Technische Universität München **Winter term 2009/10**
I7
Prof. J. Esparza / J. Křetínský / M. Luttenberger **1. Februar 2010**

Automata and Formal Languages – Homework 11

Due 28.1.2010.

Exercise 11.1

Recall the definition of star-free expressions σ over a given alphabet Σ (see also exercise 2.4):

$$\sigma ::= a \mid \varepsilon \mid \emptyset \mid \sigma + \sigma \mid \sigma \cap \sigma \mid \overline{\sigma} \mid \sigma \cdot \sigma.$$

You already know that although every star-free expression represents a regular language, not every regular language can be represented by a star-free expression. For instance, by Schützenberger's theorem $\mathcal{L}((aa)^*)$ is not star-free.

- Show how to obtain from any star-free expression σ an MSO(<)-formula ϕ_σ s.t. $\mathcal{L}(\sigma) = \mathcal{L}(\phi_\sigma)$. Does your translation need the full expressiveness of MSO(<)?

Exercise 11.2

In this exercise we study MSO(<) on infinite words. We fix some finite alphabet Σ for the following.

Similar to the case of finite words, we evaluate a formula ϕ on structures $(w, \mathcal{I})$ consisting of an infinite word $w = w_0 w_1 w_2 \ldots \Sigma^*$ and an interpretation $\mathcal{I}$ which maps any free first-order variable x, resp. second-order variable X of ϕ to a position $\mathcal{I}(x) \in \mathbb{N}$, resp. to a set of positions $\mathcal{I}(X) \subseteq \mathbb{N}$.

When interpreting MSO(<) over finite words, we of course quantify the second-order variables over *finite* sets only. When applying it to infinite words or the naturals, we can either keep this kind of quantification (*weak semantics* of MSO(<)), or we can quantify over arbitrary (also infinite) sets which is the usual interpretation of MSO(<) (*full semantics*). We denote the respective satisfaction relation w.r.t. the full, resp. weak semantics by $(w, \mathcal{I}) \models \phi$, resp. $(w, \mathcal{I}) \models_{\mathrm{w}} \phi$. Formally, these are then defined by:

$(w,\mathcal{I}) \models Q_a(x)$	iff	$w[\mathcal{I}(x)] = a$	$(w,\mathcal{I}) \models_{\mathrm{w}} Q_a(x)$	iff	$w[\mathcal{I}(x)] = a$
$(w,\mathcal{I}) \models x \in X$	iff	$\mathcal{I}(x) \in \mathcal{I}(X)$	$(w,\mathcal{I}) \models_{\mathrm{w}} x \in X$	iff	$\mathcal{I}(x) \in \mathcal{I}(X)$
$(w,\mathcal{I}) \models x < y$	iff	$\mathcal{I}(x) < \mathcal{I}(y)$	$(w,\mathcal{I}) \models_{\mathrm{w}} x < y$	iff	$\mathcal{I}(x) < \mathcal{I}(y)$
$(w,\mathcal{I}) \models \neg\phi$	iff	$(w,\mathcal{I}) \not\models \phi$	$(w,\mathcal{I}) \models_{\mathrm{w}} \neg\phi$	iff	$(w,\mathcal{I}) \not\models_w \phi$
$(w,\mathcal{I}) \models (\phi \vee \psi)$	iff	$(w,\mathcal{I}) \models \phi$ or $(w,\mathcal{I}) \models \psi$	$(w,\mathcal{I}) \models_{\mathrm{w}} (\phi \vee \psi)$	iff	$(w,\mathcal{I}) \models_w \phi$ or $(w,\mathcal{I}) \models_w \psi$
$(w,\mathcal{I}) \models \exists x\phi$	iff	$\exists i \in \mathbb{N}\ (w,\mathcal{I}[i/x]) \models \phi$	$(w,\mathcal{I}) \models_{\mathrm{w}} \exists x\phi$	iff	$\exists i \in \mathbb{N}\ (w,\mathcal{I}[i/x]) \models_{\mathrm{w}} \phi$
$(w,\mathcal{I}) \models \exists X\phi$	iff	$\exists S \subseteq \mathbb{N}\ (w,\mathcal{I}[S/X]) \models \phi$	$(w,\mathcal{I}) \models_{\mathrm{w}} \exists X\phi$	iff	$\exists S \subseteq \mathbb{N}\ \lvert S\rvert < \infty \wedge (w,\mathcal{I}[X/S]) \models_w \phi$.

