Theoretische Informatik 1
Übungsblatt 6

René Maseli
Thomas Haas
TU Braunschweig
Wintersemester 2023/24

Ausgabe: 2024-01-19 Abgabe: 2024-02-01 23:59

Geben Sie Ihre Lösungen bis Donnerstag, 01.02.2024 23:59 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 und geben Sie **alle** Gruppenmitglieder mit **Matrikelnummer, Namen und Studiengang** an.

Hausaufgabe 1: Die Syntax einer Programmiersprache als Grammatik [9 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) [3 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 $\Sigma := \{\texttt{id}, \texttt{num}, \texttt{;}, \texttt{op}, \texttt{=}, \texttt{(}, \texttt{)}, \texttt{if}, \texttt{else}, \texttt{while}, \texttt{end}, \texttt{break}\}$.
  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. `x=(y+1)`)
  - eine bedingte Verzweigung (z.B. `if x y=(z/x) else y=z end`)
  - eine Schleife (z.B. `while (x<100) x=(x*12) end`)
  - ein Ausbruch `break` **nur innerhalb einer Schleife**
  - eine durch `;` getrennte Verkettung von Programmen (z.B. `x=12 ; y=500`)

b) [1 Punkt] Geben Sie eine Ableitung in Ihrer Grammatik aus Teil a) vom Startsymbol zum folgenden Programm an. Geben Sie außer dem Startsymbol und dem Wort, mindestens drei Satzformen aus Ihrer Ableitungssequenz an.
`while (x<y) x=(x<<1) ; if (y==z) break else y=(y+1) end 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) [3 Punkte] Modifizieren Sie G zu einer weiteren Grammatik G', sodass die Programmiersprache offensichtlich unerreichbaren Code nicht erlaubt: `break` springt aus der Programmsequenz heraus. Hinter so einer Anweisung darf sich kein Code befinden. Auch Verzweigung zählen dazu, falls beide Kontroll-Pfade mit `break` enden. Das kann sich sogar beliebig verschachteln.

Hausaufgabe 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.

a) [6 Punkte] Verwenden Sie das Verfahren aus der Vorlesung, um eine Grammatik G_a in CNF zu berechnen, die $\mathcal{L}(G_a) = \mathcal{L}(G) \setminus \{\varepsilon\}$ erfüllt. Die Grammatik $G = \langle\{S, X, Y\}, \{a, b, c\}, P, S\rangle$ sei dabei durch die folgenden Produktionen definiert:

$$S \to XcX \qquad X \to a \mid YX \mid YbYb \qquad Y \to bc \mid X \mid XX$$

Nutzen Sie Ihre Grammatik und den CYK-Algorithmus, um zu entscheiden, ob das Wort *abcbca* von G erzeugt wird.

b) [4 Punkte] Entscheiden Sie mit Hilfe des CYK-Algorithmus, ob die Wörter *babaa* und *baba* von der Grammatik mit den folgenden Produktionen erzeugt wird.

$$S \to AB \mid BC \qquad A \to a \mid CC \qquad B \to b \mid BA \qquad C \to a \mid AB$$

Hausaufgabe 3: Greibach-Normalform [10 Punkte]

Überführen Sie die folgende Grammatik G in die GNF.

$$S \to WS \mid UU \qquad U \to b \mid c \mid UW \qquad V \to c \qquad W \to a \mid VW \mid VU$$

a) [4 Punkte] Geben Sie für jedes Paar aus Nichtterminal $X \in N$ und Terminal $s \in \Sigma$ einen regulären Ausdruck für die Sprache $L_{X,s} \subseteq N^*$ der Reste von Satzformen $\alpha \in N^*$ an, die durch die starke Linksableitung erzeugt werden: $X \Rightarrow_{SL}^* s\alpha$.

b) [2 Punkte] Finden Sie für alle nichtleeren Sprachen $L_{X,s}$ eine rechts-lineare Grammatik $G_{X,s}$ über **Terminale** N und Startsymbol $T_{X,s}$. Erzeugen Sie neue Nichtterminale, falls erforderlich.

