Theoretische Informatik 1
Übungsblatt 6

René Maseli
Prof. Dr. Roland Meyer
TU Braunschweig
Wintersemester 2022/23

Ausgabe: 24.01.2023 Abgabe: 03.02.2023, 09:45

Geben Sie Ihre Lösungen bis Freitag, 03.02 09:45 Uhr, im Vips-Verzeichnis der StudIP-Veranstaltung ab. Fertigen Sie dazu ihre Hausaufgaben direkt in .pdf Form an oder scannen ihre handschriftlichen Hausaufgaben ein. Geben Sie in Gruppen von **4 Personen** ab.

Es handelt sich hierbei um das letzte Übungsblatt, das abgegeben werden muss und bepunktet wird. Der weitere Stoff der Vorlesung ist dennoch klausurrelevant.

Aufgabe 1: Die Syntax einer Programmiersprache als Grammatik [8 Punkte]
Die Syntax von Programmiersprachen ist üblicherweise als kontextfreie Grammatik (oft dargestellt als EBNF oder Syntaxdiagramm) definiert. In dieser Aufgabe sollen Sie eine Grammatik konstruieren, welche die Syntax einer einfachen Programmiersprache beschreibt.

a) [2 Punkte] Geben Sie eine kontextfreie Grammatik G an, so dass $\mathcal{L}(G)$ die Menge der gemäß der weiter unten erklärten Regeln syntaktisch korrekten Programme ist.

- Verwenden Sie $\Sigma := \{\texttt{id}, \texttt{num}, \texttt{let}, \texttt{if}, \texttt{then}, \texttt{else}, \texttt{end}, \texttt{;}, \texttt{op}, \texttt{=}, \texttt{(}, \texttt{)}\}$.

  Hierbei sind `id`, `num` und `op` jeweils Platzhalter für mögliche Variablennamen, natürliche Zahlen und binäre Operatoren (einschließlich == etc.). Die anderen Symbole repräsentieren jeweils nur ein Zeichen oder Schüsselwort.

- Ein **Ausdruck** besteht aus Variablen, Zahlen und geklammerte binären Operationen, z.B. `(x+2)`, `(z<500)`, `(x*(y/3))`, `(x==(y+1))`.

- Ein **Programm** ist entweder
  - leer
  - eine Variablendefinition (z.B. `let x = (y + 1)`)
  - eine bedingte Anweisung (z.B. `if x then let y=(z/x) end`)
  - eine Fallunterscheidung (z.B. `if x then let y=(z/x) else let y=z end`)
  - eine durch `;` getrennte Verkettung von zwei Programmen (z.B. `var x; x=500`)

b) [2 Punkte] Geben Sie eine vollständige Ableitung in Ihrer Grammatik aus Teil a) vom Startsymbol zum folgenden Programm an:
`let x=10; let y=0; if (y < x) then let y=((y*2)+1) end`
(Sie müssen hierzu zunächst die Variablen durch `id`, die Zahlen durch `num` und die Operatoren durch `op` ersetzen.)

c) [2 Punkte] Nutzen Sie das Pumping-Lemma um zu zeigen, dass $\mathcal{L}(G)$ nicht regulär ist.

d) [2 Punkte] Modifizieren Sie G zu einer weiteren Grammatik G', sodass die Programmiersprache Funktionen unterstützt.

Eine **Funktion** beginnt mit dem Schlüsselwort `function` und einem Funktionsnamen, gefolgt von einer durch `,` getrennten Liste von Parametern (potentiell leer), gefolgt von einem Funktionsrumpf (einem Programm), gefolgt vom Schlüsselwort `end`. Im Funktionsrumpf darf die Rückkehr-Anweisung (z.B. `return (x*x)`) benutzt werden.

Funktionsaufrufe dürfen als Anweisungen und Ausdrücke benutzt werden.

Beispielsweise soll folgendes Wort ein valides Programm sein:

```
function f (x) var y; y=2; return g(x,y) end;
function g (x,y) if (x<y) return x else return y end end;
f(4);
```

Aufgabe 2: CFG, CNF, CYK [10 Punkte]

Der Cocke-Younger-Kasami-Algorithmus (CYK-Algorithmus) erwartet als Eingabe eine kontextfreie Grammatik (CFG) in Chomsky-Normalform (CNF). Dies bedeutet, dass alle Produktionsregeln von der Form $X \to YZ$ (für Nichtterminale Y und Z) oder von der Form $X \to a$ (für ein Terminal a) sind.

