Technische Universität München
Theoretische Informatik
Prof. J. Esparza / Dr.-Ing. J. Kreiker

Winter term 2008/2009
Exercise Sheet 8
December 4, 2008

Exercises "Automata and Formal Languages"

Exercise 8.1

Consider the MONA example of the n-adder discussed in class and available as a demo on the MONA website. Instead of an adder you are supposed to design an n-multiplexer, which is a circuit implementing a function $MUX_n : \{0,1\}^{2n+1} \to \{0,1\}^n$ such that

$$MUX_n(a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0, s) = \begin{cases} a_{n-1}, \ldots, a_0 & \text{if } s = 1 \\ b_{n-1}, \ldots, b_0 & \text{if } s = 0 \end{cases}$$

Verify your design with MONA. Follow the instructions on the back.

Exercise 8.2

Construct Büchi automata accepting the following languages over $\Sigma = \{a, b, c\}$.

(a) $L_0 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains } ab \text{ exactly once }\}$.

(b) $L_1 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains } ab \text{ at least once }\}$.

(c) $L_2 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains } ab \text{ infinitely often }\}$.

(d) $L_3 = \{\alpha \in \Sigma^\omega \mid \text{contains } ab \text{ only finitely often }\}$.

(e) $L_4 = \{\alpha \in \Sigma^\omega \mid \text{if } \alpha \text{ contains infinitely many } a\text{'s then } \alpha \text{ contains infinitely many } b\text{'s }\}$.

Exercise 8.3

Construct *deterministic* Büchi automata accepting the following languages over $\Sigma = \{a, b, c\}$.

(a) $L_1 = \{\alpha \in \Sigma^\omega \mid \alpha \text{ contains at least one letter } c\}$.

(b) $L_2 = \{\alpha \in \Sigma^\omega \mid \text{in } \alpha, \text{ every } a \text{ is immediately followed by a } b\}$.

(c) $L_3 = \{\alpha \in \Sigma^\omega \mid \text{in } \alpha, \text{ between two successive } a\text{'s there are at least two } b\text{'s }\}$.

Exercise 8.4

Let $B = (\mathcal{A}, G)$ be a Büchi automaton. We say that B co-accepts an input $\alpha : \mathbb{N}_0 \to \Sigma$, if there exists a run ρ of $\mathcal{A}$ on α, such that $\mathsf{inf}(\rho) \cap G = \emptyset$.

(a) Prove: If B co-accepts α then there exists an automaton $B' = (\mathcal{A}', G')$, a run ρ' of $\mathcal{A}'$ on α and an $i \geq 0$ such that $\rho'(j) \in G'$ for all $j \geq i$.

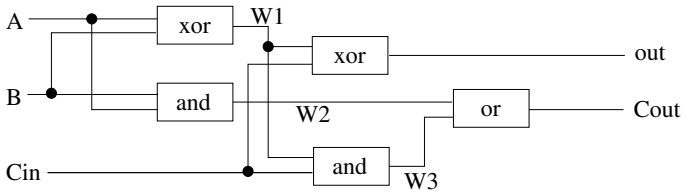
(b) Let $\Sigma = \{a, b\}$. Show that there does not exist a Büchi automaton, $(\mathcal{A}, G)$, that co-accepts the language $\mathcal{L}((b^*a)^\omega)$.

An n-bit Multiplexer in MONA

Understand the Adder

Have a close look at the n-adder example on the MONA website, which is linked from the course website. It may be a good idea to start from this file and modify it to obtain your solution. Here is a list of things to note:

- The first three lines declare an integer variable, `$`, that represents a *maximal* input length, that is, the n of the n-adder. Variables `p` and `P` are declared to be smaller than n (a subset of $\{0, \ldots, n\}$) *globally* for the whole MONA program. Keep these lines for your solution.
- Consider the predicate `full_adder`. Variables of type `var0` are essentially bits. The formula given mimics a digital circuit with input bits `A`, `B`, and `Cin` (the input carry), and output bits `out` and `Cout` (the output carry). The existentially quantified boolean variables `W1`, `W2`, and `W3` are needed to store intermediate results. In the circuit they correspond to the output of a gate. The circuit corresponding to the formula looks as follows:



- Consider the predicate `n_bit_adder(X,Y,Z,Cin,Cout)`. The first three arguments are now sets instead of bits. A set $X = \{0, 2, 4\}$ denotes a binary number whose zeroth, second, and fourth bits are 1, and all others are 0, that is, X represents the number 20. Set variables `C` and `D` represent the input and output carries at each position, while `p` ranges over all bits of the numbers to be added.
- Keep the lines below the comment "theorems". They restrict the second-order variables needed to represent $n + 1$ bit numbers. Instead of two boolean variables `Cin` and `Cout` you will only need a single one, `S`.

Design your MUX

(a) Draw a digital circuit implementing the MUX_1 function. String n such circuits together to obtain a circuit implementing MUX_n.

(b) Write a MONA formula defining the predicate `mux1(A,B,C,S)`, where all arguments are variables of type `var0`, that is, bits. The formula should use the pre-defined circuit constructor relations of the n-adder. It must correspond to the circuit you have drawn.

(c) Write a MONA formula `muxn(X,Y,Z,S)`, where S is still a bit, but where `X`, `Y`, and `Z` are set variables (`var2`), such that it computes MUX_n for an arbitrary n.

Verify your Design

Verify the following formulas with MONA.

(a) `muxn` does indeed compute MUX_n:
```
((S & X=Z) | (not(S) & Y=Z)) <=> muxn(X,Y,Z,S);
```

(b) A concrete example, which computes $MUX_5(19, 23, 0)$:
```
X = {0,1,4} & Y = {0,1,2,4} & (S <=> false) & muxn(X,Y,Z,S);
```

(c) Output can always be computed:
```
all2 X, Y: all0 S: ex2 Z: muxn(X,Y,Z,S);
```

(d) `muxn` is a *function*:
```
all2 X, Y: all0 S: ex2 Z: muxn(X,Y,Z,S) & all2 Z' : muxn(X,Y,Z',S) => (Z = Z');
```