Automata Theory and Computability

Assignment 1

(Due on Tue 4th Sep 2018)

1. Give a DFA that accepts the language of all *even* length strings over the alphabet $\{a, b\}$ in which every b is immediately preceded by an a.

2. Show that the set of strings in $\{0, 1, 2\}^*$ which are base 3 representations of even numbers, is regular.

3. Consider an alphabet A, and define $u : (A \times A)^* \to A^* \times A^*$ inductively as follows:

   (a) $u(\varepsilon) = (\varepsilon, \varepsilon)$

   (b) $\forall x \in (A \times A)^*$, $\forall (a, b) \in A \times A$, if $u(x) = (y, z)$ then $u(x.(a, b)) = (y.a, z.b)$

   If $L \subseteq A^*$ is regular, prove that the following language is regular:

   $$\{x \in (A \times A)^* \mid u(x) \in L \times L\}.$$

4. 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.

5. Let $M = (Q, s, \delta, F)$ be a smallest DFA for the regular language $L(M) \subseteq A^*$. For each $q \in Q$, let $M_q = (Q, q, \delta, F)$. Prove that $\forall p, q \in Q$, $L(M_p) = L(M_q) \implies p = q$.

6. For a language L define

   $$\textit{first-halves}(L) = \{x \mid \exists y : |x| = |y| \text{ and } xy \in L\}$$

   Here, $|y|$ denotes the length of string y. Prove that if L is regular, then *first-halves*(L) is regular. Argue for yourself that your construction is correct, but *don't* write the proof of correctness.

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

   $$\forall x \forall y (((Q_a(x) \wedge Q_b(y)) \implies x < y) \wedge (Q_a(x) \implies \exists z Q_b(z))).$$

   Give a regular expression describing the language L.

8. Give Monadic Second Order (MSO) logic sentences over the alphabet $\{a, b\}$ which define the following languages:

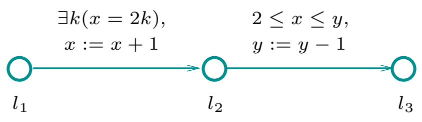
   (a) $(bab)^*$.

   (b) All strings over $\{a, b\}$ satisfying the condition that "between any two consecutive a's there are an odd number of b's."

9. Construct an automaton that accepts all the satisfying assignments of the Presburger logic formula $\exists y(x = 4y + 1)$, using the inductive procedure described in class. Show the automaton after each inductive step. What should be the output of the final automaton on the strings "00000" and "10001" respectively?

10. 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 $(pre, post)$.