Diskrete Modellierung

Wintersemester 2019/20

Mario Holldack, M. Sc.
Prof. Dr. Georg Schnitger
Hannes Seiwert, M. Sc.

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Institut für Informatik
AG Theoretische Informatik

Übungsblatt 13

Ausgabe: 30.01.20
Abgabe: 06.02.20

Aufgabe 13.1 *Grenzen der Regularität* $(12 + 12 = 24$ Punkte$)$

Zeigen Sie mit dem Satz von Myhill-Nerode II, dass die Sprachen L_1 und L_2 nicht regulär sind.

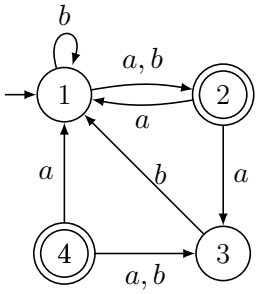
a) $L_1 := \{a^m b^n : m, n \in \mathbb{N},\ m < n\}$

b) $L_2 := \{u\#v : u \in \{a,b\}^*,\ v \in \{a,b\}^*,\ u \neq v\}$

Hinweis: Finden Sie jeweils eine unendliche Menge von Wörtern $\{u_1, u_2, u_3, \ldots\}$, die paarweise inäquivalent bzgl. der Nerode-Relation sind, und weisen Sie die paarweisen Inäquivalenzen $u_i \not\equiv_L u_j$ (für $L \in \{L_1, L_2\}$) durch Angabe geeigneter Zeugen nach.

Aufgabe 13.2 *NFAs* $((3 + 6) + 6 + 9 = 24$ Punkte$)$

a) Sei N der rechts abgebildete NFA über dem Alphabet $\Sigma := \{a, b\}$.



   i) Welche der folgenden Wörter w_i liegen in $L(N)$, welche nicht?

   $w_1 := baa$ $\qquad$ $w_2 := aba$ $\qquad$ $w_3 := baba$

   ii) Konstruieren Sie mittels Potenzmengenkonstruktion einen DFA D, der dieselbe Sprache wie N akzeptiert. Berücksichtigen Sie in D nur Zustände, die vom Startzustand von D aus erreichbar sind.

b) Konstruieren Sie einen NFA für die Sprache des regulären Ausdrucks $(ba)^*(b|\varepsilon)a(b|c)^*$.

c) Zeigen Sie: Jede endliche Sprache ist regulär.

Sie müssen Ihre Antworten in a) und b) nicht begründen.

Aufgabe 13.3 *Reguläre Ausdrücke* $((6 + 6) + (4 + 4 + 4) = 24$ Punkte$)$

a) Gegeben seien die regulären Ausdrücke $R_1 := a^*ba^*(ba^*ba^*)^*$ und $R_2 := (aa|ab|ba)(a|b)^*|a|b|\varepsilon$.

   i) Welche der folgenden Wörter w_i liegen in $L(R_1)$ bzw. $L(R_2)$, welche nicht?

   $w_1 := a$ $\qquad$ $w_2 := bbaaba$ $\qquad$ $w_3 := baba$

   ii) Beschreiben Sie die Sprachen $L(R_1)$ und $L(R_2)$ umgangssprachlich.

b) Geben Sie für die folgenden Sprachen je einen (möglichst kurzen) regulären Ausdruck an.

   i) $L_2 := \{w \in \{0,1\}^* : |w| \text{ ist gerade.}\}$

   ii) $L_1 := \{w \in \{a,b\}^* : w \text{ enthält mindestens ein } a \text{ und ein } b.\}$

   iii) $L_3 := \{w \in \{0,1\}^* : w \text{ enthält } \textit{nicht} \text{ das Teilwort } 11.\}$

Sie müssen Ihre Antworten nicht begründen.

Bitte wenden!

Definition 1: Für ein Alphabet Σ, einen Buchstaben $\sigma \in \Sigma$ und ein Wort $w \in \Sigma^*$ bezeichnet $|w|_\sigma$ die Anzahl der Vorkommen des Buchstabens σ im Wort w. (Zum Beispiel ist $|aaba|_a = 3$, $|aaba|_b = 1$ und $|aaba|_c = 0$.)

Aufgabe 13.4 *Kontextfreie Grammatiken* (8 + (8+12) = 28 Punkte + 15 Extrapunkte)

a) Sei $G := (\Sigma, V, S, P)$ die kontextfreie Grammatik mit $\Sigma := \{a, b\}$, $V := \{S\}$ und

$$P := \{S \to SaSbS \mid SbSaS \mid SS \mid \varepsilon\}$$

Beschreiben Sie die Sprache $L(G)$ umgangssprachlich oder mathematisch.

Eine Begründung ist nicht erforderlich.

b) Sei $k \in \mathbb{N}_{>0}$. Die *Klammersprache* D_k mit k Klammertypen über dem Alphabet

$$\Sigma_k := \{ \langle_1, \langle_2, \ldots, \langle_k \} \cup \{ \rangle_1, \rangle_2, \ldots, \rangle_k \}$$

ist wie folgt rekursiv definiert:

(B) Es gilt $\varepsilon \in D_k$.

(R1) Wenn $w \in D_k$ gilt, dann gilt auch $\langle_i \, w \, \rangle_i \in D_k$ für alle $i \in \{1, \ldots, k\}$.

(R2) Wenn $w_1, w_2 \in D_k$ gilt, dann gilt auch $w_1 w_2 \in D_k$.

i) Geben Sie eine kontextfreie Grammatik an, welche die Klammersprache D_2 für *zwei* Klammertypen beschreibt.
Eine Begründung ist nicht erforderlich.

ii) Die Klammersprache D_1 mit nur einem Klammertyp lässt sich auch nicht-rekursiv charakterisieren. Schreiben Sie a (wie „auf“) für eine öffnende Klammer $\langle_1$ und z (wie „zu“) für eine schließende Klammer $\rangle_1$. Für jedes Wort $w \in D_1$ gilt

$$|w'|_a \geq |w'|_z \text{ für jeden Präfix } w' \text{ von } w, \text{ und } |w|_a = |w|_z. \qquad (*)$$

Der folgende Algorithmus[1] nutzt diese Eigenschaft aus, um zu überprüfen, ob $w \in D_1$ für ein Eingabewort w gilt.

```
# Eingabe: ein Wort w = w[1] ... w[n]
count = 0
for i in range(1,n+1):  # for i in {1, ..., n}
    if w[i] == "a":
        count = count + 1
    if w[i] == "z":
        count = count - 1
    if count < 0:
        return False
return (count == 0)
```

Zeigen Sie mit vollständiger Induktion: Für jedes Wort $w \in D_1$ gibt der obige Algorithmus `True` zurück.

Hinweis: Zeigen Sie mit vollständiger Induktion über die Wortlänge, dass die Eigenschaften $(*)$ für alle Wörter $w \in D_1$ erfüllt sind

Kommentar: Tatsächlich gilt sogar auch die Umkehrung: Für jedes Wort $x \in \Sigma_1^* \setminus D_1$ gibt der Algorithmus `False` zurück.

iii) **Bonusaufgabe**: Zeigen Sie: D_k ist nicht regulär.

[1] Den Quelltext finden Sie auf der Veranstaltungswebseite unter http://thi.cs.uni-frankfurt.de/lehre/dismod/ws1920/dismod_ws1920_blatt13_quelltext.py