c) [2 Punkte] Überführen Sie nun die Vereinigung von G und all diesen Grammatiken in Pseudo-GNF, indem Sie diese neue Grammatik G' jedes Terminalsymbol erraten lassen, wie in der Vorlesung gezeigt. Sie müssen nicht beweisen, aber sich daran halten, dass $\mathcal{L}(G') = \mathcal{L}(G)$ gilt. Beschränken Sie sich auf die nützlichen Nichtterminale.

d) [2 Punkte] Eliminieren Sie die ε-Produktionen aus G', sodass nun eine Grammatik G'' in GNF entsteht, die $\mathcal{L}(G'') = \mathcal{L}(G') \setminus \{\varepsilon\}$ erfüllt.

Hausaufgabe 4: Pushdown-Automaten [11 Punkte]

Konstruieren Sie Pushdown-Automaten für folgende Sprachen und geben Sie jeweils an, welche Akzeptanzbedingung Sie annehmen (leerer Stack oder Finalzustände).

Hinweis Beachten Sie, dass beim Pushen mehrerer Symbole das Letzte zum neuen Top wird.

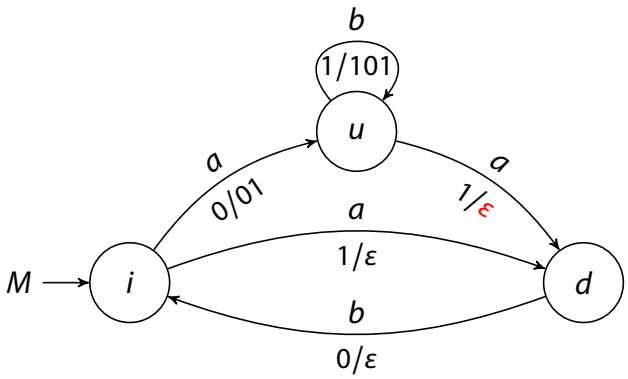
a) [2 Punkte] $L_1 = \{ w \in \{a, b, (,)\}^* \mid w \text{ ist korrekt geklammert } \}$.

b) [2 Punkte] $L_2 = \{ w \in \{a, b, (,)\}^* \mid |w|_a = 2|w|_b \}$.

c) [2 Punkte] Können Sie auch einen PDA bauen, der $L_1 \cap L_2$ akzeptiert?
Wenn nein, was ist intuitiv das Problem hier?

$$L_1 \cap L_2 = \{ w \in \{a, b, (,)\}^* \mid |w|_a = 2|w|_b \text{ und } w \text{ ist korrekt geklammert } \}$$

d) [5 Punkte] Betrachten Sie den Pushdown-Automaten $M = \langle \{i, u, d\}, \{a, b\}, \{0, 1\}, i, 0, \delta \rangle$, der mit leerem Stack akzeptiert und dessen Transitionsrelation δ durch das folgende Diagramm definiert ist.
Verwenden Sie die Tripelkonstruktion aus der Vorlesung, um eine kontextfreie Grammatik G mit $\mathcal{L}(M) = \mathcal{L}(G)$ zu bestimmen.

