Technische Universität München **Winter term 2009/10**
I7
Prof. J. Esparza / J. Křetínský / M. Luttenberger **11. Februar 2010**

Solution

Automata and Formal Languages – Homework 10

Due 18.1.2010.

Exercise 10.1

We interpret the monadic second order logic over finite words with the standard interpretation of $<$ as less than relation.

Let $MSO'(S)$ be a modification of the standard monadic second-order logic given by the following syntax. Assume a set of second-order logical variables ranged over by X, Y, Z. Let Σ be an alphabet. An $MSO'(<)$ formula over Σ is defined by the following BNF, where $a \in \Sigma$:

$$\varphi ::= X \subseteq Q_a \mid X < Y \mid \mathrm{Sing}(X) \mid X \subseteq Y \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \exists X \varphi$$

Although we quantify over set variables only, we want this logic to be equally "powerful" as the original $MSO(<)$. As there are no first-order variables, the first-order predicates $<$ will be replaced by the second-order predicates, so new atomic formulas are introduced: $\mathrm{Sing}(X)$ (meaning singleton), $X \subseteq Y$ (meaning subset inclusion), $X \subseteq Q_a$ for every $a \in \Sigma$ (meaning all elements of X are labelled by a), and $X < Y$ (true for singletons $X = \{x\}, Y = \{y\}$ satisfying $x < y$).

Show that $MSO(<)$ and $MSO'(<)$ are equally expressive, i.e., a language is definable in $MSO(<)$ iff it is definable in $MSO'(<)$.

Hint: Express the newly defined predicates in the original MSO and vice versa.

Remark: This logic can be used to create a different (a bit easier) procedure to translate formulae into automata: the problem of incorrect encodings does not arise.

Solution: The translation from $MSO(<)$ to $MSO'(<)$ is done inductively using the following rules:

- $\forall x \varphi$ is translated as $\forall X(\mathrm{Sing}(X) \to \varphi)$,
- $\exists x \varphi$ is translated as $\exists X(\mathrm{Sing}(X) \wedge \varphi)$,
- $x < y$ is translated as $\mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \wedge X < Y$,
- $Q_a(x)$ is translated as $\mathrm{Sing}(X) \wedge X \subseteq Q_a$ for each $a \in \Sigma$.

The other direction is done by the following translation:

- $\mathrm{Sing}(X)$ is translated as $\exists x(x \in X \wedge \forall y(y \in X \to y = x))$,
- $X \subseteq Y$ is translated as $\forall x(x \in X \to x \in Y)$,
- $X \subseteq Q_a$ is translated as $\forall x(x \in X \to Q_a(x))$,
- $X < Y$ is translated as $\exists x \exists y(x \in X \wedge y \in Y \wedge \forall z((z \in X \to z = x) \wedge (z \in Y \to z = y) \wedge x < y))$.

Exercise 10.2

We can show that a given formula of MSO holds on some word, i.e., is satisfiable, as follows. Firstly, we transform the given formula into the equivalent automaton. Secondly, we decide whether it has an accepting run (note that the automaton can be input-free and no letters are actually read). The formula is *satisfiable* iff there is such an accepting run.

Moreover, if the given formula is a sentence, i.e., with no free variables, it is a *tautology* iff there is an accepting run. Thus we have a method to prove (and disprove) MSO formulae.

Prove that every nonempty (finite) subset Z of natural numbers has its minimal element, i.e. show that

$$\forall Z \exists x \forall y (y \in Z \to (x \leq y \wedge x \in Z))$$

where $x \leq y$ is a shortcut for $\neg y < x$ is true on all words.

