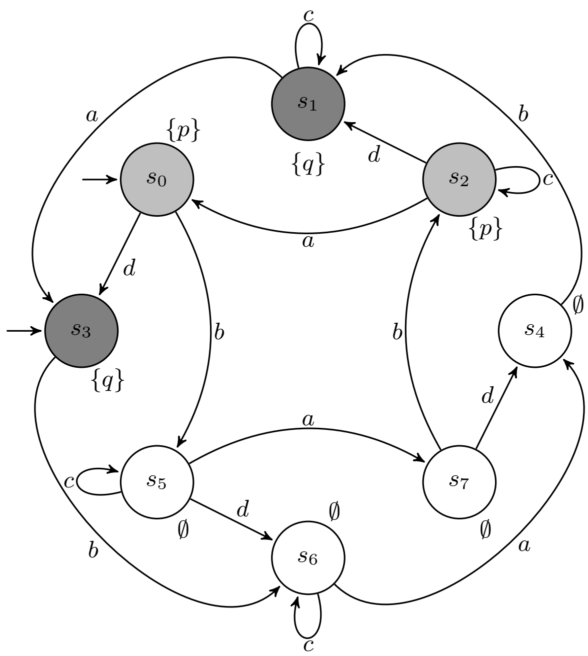
TD 8: Partial Order Reductions

1 Ample Sets

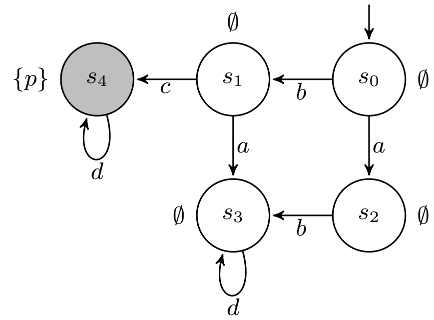
Exercise 1 (Ample Sets)**.** Consider the following transition system with state set $S = \{s_0, \ldots, s_7\}$ and transition alphabet $\Delta = \{a, b, c, d\}$:



1. Compute the independence set $I \subseteq \Delta^2$.
2. What is the set of invisible actions $U \subseteq \Delta$?
3. Propose an assignment $\mathit{red} : S \to 2^\Delta$ of ample sets satisfying conditions C_0–C_3 of the lecture notes.
4. Propose a stutter-equivalent system with a reduced set of states.

Exercise 2 (Alternate conditions)**.**

1. Consider the alternate condition C_1'': for any s with $\mathit{red}(s) \neq \mathit{en}(s)$, any a in $\mathit{red}(s)$ is independent from every b in $\mathit{en}(s)\backslash\mathit{red}(s)$. Show that C_1 implies C_1''. Does the converse implication hold? *Hint: consider the following system with red:* $s_0 \mapsto \{a\}$, $s_2 \mapsto \{b\}$, *and* $s_3 \mapsto \{d\}$.



2. Consider the alternate condition C_3'': any cycle in $\mathcal{K}'$ contains at least one state s with $red(s) = en(s)$. Show that C_0–C_2 and C_3'' together imply C_3. Do C_0–C_3 together imply C_3''?

2 Nested DFS

Partial order reduction using ample sets is especially suited for on-the-fly algorithms for the emptiness of Büchi automata. The usual, linear-time algorithm for this task uses a *nested depth-first search.*

Recall a DFS-based algorithm for cycle detection from a given state $s \in S$ in a finite directed graph (Q, T), with a global variable $V \subseteq Q$ for the set of already visited vertices:

```
found ← false                                  /* no cycle found yet */
P ← s                         /* a stack P ∈ Q* of vertices to process */
V ← V ∪ {s}                           /* the set of visited vertices */
repeat
    s' ← top(P)
    if s ∈ T(s') then
        found ← true
        push(s, P)
    else
        if T(s')\V ≠ ∅ then
            s'' ← some(T(s')\V)      /* some vertice accessible from s' */
            push(s'', P)
            V ← V ∪ {s''}
        else
            pop(P)
        end
    end
until P = ε ∨ found
return found
```

Algorithm 1: CYCLE(s)

One way to use this algorithm for Büchi automata emptiness is to first find the accepting states s in F of the automaton $\mathcal{B} = \langle Q, \Sigma, \delta, I, F \rangle$ that are reachable from I (also by an *external* DFS), and then call Cycle(s) with $V = \emptyset$ for each such state—a quadratic time algorithm. The next exercise refines this approach:

Exercise 3 (Nested DFS)**.** The idea of the nested DFS algorithm is to avoid states from previous cycle searches in latter searches—hence the global V in Cycle. Consider the following external DFS ACycle that uses a set of visited states U, and calls Cycle on reachable accepting states s' of $\mathcal{B}$ *once their reachable states have been processed* (see line 12).

