TD 6: Emptiness Test for Büchi Automata, Partial-Order Reduction

Exercise 1 (SCC-based Büchi Emptiness Test)**.** Consider an execution of Algorithm 1 on some Büchi automaton $\mathcal{B} = (\Sigma, S, s_0, \delta, F)$.

At each point during the DFS, we define the *search path* as the sequence of visited states for which the DFS call has not yet terminated (in the order in which they are visited), and the *explored graph* of $\mathcal{B}$ as the subgraph containing all visited states and explored transitions. We call an SCC of the *explored* graph *active* if the search path contains at least one of its states. A state is *active* if it is part of an active SCC in the explored graph (it is not necessary for the state itself to be on the search path). The *active graph* is the subgraph of the explored graph induced by the active states.

For all strongly connected component $C \subseteq S$ of $\mathcal{B}$, we call *root of* C the state of C that is visited first during the DFS, i.e. the node r_C such that $r_C.num = \min\{s.num \mid s \in C\}$ at the end of the DFS. We define similarly the root of an SCC in the explored graph.

Algorithm 1 Depth-first-search

$nr = 0$;
$visited = \{\ \}$;
dfs(s_0);
exit;

dfs(s) :

add s to *visited*;
$nr = nr + 1$;
$s.num = nr$;
for all $t \in \text{succ}(s)$ **do**
    if t not in *visited* **then**
        dfs(t)
    end if
end for

1. Show that an SCC of the explored graph that is inactive is also an SCC of $\mathcal{B}$.

2. Show that the roots of the SCCs in the active graph are a subsequence $r_1 \dots r_m$ of the search path, and that an activated node s is in the active SCC of r_i if and only if $i < m$ and $r_i.num \leq s.num < r_{i+1}.num$, or $i = m$ and $r_i.num \leq s.num$.

3. Figure out what Algorithm 2 roughly does.

4. Show that Algorithm 2 maintains the following invariants:

   - the stack W contains the sequence $(r_1, C_1) \dots (r_m, C_m)$ where $r_1 \dots r_m$ is the sequence of roots of the active graph, and C_i is the active SCC of r_i,
   - for all nodes s, $s.active$ is *true* if and only if s is active.

5. Show that Algorithm 2 returns *true* iff the language of the input Büchi automaton is empty, and that it terminates as soon as the explored graph contains a counterexample.

6. Adapt Algorithm 2 to test emptiness of a generalized Büchi automaton with acceptance sets $F_1, \dots, F_n$.

7. Compare with the nested DFS algorithm from the lectures.

Algorithm 2 Emptiness Test

```
nr = 0;
visited = { };
initialize empty stack W
dfs(s_0);
return true;
```

dfs(s):

```
add s to visited;
s.active = true;
nr = nr + 1;
s.num = nr;
push (s, {s}) onto W;
for all t ∈ succ(s) do
   if t not in visited then
      dfs(t)
   else if t.active then
      D = { };
      repeat
         pop (u, C) from W;
         if u is accepting then
            return false
         end if
         merge C into D;
      until u.num ≤ t.num;
      push (u, D) onto W;      // u here is the last value taken by u in the repeat-until block
   end if
end for
if s is the top root in W then
   pop (s, C) from W;
   for all t in C do
      t.active = false
   end for
end if
```

Exercise 2. Fix a set of atomic propositions AP, and $\Sigma = 2^{\mathrm{AP}}$. Recall that $\sigma, \rho \in \Sigma^\omega$ are *stuttering equivalent*, written $\sigma \sim \rho$, when there exist infinite integer sequences $0 = i_0 < i_1 < \cdots$ and $0 = k_0 < k_1 < \cdots$ such that for all $\ell \geq 0$,

$$\sigma(i_\ell) = \sigma(i_\ell + 1) = \cdots = \sigma(i_{\ell+1} - 1) = \rho(k_\ell) = \rho(k_\ell + 1) = \cdots = \rho(k_{\ell+1} - 1)\,,$$

where $\sigma(i) \in \Sigma$ denotes the letter at position i in σ.

A language $L \subseteq \Sigma^\omega$ is *stutter-invariant* if for all stuttering equivalent words $\sigma, \rho \in \Sigma^\omega$, we have $\sigma \in L$ if and only if $\rho \in L$.

1. Show that if φ is an $\mathrm{LTL}(\mathrm{AP}, \mathsf{U})$ formula, then $L(\varphi) = \{\sigma \in \Sigma^\omega \mid \sigma, 0 \models \varphi\}$ is stutter-invariant.

A word $\sigma \in \Sigma^\omega$ is *stutter-free* if, for all $i \in \mathbb{N}$, either $\sigma(i) \neq \sigma(i+1)$, or $\sigma(i) = \sigma(j)$ for all $j \geq i$.

2. Show that for all $\sigma \in \Sigma^\omega$, there exists a unique $\sigma' \in \Sigma^\omega$ such that σ' is stutter-free and $\sigma \sim \sigma'$.

3. Given $a \in \Sigma$, we write a for the formula $\bigwedge_{p \in a} p \wedge \bigwedge_{p \notin a} \neg p$. That is, $\sigma, i \models a$ if and only if $\sigma(i) = a$.

   (a) Give a formula $\psi_{a,a}$ in $\mathrm{LTL}(\mathrm{AP}, \mathsf{U})$ such that for all *stutter-free* words $\sigma \in \Sigma^\omega$, we have $\sigma, 0 \models \psi_{a,a}$ if and only if $\sigma, 0 \models a \wedge \mathsf{X}\, a$.

   (b) Let $a, b \in \Sigma$ with $a \neq b$. Give a formula $\psi_{a,b}$ in $\mathrm{LTL}(\mathrm{AP}, \mathsf{U})$ such that for all *stutter-free* words $\sigma \in \Sigma^\omega$, we have $\sigma, 0 \models \psi_{a,b}$ if and only if $\sigma, 0 \models a \wedge \mathsf{X}\, b$.

4. Let φ be any $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$ formula. Construct by induction on φ an $\mathrm{LTL}(\mathrm{AP}, \mathsf{U})$ formula $\tau(\varphi)$ such that for all *stutter-free* words $\sigma \in \Sigma^\omega$, we have $\sigma, 0 \models \varphi$ iff $\sigma, 0 \models \tau(\varphi)$.

5. Let φ be an $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$ formula such that $L(\varphi)$ is stutter-invariant. Show that $L(\varphi) = L(\tau(\varphi))$.