Theoretische Informatik 1
Übungsblatt 6

Prof. Dr. Roland Meyer
René Maseli
TU Braunschweig
Wintersemester 2025/26

Ausgabe: 2026-01-17 Abgabe: 2026-01-29 23:56

Hausaufgabe 6.1: Die Syntax einer Programmiersprache als Grammatik [4 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) [1 Punkt] 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 $\Sigma := \{\texttt{id}, \texttt{num}, \texttt{;}, \texttt{op}, \texttt{=}, \texttt{(}, \texttt{)}, \texttt{if}, \texttt{else}, \texttt{while}, \texttt{end}, \texttt{break}\}$.
- Ein **Ausdruck** besteht aus Variablen, Zahlen und geklammerte binären Operationen, z.B. `id`, `(id op num)`, `(id op id)`, `(id op (id op num))`, `(num op (num op num))`.
- Ein **Programm** ist entweder
  - leer
  - eine Variablendefinition (z.B. `id=(id op num)`)
  - eine bedingte Verzweigung (z.B. `if id id=(id op id) else id=id end`)
  - eine Schleife (z.B. `while (id op num) id=(id op num) end`)
  - ein Ausbruch `break` **nur innerhalb einer Schleife**
  - eine durch `;` getrennte Verkettung von Programmen (z.B. `id=num ; id=num`)

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(id op id) id=(id op num); if(id op id) break else id=(id op num) end end
```

c) [1 Punkt] 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.

d) [1 Punkt] Modifizieren Sie G aus Teil a) zu einer weiteren Grammatik G', sodass die Programmiersprache Funktionen unterstützt. Eine **Funktion** beginnt mit dem Schlüsselwort `fn` und einem Funktionsnamen, gefolgt von einer durch `,` getrennten Liste von Parametern (potentiell leer), dann gefolgt von einem Funktionsrumpf (einem Programm) und abgeschlossen vom Schlüsselwort `end`. Im Funktionsrumpf darf die Rückkehr-Anweisung (z.B. `return (id op id)`) benutzt werden. Funktionsaufrufe dürfen als Anweisungen und Ausdrücke benutzt werden. Beispielsweise soll folgendes Wort ein valides Programm sein:

