# Detecting Useless Transitions in Pushdown Automata

Dick Grune, Wan Fokkink, Evangelos Chatzikalymnios,
Brinio Hond, and Peter Rutgers

Vrije Universiteit Amsterdam, Department of Computer Science

**Abstract.** Pushdown automata may contain transitions that are never used in any accepting run of the automaton. We present an algorithm for detecting such useless transitions. A finite automaton that captures the possible stack content during runs of the pushdown automaton, is first constructed in a forward procedure to determine which transitions are reachable, and then employed in a backward procedure to determine which of these transitions can lead to a final state. An implementation of the algorithm is shown to exhibit a favorable performance.

## 1 Introduction

Context-free languages are used in language specification, parsing, and code optimization. They are defined by means of a context-free grammar or a pushdown automaton (PDA). Some languages can be specified more efficiently by a PDA than by a context-free grammar, as shown by Goldstine, Price, and Wotschke [6]. PDAs are at the root of deterministic parsers for context-free languages (notably LL, LR), see e.g. [10, 1]. We consider PDAs in which any number of symbols can be popped from as well as pushed onto the stack in one transition. Popping zero or multiple symbols is useful in bottom-up parsing, and facilitates the reversal of a PDA.

For context-free grammars, it is rather straightforward to determine whether a production is *useless*, i.e., cannot occur in a derivation from the start variable to a string of terminal symbols; such a method is discussed in many textbooks on formal languages (e.g., [9, Thm. 6.2]). It consists of two parts: detect which variables are reachable from the start variable, and which variables can be transformed into a string of terminal symbols. Productions that contain a useless variable, not satisfying these two properties, can be removed from the grammar without changing the associated language. A grammar generated from a program sometimes consists almost entirely of useless productions, such as in case of "parsing by intersection" [8].

This paper addresses the research question posed in [8] to develop an efficient algorithm for determining the useless transitions in a PDA, meaning that no run of the PDA from the initial configuration to a final state includes this transition. Such a transition can be removed from the PDA without changing the language accepted by the PDA, and improves the performance of running the PDA. This is especially sensible if the PDA has been generated automatically, because then there tend to be a substantial number of useless transitions. Similar to detecting useless variables in context-free grammars, our algorithm for detecting useless transitions in a PDA consists of two parts. It stays entirely in the realm of automata. The first part finds which transitions are not reachable from the initial configuration. Here we exploit an algorithm by Finkel, Willems

and Wolper [5] to construct a finite automaton (NFA) that captures exactly all possible stacks in the reachable configurations of a PDA. Their approach is modified slightly to take into account that multiple symbols may be popped from the stack at once. The second part of our algorithm, which to the best of our knowledge is novel, finds after which transitions it is impossible to reach a final state. Here we use the NFA constructed in the first part to compute in a backward fashion which transitions can lead to a final state in the PDA.

We prove that the algorithm marks exactly the useless transitions. The worst-case time complexity of the algorithm is $O(Q^4T)$, with $Q$ the number of states and $T$ the number of transitions of the PDA. This worst case actually only occurs in the unlikely case that the NFA is constructed over a large number of iterations, is saturated with $\varepsilon$-transitions, and contains a lot of backward nondeterminism. A prototype implementation of the algorithm exhibits a good performance.

An alternative approach is to use the functions $post*$ and $pre*$, to compute the reachable configurations as well as the configurations from which a final state can be reached. This alternative approach was also implemented [3], and it was shown on a large test set of randomly generated PDAs that the algorithm presented in this paper has a much better performance that this alternative approach.

*Related work* Bouajjani, Esparza and Maler [2] employed a method similar to the one in [5] to capture the reachable configurations of a PDA via an NFA, in the context of model checking infinite-state systems. Griffin [7] showed how to detect which transitions are reachable from the initial configuration in a deterministic pushdown automaton (DPDA). For each transition, the algorithm creates a temporary DPDA in which the successive state of the transition is set to a new, final state; all other states in the temporary DPDA are made non-final. Then it is checked whether the language generated by the DPDA is empty; if it is, the transition is unreachable. This algorithm determines which transitions are reachable from the initial configuration, but not which transitions can lead to a final state.

## 2 Preliminaries

A nondeterministic pushdown automaton (PDA) consists of a finite set of states $Q$, a finite input alphabet $\Sigma$, a finite stack alphabet $\Gamma$, a finite transition relation $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \to 2^{Q \times \Gamma^*}$, an initial state $q_0$, and a set $F$ of final states. Here $\varepsilon$ denotes the empty string. Note that zero or multiple symbols can be popped from the stack in one transition. It is assumed that the initial stack is empty. (An arbitrary initial stack $\sigma$ can be constructed by adding a new initial state $\hat{q_0}$ and a transition $\delta(\hat{q_0}, \varepsilon, \varepsilon) = \{(q_0, \sigma)\}$.)

