Technische Universität München
Prof. J. Esparza / J. Křetínský
Winter term 2010/11

Name:

Matrikelnummer:

SOLUTION

Automata and Formal Languages – Endterm

***Please note*: If not stated otherwise, all answers have to be justified.**

Exercise 1

each 2P=12P

Each of the following questions admits an answer fitting in one or two lines.

(a) Prove or disprove: "Every regular language is recognized by an NFA with all states final."

Solution: Consider some regular L not containing ε, e.g. $\{a\}$. If all states are final, then the intitial state is final, hence $\varepsilon \in L$, a contradiction.

(b) For every $n \in \mathbb{N}$, let us define a relation $R_n = \{(u, v) \mid lsbf^{-1}(u) = n \cdot lsbf^{-1}(v)\}$.

Assuming R_2 and R_3 are regular, prove that R_6 is regular.

Solution: $R_6 = R_2 \circ R_3$ and the join preserves regularity.

(c) Let $\Sigma = \{a, b\}$ be an alphabet. Give an MSO formula defining the language of $\Sigma(a\Sigma\Sigma)^*a$.

You may use any macros defined in the "Skript".

Solution:

$$\exists X\Big(Triple(X) \land \forall x \in X\, Q_a(x) \land \exists x \in X\, last(x)\Big)$$

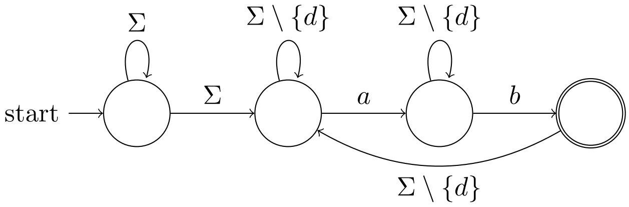
where $Triple(X) \equiv \forall x\Big(x \in X \leftrightarrow \big(\exists y(zero(y) \land x = y + 1) \lor \exists y \in X\, x = y + 3\big)\Big)$

(d) Construct a Büchi automaton recognizing the following language:

$$L = \{w \in \{a, b, c, d\}^\omega \mid a, b \in \inf(w) \text{ and } d \notin \inf(w)\}$$

where $\inf(w)$ denotes the set of letters that appear infinitely many times in w.

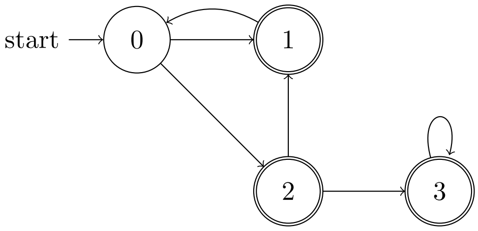
Solution:



(e) Describe a procedure to complement *deterministic* Muller automata directly without using any translation to Büchi automata.

Solution: For $(Q, \Sigma, \delta, q_0, \mathcal{F})$ take $(Q, \Sigma, \delta, q_0, 2^Q \setminus \mathcal{F})$

(f) Consider the following NBA $\mathcal{A}$.



The following sequences are accepting lassos of $\mathcal{A}$. The cycles are underlined.

i) $\underline{010}$

ii) $\underline{0210}$

iii) $02\underline{101}$

iv) $02\underline{33}$

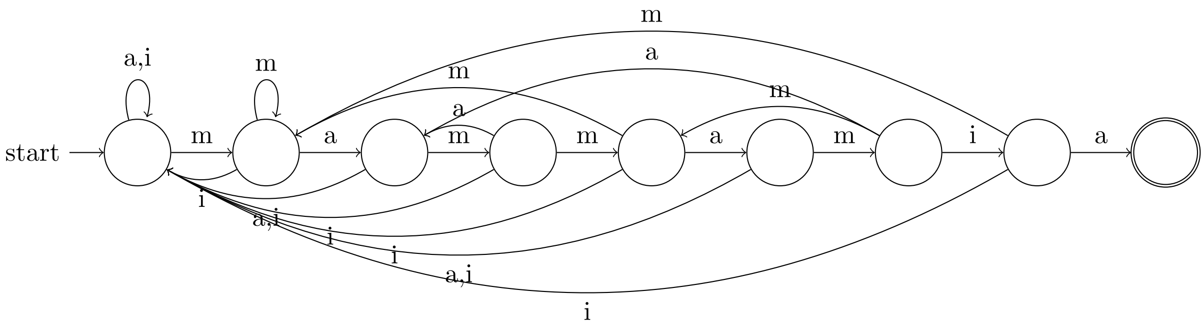
Which of the lassos can be found by a run of NestedDFS on $\mathcal{A}$?

