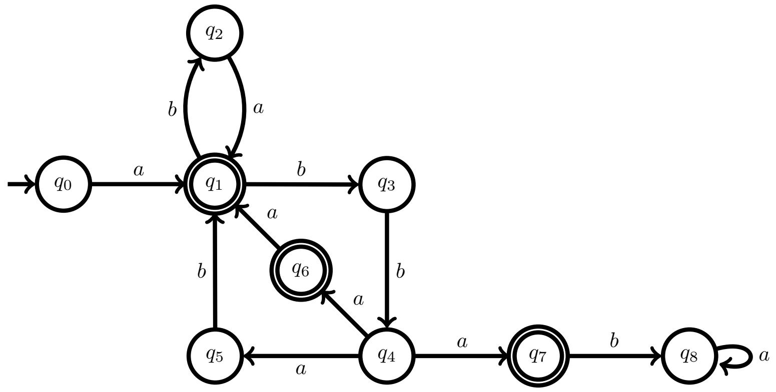
Automata and Formal Languages — Exercise Sheet 12

Exercise 12.1
Let B be the following Büchi automaton:



(a) Execute the emptiness algorithm *NestedDFS* on B.

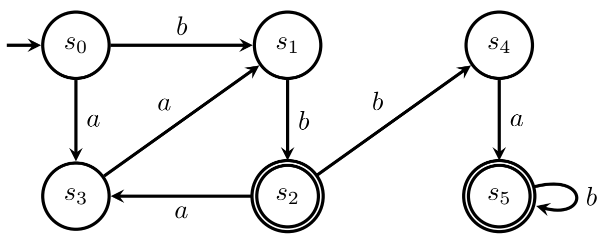
(b) Recall that *NestedDFS* is a non-deterministic algorithm and different choices of runs may return different lassos. Which lassos of B can be found by *NestedDFS*?

(c) Show that *NestedDFS* is non optimal by exhibiting some search sequence on B.

(d) Execute the emptiness algorithm *SCCsearch* on B.

(e) Which lassos of B can be found by *SCCsearch*?

Exercise 12.2
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?

Exercise 12.3

A Büchi automaton is weak if none of its strongly connected components contains both accepting and non-accepting states. Give an emptiness algorithm for weak Büchi automata. What is the complexity of the algorithm?

Solution 12.1

(a) Let us assume that the algorithms always pick states in ascending order with respect to their indices. *dfs1* visits $q_0, q_1, q_2, q_3, q_4, q_5, q_6$, then calls *dfs2* which visits $q6, q1, q2, q3, q4, q5, q6$ and reports "non empty".

(b) Since q_7 does not belong to any lasso, only lassos containing q_1 or q_6 can be found. In every run of the algorithm, *dfs1* blackens q_6 before q_1. The only lasso containing q_6 is: $q_0, q_1, q_3, q_4, q_6, q_1$. Therefore, this is the only lasso that can be found by the algorithm.

(c) The execution given in (a) shows that *NestedDFS* is non optimal since it returns the lasso $q_0, q_1, q_3, q_4, q_6, q_1$ even though the lasso q_0, q_1, q_2, q_1 was already appearing in the explored subgraph.

(d) Let us assume that the algorithm always pick states in ascending order with respect to their indices. The algorithm reports "non empty" after the following execution:

$$\xrightarrow{N.\text{push}(q_0,\{q_0\})} \begin{array}{c} N \\ \hline (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_1,\{q_1\})} \begin{array}{c} N \\ \hline (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_2,\{q_2\})} \begin{array}{c} N \\ \hline (q_2,\{q_2\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{pop}()} \begin{array}{c} N \\ \hline (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array}$$

$$\xrightarrow{N.\text{pop}()} \begin{array}{c} N \\ \hline (q_0,\{q_0\}) \end{array}$$

(e) All of them. The lasso q_0, q_1, q_2, q_1 is found by the above execution. The lasso $q_0, q_1, q_3, q_4, q_6, q_1$ is found by the following execution:

$$\xrightarrow{N.\text{push}(q_0,\{q_0\})} \begin{array}{c} N \\ \hline (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_1,\{q_1\})} \begin{array}{c} N \\ \hline (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_3,\{q_3\})} \begin{array}{c} N \\ \hline (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_4,\{q_4\})} \begin{array}{c} N \\ \hline (q_4,\{q_4\}) \\ (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array}$$

$$\xrightarrow{N.\text{push}(q_6,\{q_6\})} \begin{array}{c} N \\ \hline (q_6,\{q_6\}) \\ (q_4,\{q_4\}) \\ (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{pop}()} \begin{array}{c} N \\ \hline (q_4,\{q_4\}) \\ (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array}$$

The lasso $q_0, q_1, q_3, q_4, q_5, q_1$ is found by the following execution:

$$\xrightarrow{N.\text{push}(q_0,\{q_0\})} \begin{array}{c} N \\ \hline (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_1,\{q_1\})} \begin{array}{c} N \\ \hline (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_3,\{q_3\})} \begin{array}{c} N \\ \hline (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{push}(q_4,\{q_4\})} \begin{array}{c} N \\ \hline (q_4,\{q_4\}) \\ (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array}$$

$$\xrightarrow{N.\text{push}(q_5,\{q_5\})} \begin{array}{c} N \\ \hline (q_5,\{q_5\}) \\ (q_4,\{q_4\}) \\ (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{pop}()} \begin{array}{c} N \\ \hline (q_4,\{q_4\}) \\ (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{pop}()} \begin{array}{c} N \\ \hline (q_3,\{q_3\}) \\ (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{pop}()} \begin{array}{c} N \\ \hline (q_1,\{q_1\}) \\ (q_0,\{q_0\}) \end{array} \xrightarrow{N.\text{pop}()} \begin{array}{c} N \\ \hline (q_0,\{q_0\}) \end{array}$$

Solution 12.2

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.

Solution 12.3

The idea is to maintain a set V of the gray vertices: when a dfs meets a gray state r, by the gray-path theorem this means that there is a cycle with r in it, and since we are considering weak Büchi automata it suffices to check if r is gray. The following algorithm works in linear time:

```
Input: Weak Büchi automaton B = (Q, Σ, δ, q0, F).
Output: Lω(B) = ∅?
S, V ← ∅
dfs(q0)
report "empty"

dfs(q):
    S.add(q)
    V.add(q)
    for r ∈ succ(q) do
        if r ∉ S then
            dfs(r)
        else if r ∈ V and r ∈ F then
            report "non empty"
    V.remove(q)
```

The space complexity is $O(|V|)$, as we maintain two sets S, V that can both contain at most all the nodes of the graph. The time complexity is $O(|V| + |E|)$, same as DFS.