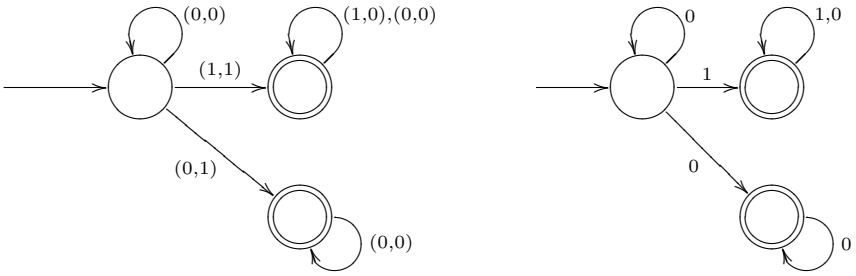
Solution: We proceed inductively by the structure of the equivalent formula

$$\neg\exists Z \neg\exists X \neg\exists Y \neg(\mathrm{Sing}(X) \wedge ((\mathrm{Sing}(Y) \wedge Y \subseteq Z) \to (X \leq Y \wedge X \subseteq Z)))$$

Obviously, an automaton corresponding to $\neg\exists Y \neg(\mathrm{Sing}(X) \wedge ((\mathrm{Sing}(Y) \wedge Y \subseteq Z) \to (X \leq Y \wedge X \subseteq Z)))$ recognizes words of the form

$$\begin{array}{c|cccccc} Z & 0 & 1 & 1 & 0 & 1 & \cdots \\ X & 0 & 1 & 0 & 0 & 0 & \cdots \end{array} \quad \text{or} \quad \begin{array}{c|cccc} Z & 0 & 0 & 0 & \cdots \\ X & 0 & 1 & 0 & \cdots \end{array}$$

and this automaton and the one that we get by projection are



The last automaton obviously has a successful run, thus proving the original formula.

Exercise 10.3

Express the addition using $MSO(<)$. More precisely, find a formula $\mathrm{Plus}(X, Y, Z) \in MSO(<)$ that is true iff $x + y = z$, where x, y, z are numbers encoded by the sets X, Y, Z, respectively, in the binary lsbf. You are allowed to use the successor macro S.

Remarks:

- Thus, we can translate any first-order formula with signature $+$, i.e. any formula of Presburger arithmetic, to an equivalent $MSO(<)$ formula: the first-order quantification is replaced by the subset quantification and $+$ by Plus. The same holds for sentences. Since $MSO(<)$ has already been proved to be decidable (see the translation into automata in the lecture and Exercise 10.2), Presburger arithmetic decidable, i.e. we have an algothm to determine whether a formula is true for every valuation.
- There is also another approach to prove the decidability of Presburger arithmetic: we find the respective automaton directly (see the lecture).

Solution: For this, we simulate the algorithm for summing digit by digit with an additional auxiliary subset T to keep the carry bits. Firstly, we express the formula that the sum on a particular place is correct. This is true iff the odd number of summands carries 1.

$$\mathrm{Sum}(a) \equiv ((a \in X \wedge a \notin Y \wedge a \notin T) \vee (a \notin X \wedge a \in Y \wedge a \notin T)$$
$$\vee (a \notin X \wedge a \notin Y \wedge a \in T) \vee (a \in X \wedge a \in Y \wedge a \in T)) \leftrightarrow a \in Z$$

where $\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$. The formula that the carry bit is 1 is true iff at least two summands carry 1. Note that if overall overflow should happen, the formula is correctly false by the existential quantification.

$$\mathrm{Carry}(a) \equiv ((a \in X \wedge a \in Y) \vee (a \in X \wedge a \in T)$$
$$\vee (a \in Y \wedge a \in T)) \leftrightarrow \exists b (S(a, b) \wedge b \in T)$$

Now, we are able to express the summing of two numbers. It is necessary to ensure that the first carry bit is 0.

$$\mathrm{Plus}(X, Y, Z) = \exists T \forall a (\mathrm{Sum}(a) \wedge \mathrm{Carry}(a) \wedge ((\neg\exists b (S(a, b))) \to a \notin T))$$

Exercise 10.4

Using the algorithms discussed in the lecutre, construct a finite automaton for the Presburger formula

$$\exists y : x = 3y.$$