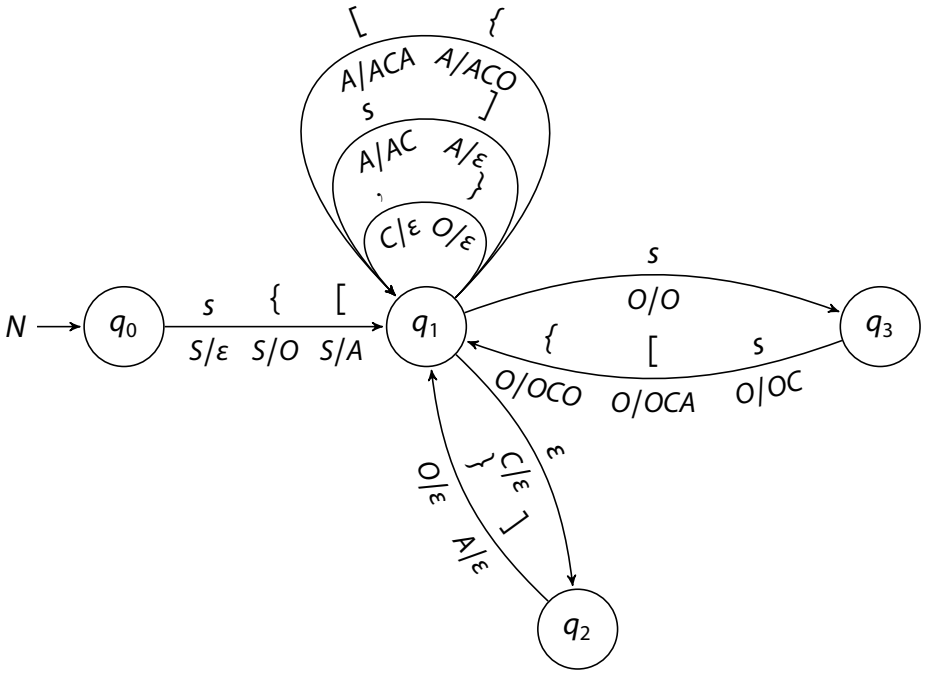


Übungsaufgabe 5:

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.

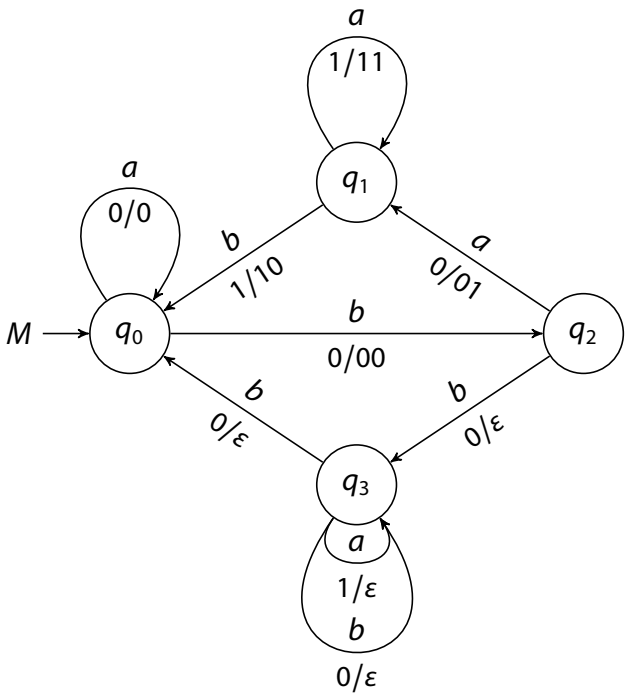
a) Konstruieren Sie Pushdown-Automaten M für die folgende Sprache L und geben Sie jeweils an, welche Akzeptanzbedingung Sie annehmen (leerer Stack oder Finalzustände). Es reicht nicht, nur eine kontextfreie Grammatik anzugeben. Ein Beweis zur Korrektheit wird nicht benötigt.
Wir betrachten eine vereinfachte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



Übungsaufgabe 6:

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

Übungsaufgabe 7:

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

$$P_G : S \to \varepsilon \mid bW$$
$$W \to a \mid XXb$$
$$X \to SS \mid ab$$

a) 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) 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) Benutzen Sie G_b und den CYK-Algorithmus, um zu entscheiden, ob das Wort *bbaab* von G erzeugt wird.

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

$$P_H : S \to UVab \mid bU$$
$$U \to aV \mid aUSc$$
$$V \to \varepsilon \mid bSc \mid U$$

e) 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) Benutzen Sie H_e und den CYK-Algorithmus, um zu entscheiden, ob das Wort *aaabca* von H erzeugt wird.