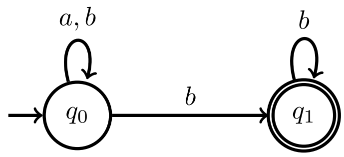
Automata and Formal Languages — Exercise Sheet 12

Exercise 12.1
Recall, from 11.4 (a) and (b), the following Büchi automaton B over $\Sigma = \{a, b\}$



as well as the ranking R of dag(w) defined by

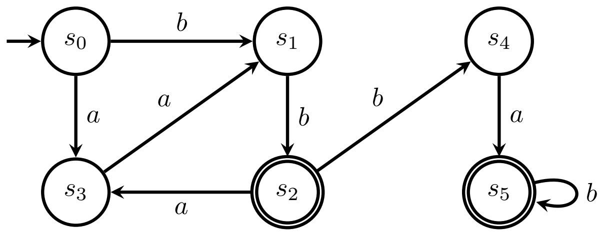
$$R(q,i) = \begin{cases} 1 & \text{if } q = q_0 \text{ and } \langle q_0, i\rangle \text{ appears in dag}(w), \\ 0 & \text{if } q = q_1 \text{ and } \langle q_1, i\rangle \text{ appears in dag}(w), \\ \bot & \text{otherwise.} \end{cases}$$

(a) Let w a ω-word. Show that R is an odd ranking for dag(w) if and only if $w \notin L_\omega(B)$.

(b) Construct a Büchi automaton accepting $\overline{L_\omega(B)}$ using the construction seen in class. *Hint*: by (a), it is sufficient to use $\{0, 1\}$ as ranks.

Exercise 12.2
Show that for every DBA A with n states there is an NBA B with $2n$ states such that $B = \overline{A}$. Explain why your construction does not work for NBAs.

Exercise 12.3
Let B be the following Büchi automaton.



(a) For every state of B, give the discovery time and finishing time assigned by a DFS on B starting in s_0 (i.e. the moment they first become grey and the moment they become black). Visit successors s_i of a given state in the ascending order of their indices i. For example, when visiting the successors of s_2, first visit s_3 and later s_4.

(b) The language of B is not empty. Give the witness lasso found by applying *NestedDFS* to B following the same convention for the order of successors as above.

(c) Given a non-empty NBA, we use the following definition of optimal execution of NestedDFS: the algorithm reports NONEMPTY at the earliest time such that all the states of a witness lasso have been explored. Is the execution in (b) optimal? Does there exists an optimal execution of $NestedDFS$ on B with a different order for visiting successors?

Solution 12.1

(a) $\Rightarrow$) (By contraposition) Let $w \in L_\omega(B)$. We have $w = ub^\omega$ for some $u \in \{a,b\}^*$. This implies that

$$\langle q_0, 0\rangle \xrightarrow{u} \langle q_0, |u|\rangle \xrightarrow{b} \langle q_1, |u|+1\rangle \xrightarrow{b} \langle q_1, |u|+2\rangle \xrightarrow{b} \cdots$$

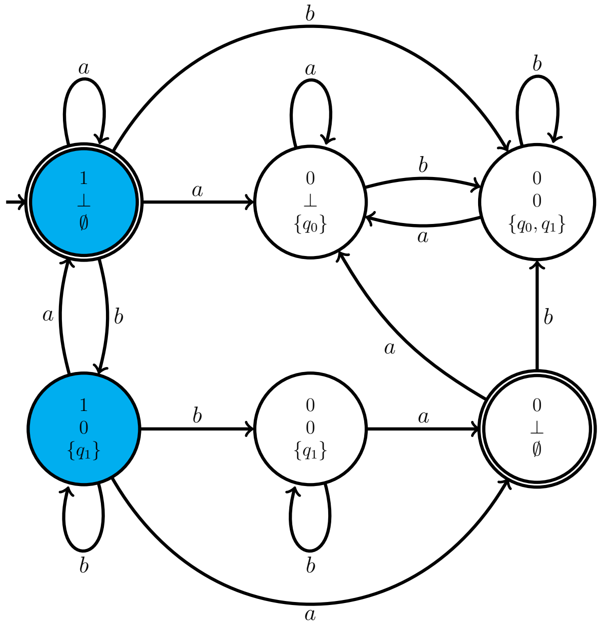
is an infinite path of dag(w). Since this path does not visit odd nodes infinitely often, R is not odd for dag(w).

$\Leftarrow$) Let $w \notin L_\omega(B)$. Suppose there exists an infinite path of dag(w) that does not visit odd nodes infinitely often. At some point, this path must only visit nodes of the form $\langle q_1, i\rangle$. Therefore, there exists $u \in \{a,b\}^*$ such that

$$\langle q_0, 0\rangle \xrightarrow{u} \langle q_1, |u|\rangle \xrightarrow{b} \langle q_1, |u|+1\rangle \xrightarrow{b} \langle q_1, |u|+2\rangle \xrightarrow{b} \cdots$$

This implies that $w = ub^\omega \in L_\omega(B)$ which is contradiction.

(b) Recall: we construct an NBA whose runs on an ω-word w are all the valid rankings of $dag(w)$. The automaton accepts a ranking R iff every infinite path of R visits nodes of odd rank i.o. By (a), for every $w \in \{a,b\}^\omega$, if dag(w) has an odd ranking, then it has one ranging over 0 and 1. Therefore, it suffices to execute *CompNBA* with rankings ranging over 0 and 1. We obtain the following Büchi automaton, for which some intuition is given below:



Any ranking r of $dag(w)$ can be decomposed into a sequence $lr_1, lr_2, \dots$ such that $lr_i(q) = r(< q, i >)$, the level i of rank r. Recall that in this automaton, the transitions $\begin{bmatrix} lr(q_0) \\ lr(q_1) \end{bmatrix} \xrightarrow{a} \begin{bmatrix} lr'(q_0) \\ lr'(q_1) \end{bmatrix}$ represent the possible next level for ranks r such that $lr(q) = r(< q, i >)$ and $lr'(q) = r(< q, i+1 >)$ for $q = q_0, q_1$.

