Technische Universität München **Winter term 2012/13**
I7
Prof. J. Esparza / J. Křetínský **19.11.2012**

Automata and Formal Languages – Homework 5

Due 26.11.2012.

Exercise 5.1

As seen in the lecture, when applying the post, pre or join operations to transducers the underlying projection operation might yield an automaton which does not accept all possible encodings anymore. You have seen how to fix this problem in the case that the representations are obtained by padding on the right, as for instance in the lsbf-representation of natural numbers where all representations of a number $n \in \mathbb{N}$ are obtained by adding 0s. In the case of the msbf-encoding, the padding does not occur on the right, but on the left. Hence, the procedure given in the lecture cannot be applied anymore.

- Give an algorithm for calculating the "pad-closure" of a transducer when using the msbf-encoding of natural numbers.

Exercise 5.2

With transducers defined to be finite automata whose transitions are labeled by pairs of symbols $(a, b) \in \Sigma \times \Sigma$ only pairs of words $(a_0 a_1 \dots a_l, b_0 b_1 \dots b_l)$ of same length can be accepted. Consider therefore finite automata whose transitions are labeled by elements of $(\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\})$ instead, and call this class ε-transducers. As in the case of transducers, we say that an ε-transducer $\mathcal{A}$ accepts a word pair (w, w') if there is a run

$$q_0 \xrightarrow{(a_0,b_0)} q_1 \xrightarrow{(a_1,b_1)} \dots \xrightarrow{(a_n,b_n)} q_n \text{ with } a_i, b_i \in \Sigma \cup \{\varepsilon\}$$

such that $w = a_0 a_1 \dots a_n$ and $w' = b_0 b_1 \dots b_n$. Note that $|w| \leq n$ and $|w'| \leq n$. As usual, we write $\mathcal{L}(\mathcal{A})$ for the language of word pairs accepted by the ε-transducer $\mathcal{A}$.

(a) Construct ε-transducers $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_1) = \{(a^n b^m, c^{2n}) \mid n, m \geq 0\}$, and $\mathcal{L}(\mathcal{A}_2) = \{(a^n b^m, c^{2m}) \mid n, m \geq 0\}$.

(b) Apply the construction for the intersection of two finite automata to $\mathcal{A}_1$ and $\mathcal{A}_2$. Which language does the resulting ε-transducer accept?

(c) Show that there is no ε-transducer which accepts the language $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Exercise 5.3

Transducers can be also seen as devices transforming input into output. Thus, they can capture the behaviour of simple programs.

Let P be the following program. The domain of the variables is assumed to be $\{0, 1\}$, and the initial value is assumed to be 0. Let $[i, x, y]$ denote the state of P that corresponds to the ith instruction, the value x in `x`, and the value y in `y`.

```
1  x ←?
2  write x
3  do
4    do
5      read y
6    until x = y
7    if eof then
8      write y
9      end
10   do
11     x ← x − 1
       or
12     y ← y + 1
13   until x ≠ y
14 until false
15 end
```

The initial state of the program P is $[1, 0, 0]$. By executing the first instruction, the program can move from state $[1, 0, 0]$ and either enter the state $[2, 0, 0]$ or the state $[2, 1, 0]$. In both cases, no input symbol is read and no output symbol is written during the transition between the states. Hence, the transition relation δ for P contains the transition rules $([1, 0, 0], (\varepsilon, \varepsilon), [2, 0, 0])$ and $([1, 0, 0], (\varepsilon, \varepsilon), [2, 1, 0])$. Similarly, by executing its second instruction, the program P must move from state $[2, 1, 0]$ and enter state $[3, 1, 0]$ while reading nothing and writing 1. Hence, δ contains also the transition rule $([2, 1, 0], (\varepsilon, 1)[3, 1, 0])$.

(a) Draw the ε-transducer that characterizes the program P.

(b) Can an overflow error occur?

(c) What are the possible values of `x` and `y` upon termination, i.e. reaching **end**?

(d) Can a pair of input 101 and output 01 occur?

(e) Let us have a regular set I of inputs and a regular set O of outputs. We may consider O to be the dangerous outputs that we want to avoid and we want to prove that using only I is safe, i.e. none of the dangerous outputs can occur. Describe an algorithm deciding given I and O whether there are $i \in I$ and $o \in O$ such that (i, o) is accepted.