Applied Automata Theory (WS 2013/2014)	Technische Universität Kaiserslautern

Exercise Sheet 7

Jun.-Prof. Roland Meyer, Georgel Calin — Due: Tue, Dec 10

Exercise 7.1 Reversal-bounded Counter Machines

Consider the code of 2-counter machine M with counters c_1, c_2 initially set to 0:

```
machine M
  l0: inc(c1);   goto l1 // initial control state
  l1: inc(c2);   goto l0
  l1: zero(c1);  goto l2
  l1: dec(c2);   goto l3
  l2: zero(c2);  goto la
  l3: dec(c1);   goto l1
  l3: inc(c2);   goto l4
  l4: dec(c1);   goto l5
  l5: inc(c1);   goto l3
  la: // accepting control state
```

Represent the above code as an automata with $\bigcup_{c\in\{c_1,c_2\}} \{inc(c), dec(c), zero(c)\}$-labeled transitions and determine how many reversals are needed to reach the accepting state.

Exercise 7.2 NBA Languages = ω-regular Languages — (a) not graded

It was discussed in class that ω-regular languages are NBA definable.

(a) Show that if there exists an NBA that accepts $L \subseteq \Sigma^\omega$ then L is ω-regular.

(b) Construct an NBA that accepts $L = (ab + c)^*((aa + b)c)^\omega + (a^*c)^\omega$

Exercise 7.3 Naive Interpretation of NFAs as NBAs

Let $A = (\Sigma, Q, q_0, \to, Q_F)$ be an NFA with $\emptyset \neq L(A) \subseteq \Sigma^+$ and, for any two states $q, q' \in Q$, define $L^{\neq\epsilon}_{q,q'} := \{w \in \Sigma^+ \mid q \xrightarrow{w} q' \text{ in } A\}$. If $L_\omega(A)$ is the ω-regular language accepted by A (interpreted as an NBA), one can **wrongly** believe that $L_\omega(A) = L(A)^\omega$.

(a) Find a counterexample to $L_\omega(A) = L(A)^\omega$ when $\emptyset \neq L^{\neq\epsilon}_{q,q} \subseteq L(A)$ for all $q \in Q_F$.

(b) Argument that if $L(A) = \bigcup_{q,q'\in Q_F} L^{\neq\epsilon}_{q,q'}$ then $L_\omega(A) = L(A)^\omega$ holds.

(c) Show that if $L(A) = L^+$ for some regular language L then $L_\omega(A) = L(A)^\omega$ holds.

Reminder: if $L \subseteq \Sigma^+$ then $L^\omega := \{w_0w_1 \ldots \in \Sigma^\omega \mid w_i \in L$ for all $i \geq 0\}$.

Exercise 7.4 Generalised ω-regular Expressions

Regular expressions over Σ can be extended to encode languages over $\Sigma^* \cup \Sigma^\omega$ as follows:

$$\alpha \ ::= \ \emptyset \mid a \mid \alpha + \alpha \mid \alpha.\alpha \mid \alpha^* \mid \alpha^\omega \qquad \text{with } a \in \Sigma.$$

The language $L_g(\alpha) \subseteq \Sigma^* \cup \Sigma^\omega$ of a generalised ω-regexp is defined recursively:

$$L_g(\emptyset) = \emptyset \qquad L_g(\alpha + \beta) = L_g(\alpha) \cup L_g(\beta)$$
$$L_g(a) = \{a\} \qquad L_g(\alpha.\beta) = (L_g(\alpha) \cap \Sigma^*).L_g(\beta) \cup (L_g(\alpha) \cap \Sigma^\omega).$$

Concatenation of a language $R \subseteq \Sigma^*$ with a subsequent $L \subseteq \Sigma^* \cup \Sigma^\omega$ means

$$R.L := \{u.v \in \Sigma^* \mid u \in R \text{ and } v \in L \cap \Sigma^*\} \cup \{u.v \in \Sigma^\omega \mid u \in R \text{ and } v \in L \cap \Sigma^\omega\}.$$

And, since $\emptyset^* = \{\epsilon\} = \emptyset^\omega$, the Kleene-iteration and ω-iteration require special care:

$$L_g(\alpha^*) = \begin{cases} (L_g(\alpha) \cap \Sigma^\omega) \cup \{\epsilon\} & \text{if } L_g(\alpha) \cap \Sigma^* \subseteq \{\epsilon\} \\ (L_g(\alpha) \cap \Sigma^\omega) \cup (L_g(\alpha) \cap \Sigma^*)^* & \text{otherwise} \end{cases}$$
$$L_g(\alpha^\omega) = \begin{cases} (L_g(\alpha) \cap \Sigma^\omega) \cup (L_g(\alpha) \cap \Sigma^+)^\omega & \text{if } L_g(\alpha) \cap \Sigma^+ \neq \emptyset \\ (L_g(\alpha) \cap \Sigma^\omega) \cup \{\epsilon\} & \text{otherwise.} \end{cases}$$

Your task: show that for every generalised ω-regexp α there is another

$$\alpha' = \gamma + \sum_{i \in I \text{ finite}} \alpha_i.\beta_i^\omega \text{ with } \gamma, \alpha_i, \beta_i \subseteq \Sigma^*,\ \beta_i \cap \Sigma^+ \neq \emptyset \text{ for every } i \in I$$

such that $L_g(\alpha) = L_g(\alpha')$.

Note: generalised ω-regular expressions over Σ^ω describe the ω-regular languages.