Automata Theory and Computability

Assignment 1

(Due on Mon 8th Sep 2019)

1. Give a DFA that accepts the language of all strings over the alphabet $\{a, b\}$ such that between every pair of consecutive a's the number of b's is odd.

2. Consider the languages L and M below over the alphabet $\{a, b\}$. One of the languages is regular while the other is not. Which is which? Justify your answer by giving an automaton for one and a proof of non-regularity for the other.

   - L is the language of all strings in which the difference between the number of a's and b's in the string is at most 2.
   - M is the language of all strings which satisfy the property that in *every* prefix the difference between the number of a's and b's is at most 2. Thus, *aabab* is in the language, while *abaaab* is not.

3. Denote by $rev(x)$ the reverse of a string x. Thus $rev(abb) = bba$. Extend this definition to languages by setting $rev(L) = \{rev(x) \mid x \in L\}$.

   (a) Give a formal definition of rev.

   (b) Prove that $rev(L)$ is regular whenever L is regular. Give (a) a formal construction and (b) the main claim (without proof) that ensures the correctness of your construction.

4. Consider the language L over the alphabet $\{a, b, c\}$ defined by the following MSO sentence:

$$\begin{aligned} \exists x(\quad & Q_b(x) \wedge \\ & \forall y(y < x \implies Q_a(y)) \wedge \\ & \forall y(y > x \implies Q_c(y)) \wedge \\ & \forall y(Q_a(y) \implies \exists z(y < z \wedge Q_c(z)))). \end{aligned}$$

   Give a regular expression describing the language L.

5. Give a Monadic Second Order (MSO) logic sentence over the alphabet $\{a, b\}$ which defines the language of Question 1.

6. Show the steps in the inductive construction described in class to construct an automaton for the Presburger logic formula

$$\forall x \exists y(x = 2y \vee x = 2y + 1).$$

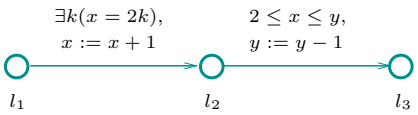
   Draw the formula tree and show the automata associated with each node in the tree.

   Examine the final automaton and tell if the given formula is valid or not.

7. Construct an automaton that accepts all the satisfying assignments of the Presburger logic formula $\exists y(x + y > 5 \wedge y \leq 2)$, using the inductive procedure described in class. Show the automaton after each inductive step.

8. A (straight-line) Presburger program is a sequence of `if`-statements, and uses two variables x and y. The guard of each `if`-statement is a Presburger logic formula with free variables in $\{x, y\}$, and the body is an assignment statement of the form $x := e$ where e is a term (over the variables x and y) in Presburger logic. More precisely, such a program can be modelled as sequence of control locations $l_1, \ldots, l_{n+1}$ $(n \geq 1)$, and transitions t_i from l_i to l_{i+1} being labelled with a Presburger guard $g(t_i)$ and an update statement $u(t_i)$. The program executes in the expected manner, beginning in the initial location l_1, in an initial state s where x and y take arbitrary values in $\mathbb{N}$, checking whether the state satisfies the guard of transition t_1, and if so, applying the update $u(t_1)$ to s and going to location l_2. A similar step is then performed from l_2, and so on. If the guard of a transition is not satisfied by the current state, or if the update assigns a negative value to a variable, the program gets "stuck."

   For example the figure below shows a Presburger program. When started in a state $(x \mapsto 2, y \mapsto 5)$, it goes to l_2 in the state $(x \mapsto 3, y \mapsto 5)$, and finally to l_3 in the state $(x \mapsto 3, y \mapsto 4)$.



   Given a precondition *pre* on the initial states, and a post-condition *post* on the final states, we say a Presburger program P satisfies the conditions $(pre, post)$ iff every execution of P that begins in a state satisfying *pre* either never reaches l_{n+1}, or reaches there in a state satisfying *post*. For example, because of the given execution, the program above does *not* satisfy the pre/post-condition $(x < y, x > y)$.

   Give a procedure to check whether a given Presburger program P, with Presburger conditions *pre* and *post*, satisfies the pair $(pre, post)$.

9. Consider interpreting Presburger logic over finite words over an alphabet A, just like $MSO(A)$. Let us call this logic $PL(A)$.

   (a) Can it be that for every sentence φ of $PL(A)$, the language $L(\varphi)$ is regular? Say why this may not hold.

   (b) Say where an inductive proof that associates an automaton with every formula, as done for $MSO(A)$, might break down.