Betrachten Sie die zwei CFG $G = \langle \{S, W, X\}, \{a, b\}, P_G, S\rangle$ und $H = \langle \{S, U, V\}, \{a, b, c\}, P_H, S\rangle$.

$$\begin{aligned} P_G : S &\to \varepsilon \mid bW \\ W &\to a \mid XXb \\ X &\to SS \mid ab \end{aligned}$$

a) [1 Punkt] Verwenden Sie das Verfahren aus der Vorlesung, um eine Grammatik G_a ohne ε-Produktionen zu berechnen, die $\mathcal{L}(G_a) = \mathcal{L}(G) \setminus \{\varepsilon\}$ erfüllt.

b) [1 Punkt] Verwenden Sie G_a und das Verfahren aus der Vorlesung, um eine Grammatik G_b in CNF zu berechnen, welche $\mathcal{L}(G_b) = \mathcal{L}(G) \setminus \{\varepsilon\}$ erfüllt.

c) [1 Punkt] Benutzen Sie G_b und den CYK-Algorithmus, um zu entscheiden, ob das Wort *bbaab* von G erzeugt wird.

d) [3 Punkte] Entscheiden Sie mit Hilfe von G_b und des CYK-Algorithmus, ob $bbababb \in \mathcal{L}(G)$ gilt.

$$\begin{aligned} P_H : S &\to UVab \mid bU \\ U &\to aV \mid aUSc \\ V &\to \varepsilon \mid bSc \mid U \end{aligned}$$

e) [1 Punkt] Verwenden Sie das Verfahren aus der Vorlesung, um eine Grammatik H_e in CNF zu berechnen, die $\mathcal{L}(G_a) = \mathcal{L}(G) \setminus \{\varepsilon\}$ erfüllt.

f) [3 Punkte] Benutzen Sie H_e und den CYK-Algorithmus, um zu entscheiden, ob das Wort *aaabca* von H erzeugt wird.

Aufgabe 3: Pushdown-Automaten [10 Punkte]

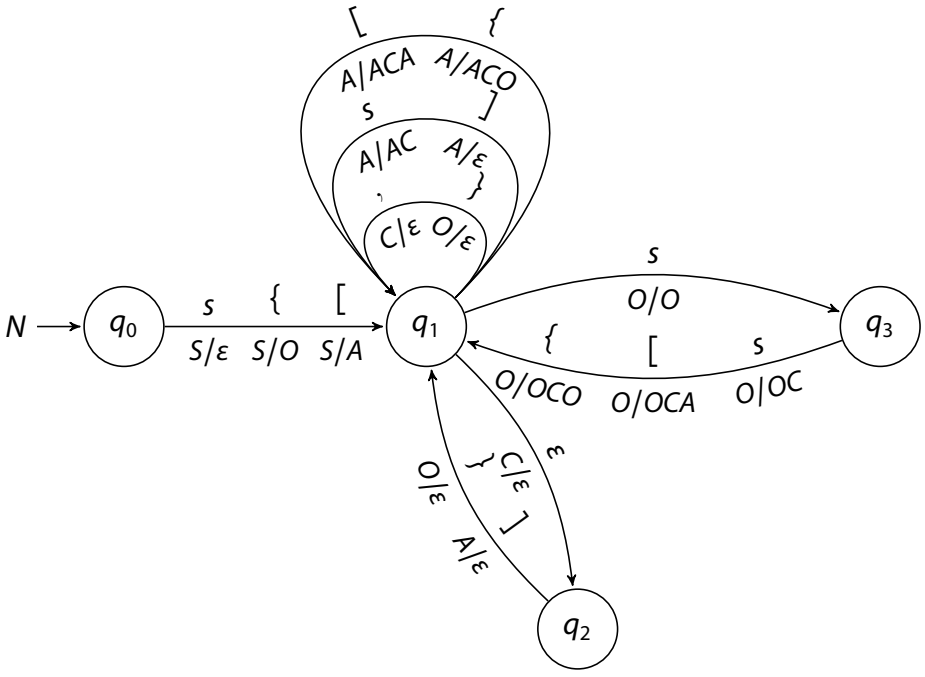
JavaScript Object Notation (JSON) ist eine Beschreibungssprache für strukturierte Sammlungen serialisierbarer Daten, die in vielen Web-Technologien zum Einsatz kommt. Dabei werden neben primitiven Datentypen auch Listen (Arrays) und Assoziative Container (Objekte) ausgedrückt.

a) [5 Punkte] Konstruieren Sie Pushdown-Automaten M für die folgende Sprache L und geben Sie jeweils an, welche Akzeptanzbedingung Sie annehmen (leerer Stack oder Finalzustände). Es reicht nicht, nur eine kontextfreie Grammatik anzugeben. Ein Beweis zur Korrektheit wird nicht benötigt.

