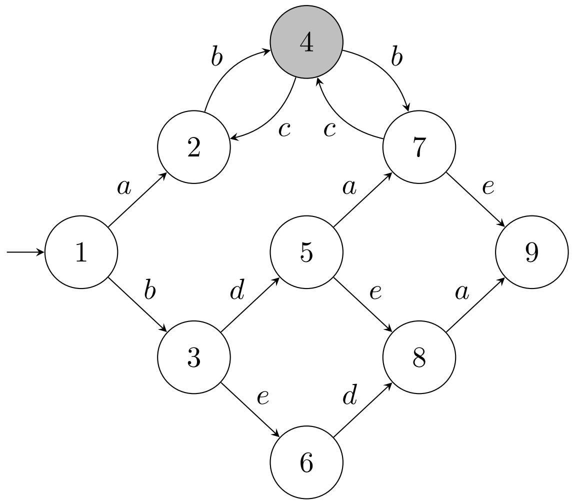
TD 7: Emptiness Test for Büchi Automata, Partial-Order Reduction

Exercise 1. Consider the labeled Kripke structure $\mathcal{K}$ shown below with actions $\{a, b, c, d, e\}$ and one atomic proposition q, where q holds only on state 4.



1. Determine a maximal independence relation I.

   (Recall that $I \subseteq A \times A$ is an independance relation for $\mathcal{K}$ if it is irreflexive, symmetric, and for all $(a, b) \in I$ and $s \in S$, if $a, b \in en(s)$, $s \xrightarrow{a} t$, and $s \xrightarrow{b} u$, then there exists $v \in S$ such that $a \in en(u)$, $b \in en(t)$, $t \xrightarrow{b} v$, and $u \xrightarrow{a} v$.)

2. Determine the maximal invisibility set U.

   (Recall that U is an invisibility set if for all $a \in U$ and $(s, a, s') \in \rightarrow$, $\nu(s) = \nu(s')$.)

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

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

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

1. Prove that if φ is an LTL(AP, U) formula, then $L(\varphi)$ is stutter-invariant.

2. A word $\sigma = a_0 a_1 \cdots$ in Σ^ω is *stutter-free* if, for all i in $\mathbb{N}$, either $a_i \neq a_{i+1}$, or $a_i = a_j$ for all $j \geq i$. We note $\mathrm{sf}(L)$ for the set of stutter-free words in a language L.

   Show that, if L and L' are two stutter-invariant languages, then $\mathrm{sf}(L) = \mathrm{sf}(L')$ iff $L = L'$.

3. Let φ be an $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$ formula such that $L(\varphi)$ is stutter-invariant. Construct inductively a formula $\tau(\varphi)$ of $LTL(\mathrm{AP}, \mathsf{U})$ such that $\mathrm{sf}(L(\varphi)) = \mathrm{sf}(L(\tau(\varphi)))$, and thus such that $L(\varphi) = L(\tau(\varphi))$ according to the previous question.

Exercise 3 (Büchi Emptiness Test)**.** Consider an execution of Algorithm 1 on some Büchi automaton $\mathcal{B} = (\Sigma, S, s_0, \delta, F)$.

Algorithm 1 Depth-first-search

```
nr = 0;
hash = { };
dfs(s0);
exit;

dfs(s) :
add s to hash;
nr = nr + 1;
s.num = nr;
for all t ∈ succ(s) do
   if t not in hash then
      dfs(t)
   end if
end for
```

1. At each point during the DFS, we define the *search path* as the sequence $s_0 s_1 \ldots s_n$ of visited states for which the DFS call has not yet terminated (in the order in which they are visited).

   Show that $s_i.num < s_j.num$ iff $i < j$, and that for all $i < j$, $s_i \rightarrow^+ s_j$.

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. Note that it is also the last state in C from which the DFS backtracks, and, at that point, all states and edges in the component C have been considered.

The *explored graph* of $\mathcal{B}$ denotes 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.

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

4. Show that the roots of the SCCs in the active graph are a subsequence of the search path.

5. Let s be an active state and t the root of its SCC in the active graph. Show that there is no active root u with $t.num < u.num < s.num$.

6. Show that if s, t are two active states with $s.num \leq t.num$, then $s \to^* t$.

7. Let C, C' be two active SCCs and $t \in C, s \in C'$ such that $t.num \leq s.num$. Show that if an edge (s, t) is added to the explored graph, after the addition, C and C' are in the same SCC of the explored graph.

8. We modify Algorithm 1 to maintain a stack W with elements of the form (s, C), where s is the root of an active SCC, and C is the set of states in the SCC of s. Show that Algorithm 2 returns *true* iff the language of the input Büchi automaton is empty.

9. Show that if $L(\mathcal{B}) \neq \emptyset$, Algorithm 2 terminates as soon as the explored graph contains a counterexample.

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

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

Algorithm 2 Emptiness Test

```
nr = 0;
hash = { };
W = { };
dfs(s_0);
return true;
```

dfs(s):

```
add s to hash;
s.active = true;
nr = nr + 1;
s.num = nr;
push (s, {s}) onto W;
for all t ∈ succ(s) do
  if t not in hash 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;
  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
```