```
fn id(id) id=num; return id(id,id) end;
fn id(id,id) if(id op id) return id else return id end end;
id(num);
```

Hausaufgabe 6.2: CFG, CNF, CYK [5 Punkte]

Sei $G = \langle N, \Sigma, S, P \rangle$ eine kontextfreie Grammatik. Um Unit-Regeln zu eliminieren, berechnet man den kleinster Fixpunkt der folgenden Transformation $F : \mathcal{P}(N \times (N \cup \Sigma)^*) \to \mathcal{P}(N \times (N \cup \Sigma)^*)$ mit $F(X) := P \cup \{ X \to \beta \mid (X \to Y), (Y \to \beta) \in X \}$.

a) [2 Punkte] Sei $G = \langle N, \Sigma, S, P \rangle$ kontextfrei. Zeigen Sie, dass die Grammatik $G'' = \langle N, \Sigma, S, P'' \rangle$ ohne Unit-Produktionen, d.h. mit $P'' := \mathrm{lfp}(F) \setminus \{ X \to Y \mid X, Y \in N \}$, die Gleichung $\mathcal{L}(G) = \mathcal{L}(G'')$ erfüllt.

b) [1 Punkt] Betrachten Sie die folgende Grammatik G_b. Entfernen Sie **zuerst** alle unerreichbaren Nichtterminale, **anschließend** alle unproduktiven Nichtterminale und geben Sie die so entstehende Grammatik an. Sind alle übrigen Nichtterminale nützlich?

$S \to XY \mid bVc$ $\quad$ $T \to cTb \mid ZaZ \mid V$ $\quad$ $U \to a \mid bWb$ $\quad$ $V \to a \mid VVV \mid VXV \mid \varepsilon$

$W \to cTUb \mid ZbZ \mid V$ $\quad$ $X \to aZb \mid cX$ $\quad$ $Y \to b \mid TV$ $\quad$ $Z \to bX \mid XX \mid aZ$

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.

c) [1 Punkt] Verwenden Sie das Verfahren aus der Vorlesung, um eine Grammatik G_c' in CNF zu berechnen, die $\mathcal{L}(G_c') = \mathcal{L}(G_c) \setminus \{\varepsilon\}$ erfüllt. Die Grammatik $G_c = \langle \{S, X, Y\}, \{a, b, c\}, P_c, S \rangle$ sei dabei durch die folgenden Produktionen definiert:

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

d) [1 Punkt] Nutzen Sie den CYK-Algorithmus, um zu prüfen, ob das Wort *bbacbbc* von der folgenden Grammatik $G_d := \langle \{S, T, U, A, B, C\}, \{a, b, c\}, P_d, S \rangle$ erzeugt werden kann:

$S \to BA \mid BC$ $\quad$ $T \to CT \mid BS$ $\quad$ $U \to TA$

$A \to a \mid AC \mid BB$ $\quad$ $B \to b \mid UU$ $\quad$ $C \to c \mid CT$

Hausaufgabe 6.3: Pushdown-Automaten [4 Punkte]

Konstruieren Sie Pushdown-Automaten für die folgende Sprache 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.

$$L := \left\{ w \in (bab)^* \sqcup\!\sqcup (ac)^* \;\middle|\; \forall x, y \in \{a, b, c\}^* : w = x.y \Rightarrow |x|_b \leq |x|_c \right\}$$

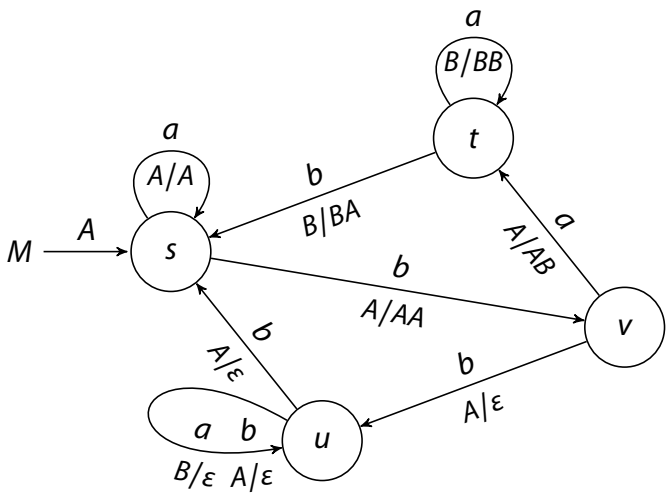
Dabei bezeichne $\sqcup\!\sqcup$ den Shuffle, der in großen Übung 3 vorgestellt wurde. Die zweite Bedingung beschreibt, dass jeder Präfix eines akzeptierten Wortes höchstens so viele b's wie c's enthalten soll.

a) [2 Punkte] Zeichnen Sie den Zustandsgraphen eines PDAs, der L genau akzeptiert.

b) [2 Punkte] Erklären Sie Ihren Automaten in drei Sätzen.

Hausaufgabe 6.4: Tripel-Konstruktion [3 Punkte]

Betrachten Sie den Pushdown-Automaten $M = \langle\{s, t, u, v\}, \{a, b\}, \{A, B\}, s, A, \delta\rangle$, 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) [2 Punkte] Verwenden Sie die Tripelkonstruktion aus der Vorlesung, um eine kontextfreie Grammatik G mit $\mathcal{L}(M) = \mathcal{L}(G)$ zu bestimmen.

Hausaufgabe 6.5: Pumping-Lemma für kontextfreie Sprachen [4 Punkte]

Nutzen Sie das Pumping-Lemma, um zu zeigen, dass die folgende Sprachen nicht kontextfrei ist:

a) [2 Punkte] $L_1 = \{ a^n b a^{2n} b a^{3n} b a^{4n} \mid n \in \mathbb{N} \} \subseteq \{a, b\}^*$.

b) [2 Punkte] $L_2 = \{ u\#v\#w \mid u, v, w \in \{a, b\}^*$ und $(u = w$ oder $v = w) \} \subseteq \{a, b, \#\}^*$.

Übungsaufgabe 6.6:

Betrachten Sie die kontextfreie Grammatik $G = \langle\{S, W, X\}, \{a, b\}, P, S\rangle$ mit den folgenden Regeln

