Solution Sheet n°4

Solution of exercise 1:

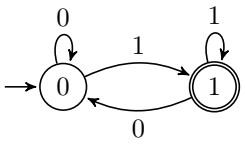
1. Any DFA can be unambiguously described as a string of the form $\langle k, n, \delta, F\rangle$ where

   - k and n are strictly positive natural numbers in decimal representing the number of states and the cardinality of the alphabet respectively (0 is the starting state by convention);
   - δ is a string of the form $\{(i,l,j),\ldots\}$, where $0 \leq i,j < k$ and $0 \leq l < n$, representing the transition function;
   - F is a string of the form $\{i,j\ldots\}$, where $0 \leq i,j < k$, representing the set of accepting states.

   For example, the string

   $$\langle 2, 2, \{(0,0,0),(0,1,1)(1,0,0),(1,1,1)\}, \{1\}\rangle$$

   represents the DFA

   We therefore define the languages

   $$\begin{aligned} \mathrm{INF}_{\mathrm{DFA}} =& \{\mathcal{A} \in A^* \mid \mathcal{A} \text{ represents a DFA whose language is infinite}\},\\ \mathrm{E}_{\mathrm{DFA}} =& \{\mathcal{A} \in A^* \mid \mathcal{A} \text{ represents a DFA whose language is empty}\}. \end{aligned}$$

2. The following procedure allows one to decide $\mathrm{E}_{\mathrm{DFA}}$:

   On input $\langle k, n, \delta, F\rangle$ coding a DFA $\mathcal{A}$:

   1: Write 0, on a second tape;
   2: Write i, in a new position on a second tape, for each state $i < k$ which is reachable in one step from state 0, avoiding repetitions. Repeat the procedure, writing down new states which are reachable in one step, for all states which are written on the second tape, until no new states are reachable.
   3: Check if some accepting state is written on the second tape. If yes, reject, if no accept.

3. The following procedure can be embodied in a Turing machine which decides $\mathrm{INF}_{\mathrm{DFA}}$:

   On input $\langle k, n, \delta_{\mathcal{A}}, F_{\mathcal{A}}\rangle$ coding a DFA $\mathcal{A}$:

   1: Write on a second tape the code of a DFA $\mathcal{B}$ recognising all strings of length strictly greater than k on the alphabet n, for example:

   $$\langle k+2, n, \delta_{\mathcal{B}}, \{k+1\}\rangle$$

   where

   $$\begin{aligned} \delta_{\mathcal{B}} =& \{(l,m,l+1) \mid \forall m < n,\ \forall l < k+2\} \cup\\ & \{(k+1,m,k+1) \mid \forall m < n\} \end{aligned}$$

   2: Write on a third tape the code of a DFA $\mathcal{C}$ recognising the intersection of the languages of $\mathcal{A}$ and $\mathcal{B}$. For example:

   $$\langle k(k+2), n, \delta_{\mathcal{C}}, F_{\mathcal{C}}\rangle$$

   where, letting $g : k \times (k+2) \to k(k+2)$ be the bijection $g(l_1,l_2) = l_2k + l_1$,

   $$\delta_{\mathcal{C}} = \left\{ \Big(g(l_1,l_2), m, g(l_1',l_2')\Big) \;\middle|\; (l_1,m,l_1') \in \delta_{\mathcal{A}},\ (l_2,m,l_2') \in \delta_{\mathcal{B}} \right\}$$

   and

   $$F_{\mathcal{C}} = \{g(l,k+1) \mid l \in F_{\mathcal{A}}\}$$

   3: Test the automata $\mathcal{C}$ for emptiness using the procedure of the previous point in this exercise. If the language of $\mathcal{C}$ is empty reject, if not accept.

Solution of exercise 2:

1. We show the equivalent assertion that for all tree T on a finite set

$$T \text{ has an infinite branch } \Leftrightarrow T \text{ is infinite.}$$

$\Rightarrow$: If T has an infinite branch, say $x : \omega \to A$, then the set $\{x_{\restriction n} \mid n \in \omega\}$ is infinite and included in T. Therefore T is infinite.

