TD 10: Pushdown Systems

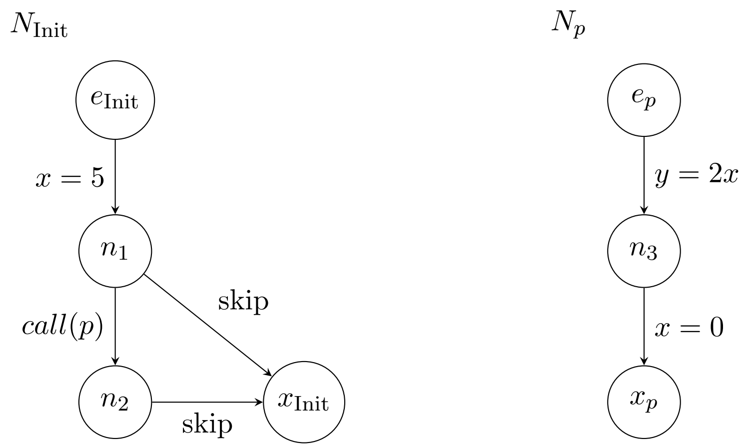
Exercise 1 (Labelled Pushdown Systems)**.** Let $\mathcal{P} = (P, \Gamma, \Delta, \Sigma)$ be a labelled pushdown system, i.e. the rules in Δ are of the form $pA \overset{a}{\hookrightarrow} qw$, where $p, q \in P$ are control locations, $A \in \Gamma$ and $w \in \Gamma^*$ are stack symbols, and additionally $a \in \Sigma$ is an *action*. The set of configurations $Con(\mathcal{P})$ consists of the tuples qw with $q \in P$ and $w \in \Gamma^*$. For two configurations c, c' we write $c \overset{w}{\Rightarrow} c'$, where $w \in \Sigma^*$, if c can be transformed into c' by a sequence of rules whose labels yield w.

Given a regular set of configurations C, it is known how to compute $pre^*(C) = \{\, c \in Con(\mathcal{P}) \mid \exists c' \in C, w \in \Sigma^* : c \overset{w}{\Rightarrow} c' \,\}$. If C is accepted by an automaton with n states, this takes $\mathcal{O}(n^2 \cdot |\Delta|)$ time.

1. Let $L \subseteq \Sigma^*$ be a regular language and C be a regular set of configurations. We define
$$pre^*[L](C) := \{\, c \in Con(\mathcal{P}) \mid \exists c' \in C, w \in L : c \overset{w}{\Rightarrow} c' \,\}.$$
One can prove that $pre^*[L](C)$ is regular. Describe how to compute a finite automaton accepting $pre^*[L](C)$.

2. Give a bound on the amount of time it takes to compute $pre^*[L](C)$.

Exercise 2 (Data-flow Analysis)**.** We consider a problem from interprocedural data-flow analysis. A program consists of a set *Proc* of procedures that can execute and recursively call one another. The behaviour of each procedure p is described by a flow graph, an example with two procedures is shown below.



Formally, a flow graph for procedure $p \in Proc$ is a tuple $G_p = (N_p, A, E_p, e_p, x_p)$, where

- N_p are the nodes, corresponding to program locations; we denote $N := \bigcup_{p \in Proc} N_p$.
- $A = A_I \cup \{ call(p) \mid p \in Proc \}$ are the actions, where A_I are *internal actions* (such as assignments etc); additionally an action can call some procedure. A is identical for all procedures.
- $E_p \subseteq N_p \times A \times N_p$ are the edges, labelled with actions from A. We denote $E := \bigcup_{p \in Proc} E_p$.
- e_p is the *entry point* of procedure p, i.e. when p is called, execution will start at e_p.
- x_p is the *exit point* of p (without any outgoing edges); when x_p is reached, p terminates and execution resumes at last call site of p.

1. Construct a labelled pushdown system with one single control location that expresses the behaviour of the procedures in *Proc*.

Suppose that the internal actions in A_I describe assignments to global variables, i.e. they are of the form $v := expr$, where v is a variable and *expr* the right-hand-side expression. If v is a variable, then $D_v \subseteq A_I$ is the set of actions that assign a value to v and $R_v \subseteq A_I$ the set of actions where v occurs on the right-hand side.

Let $Init \in Proc$ be an initial procedure and $n \in N$ a node in the flow graph. We say that variable v is *live* at n if there exists a node n' and an execution that (i) starts at e_{Init}, (ii) passes n, (iii) finally reaches n' with an action from R_v, and (iv) there is no assignment to v between n and n' in this execution. (Intuitively, this means that the value that v has at n matters for some execution; this is used in compiler construction to determine whether an optimizing compiler may "forget" the value of v at n.) For instance, in the shown example, the variable x is live at n_1 and e_p, but not in the other nodes.

2. Describe a regular language $L \subseteq A^*$ that describes the sequences of actions that can happen along such executions between n and n'.
3. Describe how, given a variable v, one can compute the set of nodes n such that v is live at n.

Exercise 3 (Basic Pushdown Processes)**.** A *Basic Pushdown Process* (BPP) is a pushdown system with one single state q. Find a pushdown system $\mathcal{P}$ such that there exists no BPP $\mathcal{Q}$ bisimilar to $\mathcal{P}$.