Solution: i,iii,iv. Before ii can be found iii is reported.

Exercise 2 **4P**

Construct an eagerDFA (not a lazyDFA) for the word *mammamia* over the alphabet $\{m, a, i\}$.

Solution:



Exercise 3 **5P**

Consider a variable x with domain $\{0, 1, 2\}$ initialized to 0 and the following program with two parallel processes:

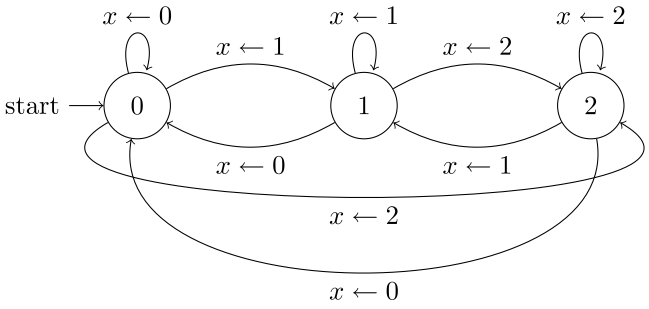
Process 1:		Process 2:	
	loop		**loop**
1:	$x \leftarrow 1$	1:	$x \leftarrow 2$
2:	$x \leftarrow 0$		

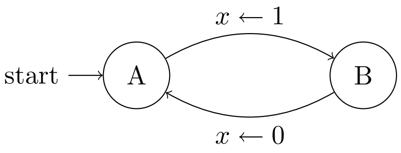
(a) Construct the corresponding network of the three automata and their asynchronous product.

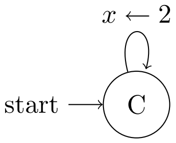
(b) Consider a set of atomic propositions $AP = \{x = 0, x = 1, x = 2\}$. Construct a Büchi automaton over AP corresponding to the property that from some point on $x = 0$ holds forever. Give the corresponding LTL formula, too.

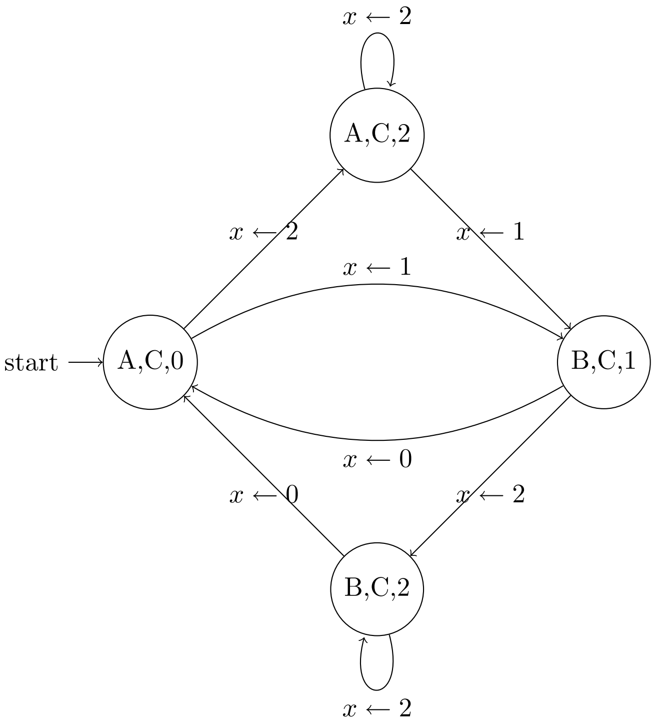
(c) Is there an ω-execution of the program that satisfies the property in (b)? Why?/Why not?