The additional set of states in the automaton represents the set of states that "owe" a visit to a state of odd rank. Formally, the transitions are the triples $[lr, O] \xrightarrow{a} [lr', O']$ such that $lr \xrightarrow{a} lr'$ and $O' = \{q' \in \delta(O,a) | lr'(q') \textit{ is even}\}$ if $O \neq \emptyset$, and $O' = \{q' \in Q | lr'(q') \textit{ is even}\}$ if $O = \emptyset$.

Finally the accepting states of the automaton are those with no "owing" states, which represent the *breakpoints* i.e. a moment where we are sure that all runs on w have seen an odd rank since the last breakpoint.

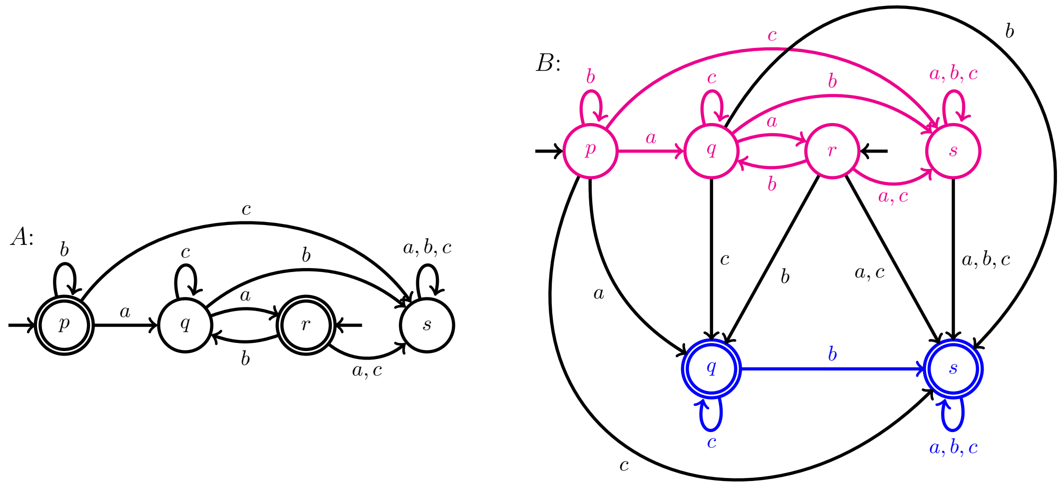
★ It is enough to only consider the blue states, as any other state cannot reach a level in which there is an odd rank; descendants of *dag* states with rank 0 can never be assigned an odd rank.

Solution 12.2

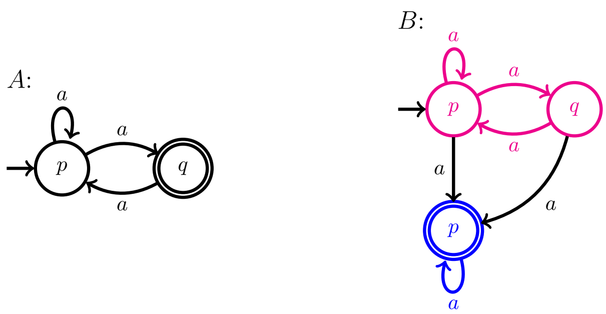
Observe that A rejects a word w iff its *single* run on w stops visiting accepting states at some point. Hence, we construct an NBA B that reads a prefix as in A and non deterministically decides to stop visiting accepting states by moving to a copy of A without its accepting states.

More precisely, we assume that each letter can be read from each state of A, i.e. that A is complete. If this is not the case, it suffices to add a rejecting sink state to A. The NBA B consists of two copies of A. The first copy is exactly as A. The second copy is as A but restricted to its non accepting states. We add transitions from the first copy to the second one as follows. For each transition (p, a, q) of A, we add a transition that reads letter a from state p of the first copy to state q of the second copy. All states of the first copy are made non accepting and all states of the second copy are made accepting. Note that B contains at most $2n$ states as desired.

Here is an example of the construction:



This construction does not work on NBAs. Indeed, we have $A = B = \{a^\omega\}$ below:



Solution 12.3

a. We note "state[discovery time/finishing time]".
$s_0[1/12], s_1[2/11], s_2[3/10], s_3[4/5], s_4[6/9], s_5[7/8]$.

b. The lasso found by *NestedDFS* from s_0 is $s_0s_1s_2s_4s5s5$.

c. Given a non-empty NBA, we use the following definition of optimal execution of NestedDFS: the algorithm reports NONEMPTY at the earliest time such that all the states of a witness lasso have been explored.

The execution given in (b) is non optimal since it does not return the lasso $s_0s_1s_2s_3s_1$ which already appeared in the explored subgraph.

There is no execution of *NestedDFS* which blackens s_2 before s_5. But there is an execution of *NestedDFS* on B which returns the lasso $s_0s_1s_2s_3s_4s5s5$ before it has visited the only other witness lasso $s_0s_1s_2s_3s_1$ and thus is optimal: the execution which does dfs1 via $s_0s_1s_2s_4s_5$, blackens s_5 then launches dfs2 from s_5

and finds a cycle. Node s_3 is not part of the explored subgraph so the algorithm reports NONEMPTY at the earliest time such that all the states of a witness lasso have been explored.