Exercise 1. Consider the labeled Kripke structure $K$ 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 independence relation for $K$ 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 $\Sigma = 2^{AP}$. 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 $\varphi$ is an LTL(AP, U) formula, then $L(\varphi)$ is stutter-invariant.
2. A word \( \sigma = a_0 a_1 \cdots \) in \( \Sigma^\omega \) 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 \( \operatorname{sf}(L) \) for the set of stutter-free words in a language \( L \).

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

3. Let \( \varphi \) be an \( \text{LTL}(\text{AP}, X, U) \) formula such that \( L(\varphi) \) is stutter-invariant. Construct inductively a formula \( \tau(\varphi) \) of \( \text{LTL}(\text{AP}, U) \) such that \( \operatorname{sf}(L(\varphi)) = \operatorname{sf}(L(\tau(\varphi))) \), 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 \( B = (\Sigma, S, s_0, \delta, F) \).

**Algorithm 1** Depth-first-search

1. \( nr = 0; \)
2. \( \text{hash} = \{ \}; \)
3. \( \text{dfs}(s_0); \)
4. \( \text{exit}; \)

\( \text{dfs}(s) : \)

1. add \( s \) to \( \text{hash} \);
2. \( nr = nr + 1; \)
3. \( s.num = nr; \)
4. for all \( t \in \text{succ}(s) \) do
5. \( \text{if} \ t \text{ not in } \text{hash} \text{ then} \)
6. \( \text{dfs}(t) \)
7. \( \text{end if} \)
8. \( \text{end for} \)

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 \( B \), 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 \( B \) 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 $B$.

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 $s \rightarrow^* t$.

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 $L(B) \neq \emptyset$, 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

1. \( nr = 0; \)
2. \( hash = \{ \}; \)
3. \( W = \{ \}; \)
4. \( dfs(s_0); \)
5. return true;

dfs(s):
1. add \( s \) to hash;
2. \( s.active = true; \)
3. \( nr = nr + 1; \)
4. \( s.num = nr; \)
5. push \( (s, \{s\}) \) onto \( W; \)
6. for all \( t \in \text{succ}(s) \) do
7. if \( t \) not in \( hash \) then
8. dfs(t)
9. else if \( t.active \) then
10. \( D = \{ \}; \)
11. repeat
12. pop \( (u, C) \) from \( W; \)
13. if \( u \) is accepting then
14. return false
15. end if
16. until \( u.num \leq t.num; \)
17. push \( (u, D) \) onto \( W; \)
18. end if
19. end if
20. end for
21. if \( s \) is the top root in \( W \) then
22. pop \( (s, C) \) from \( W; \)
23. for all \( t \) in \( C \) do
24. \( t.active = false \)
25. end for
26. end if