Theoretische Informatik 1
Übungsblatt 6

Sebastian Muskalla
Prof. Dr. Roland Meyer
TU Braunschweig
Wintersemester 2018/19

Ausgabe: 09.01.2019 Abgabe: 17.01.2019, 14:00

Geben Sie Ihre Lösungen bis Donnerstag, 17.01.2019, 14:00 Uhr, durch Einwerfen in die Übungskästen neben Büro IZ 343 ab. 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. Es wird ein weiteres Übungsblatt zur Klausurvorbereitung geben, welches nicht abgegeben werden muss und im Rahmen einer zusätzlichen Großübung besprochen wird.

Aufgabe 1: Kontextfreie Grammatiken

Geben Sie kontextfreie Grammatiken für die folgenden Sprachen an und begründen Sie kurz warum die angegebene Grammatik die Sprache erzeugt.

a) $\mathcal{L}_1 = \{a^n b^m \in \{a,b\}^* \mid n \geqslant m \text{ für } n,m \in \mathbb{N}\}$.

b) $\mathcal{L}_2 = \{w \in \{a,b,c\}^* \mid |w|_a = |w|_b\}$, wobei $|w|_a$ die Anzahl der Vorkommen von Buchstabe a in w ist ($|w|_b$ analog).

c) Können Sie eine kontextfreie Grammatik für $\mathcal{L}_3 = \{a^n b^n c^n \in \{a,b,c\}^* \mid n \in \mathbb{N}\}$ angeben? Wo liegt intuitiv das Problem?

Aufgabe 2: Reguläre Grammatiken

Eine kontextfreie Grammatik G heißt **regulär**, wenn sie linkslinear oder rechtslinear ist. Linkslinear bedeutet, dass alle Produktionsregeln auf ihrer rechten Seite höchstens ein Nichtterminal besitzen, welches (wenn es existiert) das linkeste Symbol ist. Die Regeln sind also alle von der Form $X \to w$ oder $X \to Y.w$ mit $w \in \Sigma^*$. Rechtslinearität ist analog definiert.

In dieser Aufgabe beweisen Sie, dass die Sprachen von regulären Grammatiken genau die regulären Sprachen sind.

a) Beweisen Sie, dass die regulären Sprachen genau die Sprachen sind, die als $\mathcal{L}(G)$ für eine rechtslineare Grammatik auftreten.

   - Erklären Sie, wie man zu einem gegebenen NFA A eine rechtslineare Grammatik G mit $\mathcal{L}(G) = \mathcal{L}(A)$ konstruieren kann.
   - Erklären Sie, wie man zu einer gegebenen rechtslineare Grammatik G einen NFA A mit $\mathcal{L}(G) = \mathcal{L}(A)$ konstruieren kann.

Aufgabenteil b) ist auf der nächsten Seite.

b) Zu einem Wort $w = w_1 w_2 \ldots w_{n-1} w_n$ definieren wir $\text{reverse}(w) = w_n w_{n-1} \ldots w_2 w_1$. Zu einer Sprache $\mathcal{L}$ sei $\text{reverse}(\mathcal{L}) = \{\text{reverse}(w) \mid w \in \mathcal{L}\}$.

- Zeigen Sie, wie man aus einem NFA A einen NFA A' mit $\mathcal{L}(A') = \text{reverse}(\mathcal{L}(A))$ konstruieren kann.
- Zeigen Sie, wie man aus einer kontextfreien Grammatik G eine kontextfreie Grammatik G' mit $\mathcal{L}(G') = \text{reverse}(\mathcal{L}(G))$ konstruieren kann.
- Zeigen Sie, dass jede Sprache der Form $\mathcal{L}(G)$ für eine linkslineare Grammatik G regulär ist. Verwenden Sie hierzu die beiden obigen Punkte und Aufgabenteil a).

Aufgabe 3: CFG, CNF, CYK

Der 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, Z) oder von der Form $X \to a$ (für ein Terminal a) sind.

a) Geben Sie ein Verfahren an, dass eine beliebige kontextfreie Grammatik G in eine sprachäquivalente Grammatik G' in CNF umwandelt.

Sie dürfen davon ausgehen, dass Grammatik G *keine* Regeln der Form $X \to \varepsilon$ hat.

b) Verwenden Sie Ihr Verfahren aus Aufgabenteil a), um eine zur CFG $G = (\{S, X, Y\}, \{a, b, c\}, P, S)$ sprachäquivalente Grammatik in CNF zu berechnen, wobei P durch die folgenden Regeln definiert ist:

$$S \to aXbXc\,,$$
$$X \to Y \mid YYY \mid a\,,$$
$$Y \to bcb\,.$$

c) Entscheiden Sie mit Hilfe des Cocke–Younger–Kasami-Algorithmus, ob die Wörter *babaa* und *baba* von der Grammatik mit den folgenden Regeln erzeugt werden:

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

Aufgabe 4: Die Syntax einer Programmiersprache als Grammatik

In dieser Aufgabe sollen Sie eine Grammatik konstruieren, welche die Syntax einer einfachen Programmiersprache beschreibt.

a) 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 als Terminale id, num, `var`, `if`, `then`, `else`, `end`, `while`, `do`, `;`, `+`, `-`, `*`, `/`, `<`, `>`, `=`, `(`, `)` .

     Hierbei ist „id“ ein Platzhalter für mögliche Variablennamen und „num“ ein Platzhalter für natürliche Zahlen. Die anderen Symbole sind selbsterklärend.

   - Ein **Ausdruck** in der Programmiersprache besteht aus Variablen, Zahlen und Operationen, die diese verknüpfen, z.B. `(x+2)`, `(z<500)`, `(x*(y/3))`, `(x==(y+1))`.

   - Ein **Programm** ist leer, oder eine Variablendeklaration (z.B. `var x;`),
     oder eine Zuweisung eines Ausdrucks an eine Variable (z.B. `x=(x+5);`), oder
     eine bedingte Anweisung (z.B. `if x then y=(z/x); end`), oder
     eine Fallunterscheidung (z.B. `if x then y=(z/x); else y=z; end`), oder
     eine Schleife (z.B. `while x do y=(z/x); else x=(x+1); end`), oder
     eine Verkettung von zwei Programmen (z.B. `var x; x=500;`).

b) Geben Sie die vollständige Ableitung in Ihrer Grammatik aus Teil a) vom Startsymbol zum Programm
   `var x; x=10; var y; y=(x-9); while x do x=(x-1); y=(y+1); end`
   an.

   (Sie müssen hierzu zunächst die Variablen durch id und die Zahlen durch num ersetzen.)

c) Beweisen Sie, dass $\mathcal{L}(G)$ nicht regulär ist.

d) Beschreiben Sie, wie die Grammatik aus Teil a) modifiziert werden muss, damit die Programmiersprache **Funktionen** unterstützt.

   Funktionen haben einen Namen, eine Parameterliste (potentiell leer) und einen Funktionsrumpf. In diesem dürfen `return;` sowie `return` AUSDRUCK`;` benutzt werden. Funktionsaufrufe dürfen als Anweisungen und Ausdrücke benutzt werden.

   Beispielsweise soll folgendes Wort ein valides Programm sein:

```
function f (var x) return (x+1); end
function g () var y; y=2; y=f(y); return; end
g();
```