In the lecture you have seen how to encode a structure $(w, \mathcal{I})$ if w is a finite word. This encoding can also be used for an infinite word w. For instance, consider the formula $x \in X \to Q_a(x)$. A structure for this formula is given by

$$w = (ab)^\omega \text{ and } \mathcal{I}(x) = 2, \mathcal{I}(X) = \{i \in \mathbb{N} \mid i \text{ is even }\}.$$

We encode this structure by the infinite word

$$\begin{matrix} w & \to \\ x & \to \\ X & \to \end{matrix} \begin{bmatrix} a \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} a \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \left(\begin{bmatrix} a \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \right)^\omega.$$

We write $\mathcal{L}(\phi)$, resp. $\mathcal{L}_{\mathrm{w}}(\phi)$ for the language consisting of the encodings of all $(w, \mathcal{I})$ satisfying $(w, \mathcal{I}) \models \phi$, resp. $(w, \mathcal{I}) \models_{\mathrm{w}} \phi$. Obviously, we have $\mathcal{L}_{\mathrm{w}}(\phi) \subseteq \mathcal{L}(\phi)$.

(a) Give an MSO(<)-formula finite(X) with one free second-order variable X s.t.

$$(w, \mathcal{I}) \models \text{finite}(X) \text{ iff } \mathcal{I}(X) \text{ is a finite set.}$$

(b) Construct a Büchi automaton $\mathcal{B}$ representing $\mathcal{L}(\text{finite}(X))$.

(c) Let ϕ be an MSO(<)-formula. Show that there exists a Büchi automaton $\mathcal{B}$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\phi)$, resp. $\mathcal{L}(\mathcal{B}) = \mathcal{L}_{\mathrm{w}}(\phi)$

(d) Let $\mathcal{B}$ be a Büchi automaton. Show how to construct an MSO(<)-formula ϕ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\phi)$.

(e) Give an example of an MSO(<)-formula that is a tautology w.r.t. the full semantics, but a contradiction w.r.t. weak semantics.

Exercise 11.3

You have seen in the lecture how to construct a finite automaton which represents all solutions for a given linear inequation

$$a_1x_1 + a_2x_2 + \ldots + a_kx_k \leq b \text{ with } a_1, a_2, \ldots, a_k, b \in \mathbb{Z} \qquad (*)$$

w.r.t. the least-significant-bit-first representation of $\mathbb{N}^k$ (see the algorithm PAtoDFA).

We may also use the most-significant-bit-first (msbf) representation of $\mathbb{N}^k$, e.g.,

$$\text{msbf}\left(\begin{bmatrix} 2 \\ 3 \end{bmatrix}\right) = \mathcal{L}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}^* \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$$

(a) Construct a finite automaton for the inequation $2x - y \leq 2$ w.r.t. the msbf representation.

(b) Try now to adapt the algorithm PAtoDFA to the msbf encoding.

(c) Recall that integers can be encoded as binary strings using two's complement: a binary string $s = b_0b_1b_2\ldots b_n$ is interpreted, assuming msbf, as the integer

$$-b_0 \cdot 2^n + b_1 \cdot 2^{n-1} + b_2 \cdot 2^{n-2} + \ldots + b_n \cdot 2^0.$$

In particular, s and $(b_0)^*s$ represent the same integer. This extends in the standard way to tuples of integers, e.g., the pair $(-3, 5)$ has the following encodings:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}^* \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

- Construct an automaton accepting all (encondings of) *integer* solutions of the inequation $2x - y \leq 2$.
- Extend your algorithm from (b) such that the constructed automaton accepts all two's complement encodings of all integer solutions of $(*)$.