

$$(1)\; S \to BT$$

(2)	T	$\to$	$AT \mid DCE$	(5)	AD	$\to$	FD	(8)	CD	$\to$	GD	(11)	B	$\to$ a
(3)	E	$\to$	$DCE \mid a$	(6)	FD	$\to$	FCA	(9)	GD	$\to$	GC	(12)	A	$\to$ a
(4)	BD	$\to$	BC	(7)	FC	$\to$	DC	(10)	GC	$\to$	DC	(13)	C	$\to$ a

Dabei ist $X \to Y \mid Z$ eine Abkürzung für $X \to Y, X \to Z$.
Geben Sie eine Ableitung für das Wort *aaaa* in der Grammatik an (mit allen Zwischenschritten).

Aufgabe 5: Reguläre Ausdrücke 2 Punkte

Betrachten Sie das Alphabet $A = \{a_1, \ldots, a_n\}$. Geben Sie eine kontextfreie Grammatik an, die die Menge der (vollständig geklammerten) regulären Ausdrücke über A erzeugt. Benutzen Sie dazu die folgenden Terminalsymbole:

$$\Sigma = A \cup \left\{ \boxed{\emptyset}, \boxed{\varepsilon}, \boxed{+}, \boxed{\cdot}, \boxed{^*}, \boxed{(}, \boxed{)} \right\}$$

Aufgabe 6: grep 2,5 Punkte

Das Unix Kommandozeilentool `grep` dient dem Finden und Filtern von Zeichenketten. Das Tool is auf allen gängingen Linuxdistributionen vorinstalliert und Sie haben es in der Vorlesung bereits kurz gesehen. Für einen regulären Ausdrucks `REGEXP` betrachtet der Befehl

```
grep -Er "REGEXP" .
```

alle Dateien im aktuellen Verzeichnis und dessen Unterverzeichnissen und liefert jeweils alle Zeilen, die ein Wort enthalten, das in der Sprache von `REGEXP` liegt.
Ihre Aufgabe ist es nun, für jede der folgenden Teilaufgaben einen regulären Ausruck für `grep` anzugeben. Wir wollen dabei `grep` mit dem Argument `-E` verwenden (`E` für *Extended* Regular Expressions).
Die Syntax von `grep` ist sehr reichhaltig. Wir wollen uns deshalb auf die folgenden Konstrukte beschränken.

Einzelne ASCII-Zeichen	Entsprechen den Elementen des Alphabets. Dabei müssen Sonderzeichen typischerweise mit einem Backslash maskiert werden. Wir schreiben also z.B. „`\?`“ statt „`?`“.
Punkt „.“	Beschreibt ein beliebiges Zeichen.
Runde Klammern „(“, „)“	Entsprechen den runden Klammern in der Notation der Vorlesung.
Strich-Operator „|“	Entspricht dem Plus-Operator für reguläre Ausdrücke aus der Vorlesung.
Stern-Operator „*“	Entspricht dem Stern-Operator für reguläre Ausdrücke aus der Vorlesung.
Eckige Klammern „[“, „]“	Mit eckigen Klammern lässt sich eine *Zeichenauswahl* beschreiben. So steht z.B. `[A-Za-z0-9]` für einen beliebigen Buchstaben oder eine beliebige Ziffer.
Hutsymbol „^“	Zeilenanfang
Dollarsymbol „$“	Zeilenende

(a) Zeilen, die eine Freiburger Telefonnummer enthalten.

Eine Freiburger Telefonnummer beginnt mit `+49` oder `0`, anschließend kommt die Folge `761` und anschließend eine beliebig lange Folge von Ziffern.

(b) Zeilen, die einen *schönen Domainnamen* aus Fidschi enthalten.

Ein schöner Domainname besteht nur aus kleinen Buchstaben, Ziffern und Punkten. Dabei darf der Domainname nicht mit einem Punkt beginnen und es dürfen niemals zwei Punkte aufeinander folgen. Außerdem soll am Ende die Top-Level-Domain von Fidschi `.fj` stehen.

(c) Zeilen, die eine *schöne E-Mailadresse* aus Fidschi enthalten.

Eine schöne E-Mailadresse beginnt mit einer nicht leeren Folge, die aus kleinen Buchstaben, Ziffern und dem Minuszeichen besteht. Es folgt das At-Symbol (@) und ein schöner Domainname aus Fidschi.

(d) Zeilen, die höchstens Leerzeichen enthalten.

(e) Zeilen, die einen HTML-Tag ohne Attribut enthalten.

HTML-Tags beginnen und enden mit spitzen Klammern. Ein HTML-Tag ohne Attribut hat zwischen diesen Klammern genau eine nicht leere Folge von Buchstaben und Ziffern. Zwischen dieser Folge und den Klammern ist aber noch eine beliebig lange Folge von Leerzeichen erlaubt. Beispiele sind `<h1>`, `< title >`.