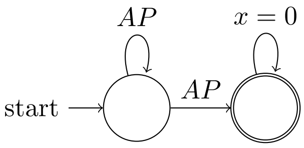
Solution:











$FG(x = 0)$

No, since there is no reachable cycle in the product with the last component being 0 (from A,C,0 we immediately leave to either A,C,2 or B,C,1).

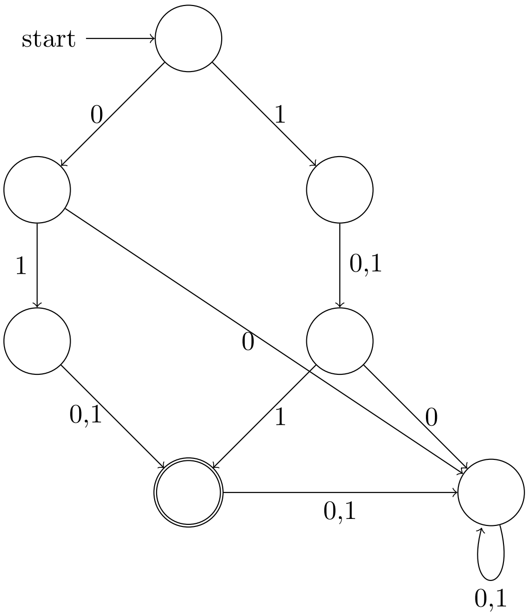
Exercise 4 **5P**

Consider languages over $\{0, 1\}$ with fixed length of 3.

(a) Construct a fragment of the master automaton for the language $L \subseteq \{0, 1\}^3$ of msbf binary encodings of all prime numbers in the range from 0 to 7. (Recall that the smallest prime number is 2.)

(b) Is there a language $L \subseteq \{0,1\}^3$ with a minimal DFA having 10 states?

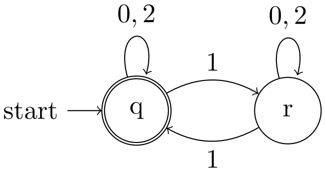
Solution: The numbers are 2,3,5,7 corresponding to 010,011,101,111.



No. Since the alphabet has two elements, every state can have at most two children. Moreover, since the length is fixed to 3, we only have four "layers". Hence we cannot have more than one initial state, its two children, four states in the third layer and one accepting and one rejecting state in the fourth layer, i.e. 1+2+4+1+1=9 states at most.

Exercise 5 **4P**

Prove that the following finite automaton over $\{0, 1, 2\}$ accepts precisely the msbf ternary encodings of even numbers. (E.g. 211 is accepted because $2 \cdot 3^2 + 1 \cdot 3^1 + 1 \cdot 3^0 = 22$ is even.) Proceed by induction on the length of the word.



Solution: For simplicity, we omit the $msbf_3^{-1}$ notation and identify the numbers and their encodings here.

We prove by induction a stronger statement: For every $u \in \{0, 1, 2\}^*$, if $\delta^*(q, u) = q$ then u is even, and if $\delta^*(q, u) = r$ then u is odd.

Induction base: $|u| = 0$. Then $u = \varepsilon$ and $\delta^*(q, \varepsilon) = q$. And indeed, ε is an encoding of zero which is even.

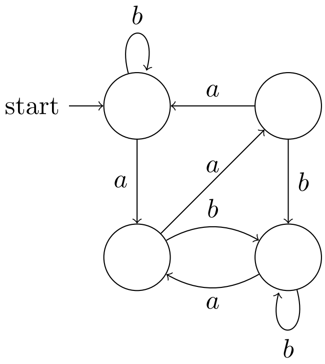
Induction step: Let $|u| = k + 1$ and we can thus write $u = va$ for some $a \in \{0, 1, 2\}$. Note that $u = 3v + a$.