A configuration consists of a state from $Q$ together with a stack from $\Gamma^*$. We let $a, b, c, d$ denote elements in $\Gamma$, and $\rho, \sigma, \tau, \upsilon, \zeta$ strings in $\Gamma^*$, where the leftmost element represents the top of the stack. The reverse of a string $\sigma$ is denoted by $\sigma^R$. A transition $(r, \tau) \in \delta(q, \alpha, \sigma)$ with $\alpha \in \Sigma \cup \{\varepsilon\}$ gives rise to moves $(q, \sigma\rho) \xrightarrow{\alpha} (r, \tau\rho)$ between configurations, for any $\rho \in \Gamma^*$. The language accepted by a PDA consists of the strings in $\Sigma^*$ that give rise to a run of the PDA from the initial configuration $(q_0, \varepsilon)$ to a configuration $(r, \sigma)$ with $r \in F$. A transition of a PDA is *useless* if no run of the PDA from the initial configuration $(q_0, \varepsilon)$ to a configuration $(r, \sigma)$ with $r \in F$, for any

input string from $\Sigma^*$, includes this transition. To determine the useless transitions, input strings from $\Sigma^*$ are irrelevant. The point is that, since a run for any input string suffices to make a transition useful, we can assume that any desired terminal symbol from $\Sigma$ is available as input at any time. Input strings from $\Sigma$ are therefore disregarded in our algorithm to detect useless transitions. (In the context of model-checking infinite-state systems, PDAs in which input strings are disregarded are called "pushdown systems" or "pushdown processes" [11].) A transition $(r, \tau) \in \delta(q, \sigma)$ is written as $q \overset{\sigma/\tau}{\rightarrow} r$. It gives rise to moves $(q, \sigma\rho) \rightarrow (r, \tau\rho)$. We write $(s, \upsilon) \rightarrow^* (t, \zeta)$ if there is a run from $(s, \upsilon)$ to $(t, \zeta)$ of the PDA, consisting of zero or more moves.

A nondeterministic finite automaton (NFA) consists of a finite set of states $Q$, a finite input alphabet $\Sigma$, a transition relation $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$, an initial state $q_0$, and a set $F$ of final states. In our application of NFAs, the input alphabet is the stack alphabet $\Gamma$ from the PDA. A transition $r \in \delta(q, a)$ is written as $q \overset{a}{\rightarrow} r$. We write $q \overset{a_1 \ldots a_k}{\rightsquigarrow} r$ if there is a path from $q$ to $r$ in the NFA with consecutive labels $a_1, ..., a_k \in \Gamma$. We write $q \overset{a_1 \ldots a_k}{\Longrightarrow} r$ if there is such a path from $q$ to $r$, possibly intertwined with transitions labeled by $\varepsilon$. The language accepted by an NFA consists of the strings $a_1 ... a_k$ in $\Sigma^*$ for which there exists a path $q_0 \overset{a_1 \ldots a_k}{\Longrightarrow} r$ with $r \in F$.

## 3 Detecting the useless transitions in a PDA

Our algorithm for detecting useless transitions in a PDA summarizes all reachable configurations of the PDA in an NFA. As a first step, an NFA is constructed that accepts the stacks that can occur during any run of the PDA. A second step determines which transitions can lead to a configuration from which a final state can be reached. Transitions that cannot be reached from the initial state (as determined in step 1) or that cannot lead to a final state (as determined in step 2) are useless.

### 3.1 Detecting the unreachable transitions

A configuration or transition of a PDA $P$ is reachable if it is employed in a run of $P$, starting from the initial configuration. The reachable configurations of $P$ are captured by means of an NFA $N$. The stacks in $\Gamma^*$ that can occur at a state $q$ in $P$ are accepted at the state $q$ in $N$, in reverse order. During the construction of $N$, intermediate non-final states are created when multiple symbols are pushed onto the stack in one transition. They are denoted by $n, m$, to distinguish them from the states $q, r, s, t$ that are inherited from $P$ and which are taken to be final in $N$. A state in $N$ that may be either final or non-final is denoted by $x, y, z$.

Fix a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$; as said, we will disregard $\Sigma$. To achieve a single final state without outgoing transitions that is only reached with an empty stack, we perform a standard transformation on $P$. A fresh stack symbol $b_0$ is added to $\Gamma$, and the initial stack is $b_0$ (instead of $\varepsilon$). In each run of the PDA, $b_0$ is always at the bottom of the stack. Fresh states $q_e$ and $q_f$ are added to $Q$, and $\delta$ is extended with transitions $q \overset{\varepsilon/\varepsilon}{\rightarrow} q_e$ for every $q \in F$, $q_e \overset{a/\varepsilon}{\rightarrow} q_e$ for every $a \in \Gamma \backslash \{b_0\}$, and $q_e \overset{b_0/\varepsilon}{\rightarrow} q_f$. We change $F$ to $\{q_f\}$. The resulting PDA is called $P_0$.