1 $P' \leftarrow s$ `/* a stack` $P' \in Q^*$ `of vertices to process */`
2 $U \leftarrow U \cup \{s\}$ `/* the set of visited vertices */`
3 **repeat**
4     $s' \leftarrow top(P')$
5   **if** $T(s') \backslash U \neq \emptyset$ **then**
6         $s'' \leftarrow some(T(s') \backslash U)$ `/* some vertice accessible from` s' `*/`
7         $push(s'', P')$
8         $U \leftarrow U \cup \{s''\}$
9   **else**
10         $pop(P')$ `/* all the successors of` s' `have been processed */`
11   **if** $s' \in F$ **then**
12             $found \leftarrow$ Cycle(s') `/* call Cycle on` s' `*/`
13   **end**
14   **end**
15 **until** $P' = \varepsilon \vee found$

Algorithm 2: ACycle(s)

1. Consider a call to ACycle(s_0) with empty initial U and V. Assume there exists a call to Cycle(s) performed by ACycle such that, before the call,

   $$\text{there is a cycle } q_0 q_1 \cdots q_k,\ q_0 = s = q_k \wedge \exists i,\ q_i \in V\ ; \qquad (\dagger)$$

   without loss of generality assume that s is the first state s.t. ($\dagger$) occurs. Note that there has to be $s' \in Q$ s.t. Cycle(s') was invoked before Cycle(s) and q_i was visited and added to V during this call to Cycle(s').

   (a) Consider the two cases: s was visited (i.e. pushed on P') before or after s' in the run of ACycle, and derive a contradiction in both cases.

   (b) Why does ACycle succeeds in finding acceptance cycles from s_0?

2. Provide the missing invocation context for ACycle to solve Büchi automata emptiness.

3. Show that the algorithm works in linear time.

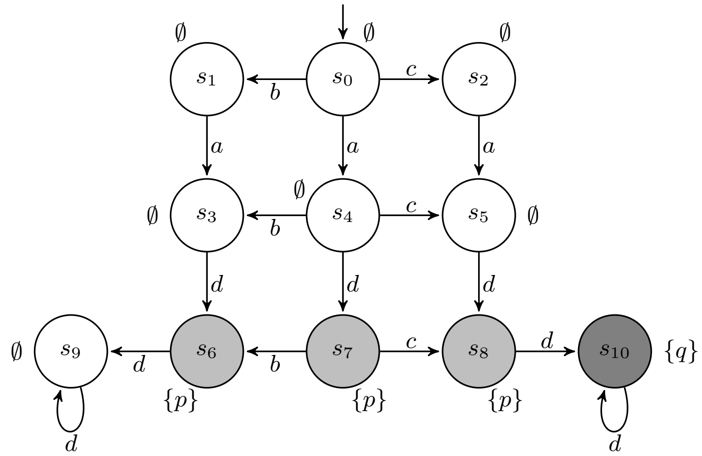
Exercise 4 (Ample Sets in Nested DFS)**.**

1. Assume you are given ample sets for each reachable state (i.e. you can call $red(s)$ for any reachable state s and obtain the ample set for s). Adapt the nested DFS algorithm to only explore the reduced system.

2. Assume now that you are only provided with a $red'(s)$ function that provides ample sets verifying C_0–C_2, but not necessarily C_3. Adapt your algorithm to enforce C_3'' on the fly. How do C_3' and C_3'' compare?

3 CTL(U) Model Checking

Exercise 5 (C_0–C_3 are not Sufficient)**.** Consider the following system with $\Delta = \{a, b, c, d\}$:



1. Let $red(s_0) = \{b, c\}$ and $red(s) = en(s)$ for $s \neq s_0$; show that this ample set assignment is compatible with C_0–C_3.

2. Exhibit a CTL(U) formula that distinguishes between the original system and its reduction.

3. Can you propose an assignment that also complies with C_4: if $red(s) \neq en(s)$, then $|red(s)| = 1$?