Institut für Informatik
der Universität München
Prof. Dr. M. Hofmann
PD Dr. M. Lange
Hermann Gruber

SS 08

21.04.08

# Übungen zur Vorlesung
# **Automatentheorie**

Blatt 2
Besprechung in der Übung am 28.04.08

**Aufgabe 4:** Seien $\mathcal{A}_i = (Q_i, \Sigma, q_{0,i}, \delta_i, F_i)$ für $i = 1, 2$ zwei NFAs. Konstruieren Sie einen NFA $\mathcal{A}$, so dass $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ gilt.

**Aufgabe 5:** Geben Sie jeweils eine MSO[<]-Formel $\varphi(x, y)$ an, die folgenden Sachverhalt beschreibt:

a) $x = y$, (ohne selbst das Symbol = zu benutzen)

b) $y$ liegt $k$-Positionen hinter $x$ für ein festes $k$.

Geben Sie jeweils einen MSO[<]-Satz $\varphi$ an, so dass $L(\varphi)$ die folgende Sprache ist:

c) An jeder $k$-ten Position steht ein $a$ (für festes $k$).

d) Sei $\Sigma = \{a_0, \ldots, a_{n-1}\}$. Auf den Buchstaben $a_i$ folgt jeweils $a_{(i+1) \bmod n}$.

e) Auf jedes $a$ folgt irgendwann ein $b$ und umgekehrt, solange das Wortende noch nicht erreicht ist. Dazwischen stehen nur $c$'s.

f) Das Wort ist eine Permutation von $v$ für ein festes $v \in \Sigma^+$.

Welche dieser Eigenschaften sind bereits FO[<]-definierbar?

**Aufgabe 6:** Geben Sie jeweils einen regulären Ausdruck $\alpha$ an, so dass $L(\alpha) = L(\phi)$ für die folgenden MSO[<]-Formeln $\phi$ über dem Alphabet $\{a, b\}$ gilt.

a) $\phi := \exists x.P_b(x) \wedge \forall y.y < x \rightarrow P_a(y)$

b) $\phi := \forall X.\exists y.\forall x.X(x) \rightarrow x \leq y \wedge P_a(y)$

c) $\phi := \exists X.\exists Y.(\forall z.X(z) \vee Y(z)) \wedge \forall x.\forall y.X(x) \wedge Y(y) \rightarrow x < y \wedge P_a(x) \wedge P_b(y)$

d) $\phi := \forall x.(\exists z.x < z) \rightarrow \exists y.x < y \wedge (P_a(x) \leftrightarrow \neg P_a(y))$

**Aufgabe 7:** Das *Wortproblem* (oder auch *Model Checking Problem*) für MSO ist: gegeben ein Wort $w \in \Sigma^+$ und ein MSO-Satz $\varphi$, entscheide ob $w \in L(\varphi)$ gilt oder nicht.

Geben Sie einen Algorithmus an, der das Wortproblem für MSO löst, dessen Platzverbrauch aber polynomiell in der Größe der Eingabe ($|w|+|\varphi|$) beschränkt ist — auf die Laufzeit hingegen brauchen Sie nicht besonders zu achten.

*Hinweis:* Beschreiben Sie zunächst eine speicher-effiziente rekursive Methode `models`$(\varphi, w, I)$, die zu einem gegebenen Satz $\varphi$, einem Wort $w$ und einer Interpretation $I$ prüft, ob $w \models_I \varphi$. Es reicht dann, `models` für endlich viele Interpretationen $I$ aufzurufen, um das Wortproblem zu entscheiden(?).