Initially the NFA $N$ under construction consists only of the transition $m_0 \overset{b_0}{\rightarrow} q_0$; the fresh state $m_0$ is non-final and $q_0$ final. Intuitively, this transition builds the initial stack

of $P_0$. The set $U_1$ of unreachable transitions in $P_0$ initially, as an overapproximation, contains all transitions in $P_0$. The NFA $N$ and the set $U_1$ are constructed as follows.

**Algorithm 1** Procedure *forward* to detect the unreachable transitions in a PDA $P_0$.

For each transition $\theta = q \xrightarrow{\sigma/\tau} r$ in $P_0$ do:

1. If $q$ is not a state in $N$, then stop this iteration step.
2. Determine the set $S_{q,\sigma}$, which either consists only of $q$, if $\sigma = \varepsilon$, or of the (non-final) states $n$ for which there exists a path $n \xrightarrow{a} y \overset{\sigma'^R}{\Longrightarrow} q$ in $N$, if $\sigma = \sigma' a$.
3. If $S_{q,\sigma} = \emptyset$, then stop this iteration step.
4. If $\theta \in U_1$, then delete $\theta$ from $U_1$, and establish a path $\overset{\tau^R}{\rightsquigarrow} r$ in $N$ (see below); the state $r$ in $N$ is final.
5. Let $y$ be the first state in the path $\overset{\tau^R}{\rightsquigarrow} r$. For each state $x \in S_{q,\sigma}$, if the transition $x \xrightarrow{\varepsilon} y$ is not yet present in $N$, then add this transition to $N$.

If $N$ changed during this run, then perform *forward* again, over all transitions in $P_0$.

Else, stop, and return the constructed NFA $N$ and the set $U_1$ of unreachable transitions in $P_0$. These transitions are then culled from $P_0$, producing the PDA $P_1$.

The sets $S_{q,\sigma}$ need to be recomputed in every run of the *forward* procedure. The sets $S_{q,\sigma}$ computed in the last run of the *forward* procedure are stored, as they will be used in the *backward* procedure in the next section.

**Algorithm 2** Procedure, called in step 4 of *forward*, which establishes a path $\overset{a_1 \ldots a_k}{\rightsquigarrow} z$ in the NFA $N$ and returns the first state in this path.

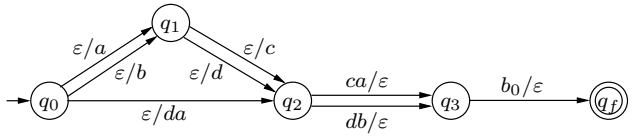
4.1 If $k = 0$, then stop and return $z$.
4.2 If there is a transition $n \xrightarrow{a_k} z$ in $N$ with $n$ non-final (there is at most one such transition), then establish a path $\overset{a_1 \ldots a_{k-1}}{\rightsquigarrow} n$ in $N$.
4.3 Else add non-final states $n_1, ..., n_k$ and transitions $n_1 \xrightarrow{a_1} \cdots n_k \xrightarrow{a_k} z$ to $N$, stop, and return $n_1$.

The idea behind the construction of $N$ is as follows. Given a transition $\theta = q \xrightarrow{\sigma/\tau} r$ in $P_0$, pushing $\tau$ onto the stack and moving to state $r$ corresponds to a path $y \overset{\tau^R}{\rightsquigarrow} r$ in $N$. A state $x$ in $N$ can jump to the first state in this path if there is a path $x \overset{\sigma^R}{\Rightarrow} q$ in $N$, because then we can reach $q$ from $x$ by pushing $\sigma$ onto the stack. By executing $\theta$, we pop $\sigma$ from the stack, leading back to $x$, then jump to $y$, and push $\tau$ onto the stack via the path $y \overset{\tau^R}{\rightsquigarrow} r$. This jump is captured in $N$ by an $\varepsilon$-transition from (every possible) $x$ to $y$. To reduce the number of $\varepsilon$-transitions in $N$, we only consider those $x$ with a path $x \overset{\sigma^R}{\Rightarrow} q$ in $N$ that does not start with an $\varepsilon$-transition.

For each transition of $P_0$ at most one path is established in $N$, and for the rest $N$ consists of $\varepsilon$-transitions (between states in such a path), so the construction of $N$ always terminates. The set $U_1$ returned at the end contains exactly the unreachable transitions in $P_0$. A proof of this fact is presented at the end of this section.