- Either $\delta^*(q, v) = q$ and by induction hypothesis v is even. If $a = 1$ then $\delta^*(q, u) = \delta^*(q, va) = \delta(\delta^*(q, v), a) = \delta(q, 1) = r$ and indeed $u = 3v + a$ is odd (as $3v$ is even). On the other hand if $a \in \{0, 2\}$ then $\delta^*(q, u) = q$ and $u = 3v + a$ is even as both $3v$ and a are even.
- Or $\delta^*(q, v) = r$ and by induction hypothesis v is odd. If $a = 1$ then $\delta^*(q, u) = q$ and indeed $u = 3v + a$ is even as both $3v$ and a are odd. On the other hand if $a \in \{0, 2\}$ then $\delta^*(q, u) = r$ and $u = 3v + a$ is odd as $3v$ is odd and a even.

Exercise 6 **4P**

A DFA is *synchronizing* if there is a word w and a state q such that after reading w from *any* state we are always in state q.

(a) Give such a word w showing that the following DFA is synchronizing.



(b) Give an algorithm to decide if a given DFA is synchronizing.

(c) Give a *polynomial time* algorithm to decide if a given DFA is synchronizing.

Solution: *ba*

Let A be a DFA with states $q_1, \dots q_n$. For every two states q, r let $A_{q,r}$ be the DFA with the same states and transitions as A, but with q as initial state and r as unique final state. The following algorithm checks whether A is synchronizing:

```
for every state $s$
  if $A_{q_1,s} \cap \ldots \cap A_{q_n,s}$ is nonempty then
    stop and answer YES
answer NO
```

However, the algorithm is exponential, because $A_{q_1,s} \cap \ldots \cap A_{q_n,s}$ may have an exponential number of states.

Polynomial algorithm.

Given two states r, s, we can compute in polynomial time a synchronizing word $w_{r,s}$ and a state q such that after reading $w_{r,s}$ from r and s we are in state q (if the word exists): it suffices to check for every state q whether $A_{r,q} \cap A_{s,q}$ is nonempty, and if so return any word that belongs to the intersection, and the state q.

The polynomial algorithm is a greedy algorithm: starting with $\emptyset$, it computes a synchronizing word for an increasingly large set of states. Assume the algorithm has found a word w for a set of states Q, and that after reading w from any state in Q we are always in state q. Let $r \notin Q$. Then, the concatenation $ww_{q,\delta^*(r,w)}$ is a synchronizing word for the set $Q \cup \{r\}$. Since every $w_{q,r}$ can be computed ion polynomial time, and we only have to compute n of them, the algorithm is polynomial.

Alternative solution

A DFA $(Q, \Sigma, \delta, q_0, F)$ is synchronizing if and only if the following holds for the automaton $(Q, \Sigma, \delta^{-1}, q_0, F)$ where the direction of the transitions is reversed (note it is an NFA): "there is a word w and a state q such that we can reach any state from q under w". Intuitively, we revert the synchronizing word. The algorithm thus determinizes the reversed NFA for every possible initial state and checks if a macrostate containing all states is reachable. Nevertheless, the subset construction takes exponential time.

A polynomial algorithm proceeds as follows. We can decide in polynomial time whether two states q, r can synchronize: take the intersection of $(Q, \Sigma, \delta, q, Q)$ and $(Q, \Sigma, \delta, r, Q)$ and check for emptiness. If a word is accepted then put one token in each state and move all tokens along this word. The two tokens from q and r now merge and we repeat the whole procedure for some other two states that still have a token. In every step, the number of tokens decreases and the synchronizing word corresponds to the concatenation of the words. The automaton is thus synchronizing if we end up with one token only. If some intersection is empty then the automaton is not synchronizing, because in particular there are the two states that cannot synchronize.