Wir betrachten eine vereinfachte Variante von JSON über dem Alphabet $\{a, b, \{, \}\}$: ‚Objekte' sind zwischen geschweiften Klammern ‚{' und ‚}' eingeschlossen. Darin befinden sich beliebig viele Schlüssel-Wert-Paare. Schlüssel sind Worte aus $a.b^*$ und müssen innerhalb eines Objekts nicht eindeutig sein. Werte sind entweder Worte aus $a.b^*$, oder wiederum Objekte. Der Automat M soll ausschließlich wohlgeformte Objekte akzeptieren.

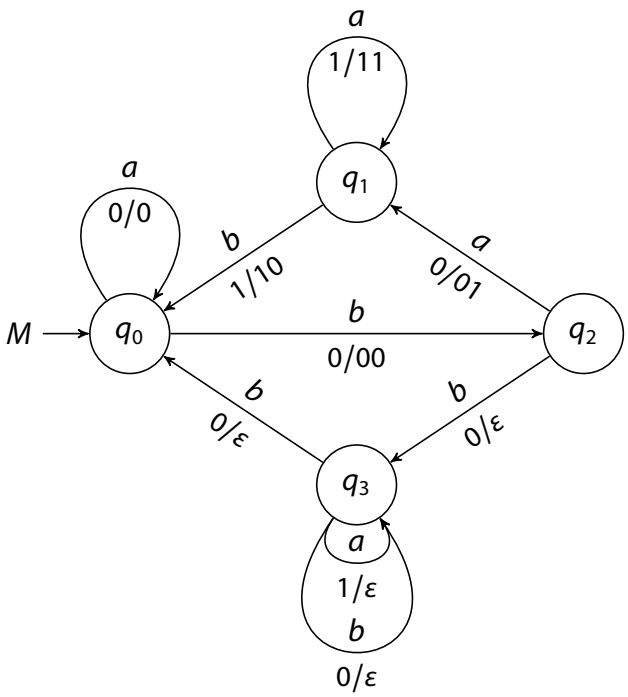
Zum Beispiel sollen $\{abbababb\{\}\} \in L$ und $\{abb\{ab\{aba\}a\{abbab\}\}\} \in L$ akzeptiert werden, aber nicht $\{ababab\} \notin L$, $abb\{aa\} \notin L$ oder $\{ab\{\} \notin L$.

b) [5 Punkte] Beschreiben Sie die Funktionsweise des folgenden Pushdown-Automaten $N = \langle\{q_0, q_1, q_2, q_3\}, \{s, \{, \}, ,, [,]\}, \{C, A, O, S\}, q_0, S, \delta\rangle$, welcher mit leerem Stack akzeptiert, indem Sie die Rolle jedes Zustandes und jedes Stack-Symbols mit jeweils einem Satz erklären.



Aufgabe 4: Tripel-Konstruktion [7 Punkte]

Betrachten Sie den Pushdown-Automaten $M = \langle\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{0, 1\}, q_0, 0, \delta\rangle$, der mit leerem Stack akzeptiert und dessen Transitionsrelation δ durch das folgende Diagramm definiert ist.



a) [1 Punkt] Betrachten Sie nur die einzelnen Zustände, um die folgenden beiden Fragen zu beantworten. Welche zwei Zustände kommen als Ziel $q \in Q$ eines Tripels $\langle p, s, q\rangle$ in Frage? Welche fünf Paare aus $p \in Q$ und $s \in \Gamma$ könnten auftauchen?

b) [6 Punkte] Verwenden Sie die Tripelkonstruktion aus der Vorlesung, um eine kontextfreie Grammatik G mit $\mathcal{L}(M) = \mathcal{L}(G)$ zu bestimmen.