$$S \to \varepsilon \mid bW \qquad W \to a \mid XXb \qquad X \to SS \mid ab\,.$$

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 *babba* von G erzeugt wird.

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

Betrachten Sie die folgende Grammatik G_e.

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

e) Verwenden Sie das Verfahren aus der Vorlesung, um eine Grammatik G_e' in CNF zu berechnen, die $\mathcal{L}(G_e') = \mathcal{L}(G_e) \setminus \{\varepsilon\}$ erfüllt.

f) Benutzen Sie G_e' und den CYK-Algorithmus, um zu entscheiden, ob das Wort *aaabca* von G_e erzeugt wird.

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

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

Übungsaufgabe 6.7:

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

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

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

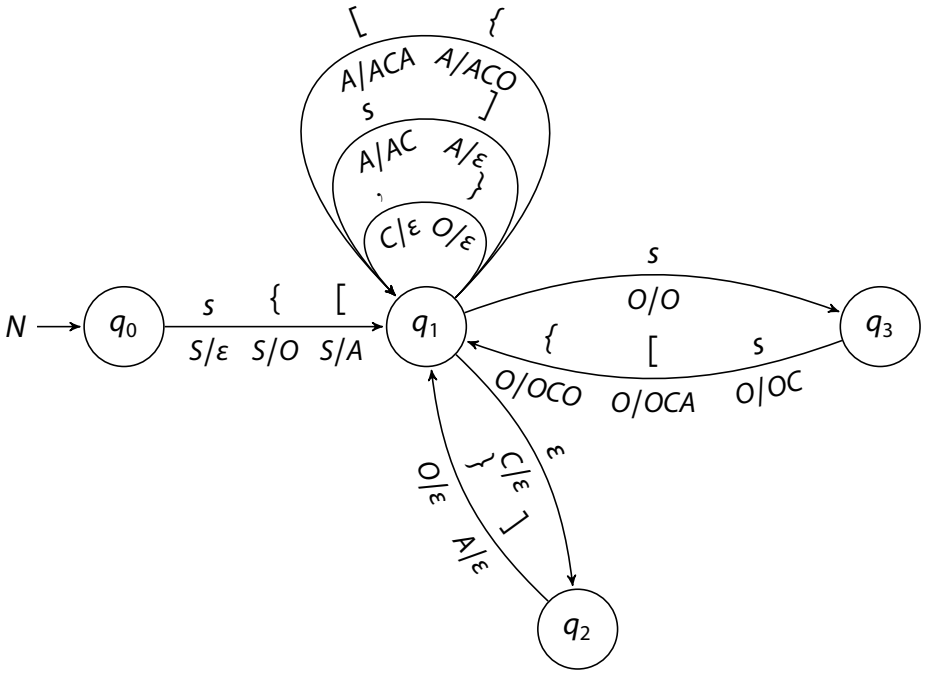
c) Können Sie auch einen PDA bauen, der $L_1 \cap L_2$ akzeptiert?
Falls ja, erklären Sie kurz die Funktionsweise. Falls nein, was ist intuitiv das Problem hier?

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

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.

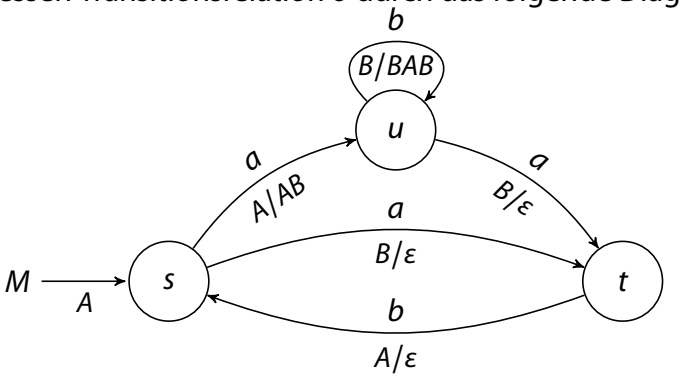
d) Konstruieren Sie Pushdown-Automaten M für die folgende Sprache L und geben Sie jeweils an, welche Akzeptanzbedingung Sie annehmen (leerer Stack oder Finalzustände). Es reicht nicht, nur eine kontextfreie Grammatik anzugeben. Ein Beweis zur Korrektheit wird nicht benötigt.
Wir betrachten eine vereinfachte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

e) 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 6.8:

Betrachten Sie den Pushdown-Automaten $M = \langle\{s, t, u\}, \{a, b\}, \{A, B\}, s, A, \delta\rangle$, 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 Zustände kommen als Ziel $q \in Q$ eines Tripels $\langle p, s, q\rangle$ in Frage? Welche 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 6.9:

Gegeben eine linkslineare Grammatik $G = \langle \Sigma, N, S, P \rangle$ (mit jeweils höchstens einem Nichtterminal als Präfix auf rechten Seiten von Produktionen), sei $A_G = \langle Q, q_0 \rightarrow, Q_F \rangle$ ein NFA mit ε-Übergängen. Dieser besteht aus „originalen" Zuständen $N \subset Q$, einen neuen Startzustand $q_0 \in Q$, einzigem Finalzustand $Q_F = \{S\}$ und Transitionen

$$Y \xrightarrow{w_1} \cdots \xrightarrow{w_n} X \text{ gdw. } X \rightarrow Yw_1 \ldots w_n \in P$$
$$q_0 \xrightarrow{w_1} \cdots \xrightarrow{w_n} X \text{ gdw. } X \rightarrow w_1 \ldots w_n \in P$$

über weiteren neuen Zuständen, falls gebraucht.

Beweisen Sie, dass $\mathcal{L}(A_G) = \mathcal{L}(G)$ gilt.

Übungsaufgabe 6.10:

Überführen Sie die folgende Grammatik G in die GNF.

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

a) 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) 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) 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) 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.

Betrachten Sie nun die folgende Grammatik G in CNF mit

$$S \rightarrow d \mid TU$$
$$T \rightarrow a \mid b \mid UT$$
$$U \rightarrow c \mid d \mid TS \mid US\,.$$

$L_{X,s}$	S	T	U
a	$U \cup S(S \cup TS)^*TU$	$\varepsilon \cup S(S \cup TS)^*T$	$S(S \cup TS)^*$
b	$U \cup S(S \cup TS)^*TU$	$\varepsilon \cup S(S \cup TS)^*T$	$S(S \cup TS)^*$
c	$(S \cup TS)^*TU$	$(S \cup TS)^*T$	$(S \cup TS)^*$
d	$\varepsilon \cup (S \cup TS)^*TU$	$(S \cup TS)^*T$	$(S \cup TS)^*$

e) Nutzen Sie die Sprachen der Nebenprodukte der starken Linksableitung aus der Tabelle, um eine Grammatik G' in GNF mit $\mathcal{L}(G') = \mathcal{L}(G)$ zu konstruieren.