$\Leftarrow$: Suppose T is an infinite tree on a finite set A and let $\leq$ be a total ordering of the finite set A. We define an infinite branch of T by induction. Since T is infinite but A is finite, there exists a unique $\leq$-minimal $a_0 \in A$ such that $\{s \in T \mid s_{\restriction 0} = a_0\}$ is infinite. Otherwise $T \setminus \{\emptyset\} = \bigcup_{a \in A}\{s \in T \mid s_{\restriction 0} = a\}$ would be a finite union of finite sets and thus finite. Suppose now that for $n \in \omega$ the sequence $(a_0, \ldots, a_n)$ has been be defined so that $\{s \in T \mid s_{\restriction n} = (a_0, \ldots, a_n)\}$ is infinite. In particular $(a_0, \ldots, a_n) \in T$. Then as before there exists a unique $\leq$-minimal $a_{n+1} \in A$ such that $\{s \in T \mid t_{\restriction n+1} = (a_0, \ldots, a_n, a_{n+1})\}$ is infinite. We have defined by induction (and a week form of the axiom of choice, that is, DC, the *axiom of dependent choice*) a sequence $x \in A^\omega$ with $x(n) = a_n$. It is an infinite branch of T since by construction for all $n \in \omega$ we have $x_{\restriction n} = (a_0, \ldots, a_{n-1}) \in T$.

2. Let $\mathcal{M} = (Q, \Sigma, \Gamma, \Delta, q_0, q_{acc}, q_{rej})$ be a non deterministic Turing machine and view Δ as a relation, in symbols $\Delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$. For any input word $w \in \Sigma^*$ we define the *computation tree* of $\mathcal{M}$ on w as the tree $T_{\mathcal{M}}(w)$ on Δ given by

$$(s_0, \ldots, s_{n-1}) \in T_{\mathcal{M}}(w) \Leftrightarrow \begin{cases} s_i \in \Delta \text{ for } 0 \leq i \leq n-1 \text{ and} \\ \text{there exists a sequence of configurations} \\ (c_0, \ldots, c_n) \text{ such that} \\ \quad \bullet\ c_0 = q_0 w, \text{ and} \\ \quad \bullet\ c_i \text{ yields } c_{i+1} \text{ through } s_i \text{ for all } i < n. \end{cases}$$

Firstly, observe that a (deterministic) Turing decidable language is trivially nondeterministic Turing decidable.

Secondly, observe that by definition a language L is nondeterministic Turing decidable iff there exists a non deterministic Turing machine $\mathcal{M}$ such that

(i) on every input word w the computation tree of $\mathcal{M}$ on w has no infinite branch.

(ii) for every input word w, $w \in L$ iff there is at least an accepting run of $\mathcal{M}$ on w.

By the first part of this exercise, condition (i) is equivalent to saying that for all input word w the tree $T_{\mathcal{M}}(w)$ is finite. Now if L is nondeterministic Turing decidable witnessed by a machine $\mathcal{M}$ we design a (deterministic) decider for L as follows:

On input w:
1: Compute $T_{\mathcal{M}}(w)$ and write it (with an appropriate coding) on the tape.
2: Explore $T_{\mathcal{M}}(w)$ either by breadth first search or by depth first search (anyway $T_{\mathcal{M}(w)}$ is finite). If an accepting computation is found, accept.
3: When all branches of the computation tree are exhausted without having found an accepting run, reject.

Solution of exercise 3:

1. Let $\mathrm{Card}(\Sigma) = n$ and let l be the smallest natural number such that $n < 2^l$. Choose any injective function $\tilde{c} : \Sigma \to \{0,1\}^l$ and define $c : \Sigma^{<\omega} \to \{0,1\}^{<\omega}$ by $c(a_1 \cdots a_k) = \tilde{c}(a_1) \cdots \tilde{c}(a_k)$. This function is easily computed by a Turing machine on the basis of the finite data $\tilde{c}$:

On input w:
1: When reading $a \in \Sigma$ on the first tape, write $\tilde{c}(a)$ on the second tape.
2: Move right the head reading the first tape and go back to step 1.

2. Let $M = (Q, \Sigma, \Gamma, \Delta, q_0, q_{acc}, q_{rej})$ be a Turing machine. The set of states of M_c is given by $\widetilde{Q} = Q \times \{0,1\}^{\leq l}$.

On input $w \in \{0,1\}^{<\omega}$, M_c starts on state $(q_0, \emptyset)$ and then compute as follows:
1: The current state is of the form $(q, \emptyset)$ for some $q \in Q$. For $i = 1$ to l do
  • read b_i, move right in state $(q, b_1 \cdots b_i)$.
2: The current state is of the form $(q, b_1 \cdots b_l)$ and
  (i) $b_1 \cdots b_l$ is different from $\tilde{c}(a)$ for all $a \in \Sigma$ then reject, since the input is not of the form $c(w)$ for some $w \in \Sigma^*$. (Not strictly necessary)
  (ii) $b_1 \cdots b_l = \tilde{c}(a)$ for some $a \in \Sigma$ and $\delta(q, a) = (q', b, \epsilon)$. If q' is either an accepting or rejecting state, accept or reject accordingly. Otherwise
    (a) write $\tilde{c}(b)$ over $b_1 \cdots b_l$;
    (b) move to the beginning of the sequence of l symbols to the left or to the right according to $\epsilon \in \{L, R\}$.