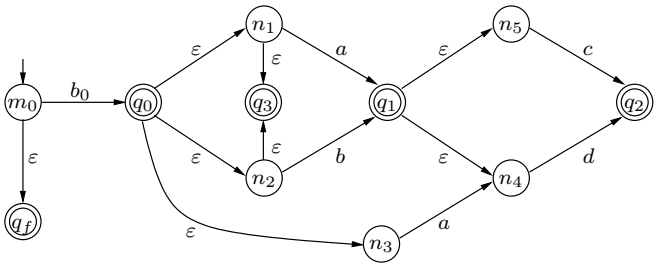
We give an example construction of NFA $N$ from a PDA. It will serve as running example in the remainder of this paper. As usual, the initial state in PDAs and NFAs is drawn with an incoming arrow, and final states with a double circle.

*Example 1.* Consider the following PDA $P_0$.




We have taken the liberty to omit the state $q_e$ from $P_0$, to keep the example small, and since the state $q_3$ is always reached with the stack $b_0$.

To determine the reachable transitions in $P_0$, the following NFA $N$ is constructed.




- Initially $N$ consists of $m_0 \xrightarrow{b_0} q_0$.
- First the paths $q_0 \xrightarrow{\varepsilon} n_1 \xrightarrow{a} q_1$ and $q_0 \xrightarrow{\varepsilon} n_2 \xrightarrow{b} q_1$ and $q_0 \xrightarrow{\varepsilon} n_3 \xrightarrow{a} n_4 \xrightarrow{d} q_2$ are added to $N$, by the transitions $q_0 \xrightarrow{\varepsilon/a} q_1$ and $q_0 \xrightarrow{\varepsilon/b} q_1$ and $q_0 \xrightarrow{\varepsilon/da} q_2$, respectively, in $P_0$. These transitions are deleted from $U_1$.
- Next the paths $q_1 \xrightarrow{\varepsilon} n_5 \xrightarrow{c} q_2$ and $q_1 \xrightarrow{\varepsilon} n_4$ are added to $N$, by the transitions $q_1 \xrightarrow{\varepsilon/c} q_2$ and $q_1 \xrightarrow{\varepsilon/d} q_2$, respectively, in $P_0$, which are deleted from $U_1$.
- Next the transitions $n_1 \xrightarrow{\varepsilon} q_3$ and $n_2 \xrightarrow{\varepsilon} q_3$ are added to $N$, by the transitions $q_2 \xrightarrow{ca/\varepsilon} q_3$ and $q_2 \xrightarrow{db/\varepsilon} q_3$, respectively, in $P_0$, which are deleted from $U_1$.
- Finally the transition $m_0 \xrightarrow{\varepsilon} q_f$ is added to $N$, by the transition $q_3 \xrightarrow{b_0/\varepsilon} q_f$ in $P_0$, which is deleted from $U_1$.

Since all transitions in $P_0$ are applied in the construction of $N$, they are all reachable. That is, at the end $U_1 = \emptyset$, and $P_1$ coincides with $P_0$.

*Correctness proof* The following two properties of $N$, which follow immediately from its construction, give insight into the structure of $N$. In particular, Lem. 3 implies that in $N$, the outgoing transitions of a final state always carry the label $\varepsilon$, while each non-final state has exactly one outgoing transition with a label from $\Gamma$.

The following two lemmas can be proved by induction on the construction of $N$.

**Lemma 2.** *For each state $x$ in $N$ there is a path $m_0 \stackrel{\sigma}{\Rightarrow} x$ in $N$, for some $\sigma$.*

**Lemma 3.** *For each state $x$ in $N$ there is exactly one path $x \stackrel{\sigma}{\rightsquigarrow} q$ in $N$, for some $\sigma$, $q$.*

*Example 4.* Lem. 3 holds for all states in the NFA $N$ from Exm. 1.

| $x$ | $m_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $q_0$ | $q_1$ | $q_2$ | $q_3$ | $q_f$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma$ | $b_0$ | $a$ | $b$ | $ad$ | $d$ | $c$ | $\varepsilon$ | $\varepsilon$ | $\varepsilon$ | $\varepsilon$ | $\varepsilon$ |
| $q$ | $q_0$ | $q_1$ | $q_1$ | $q_2$ | $q_2$ | $q_2$ | $q_0$ | $q_1$ | $q_2$ | $q_3$ | $q_f$ |

**Lemma 5.** *If there are paths $x \overset{\sigma^R}{\rightsquigarrow} q$ and $x \overset{\tau^R}{\Rightarrow} r$ in $N$, then there is a run $(q, \sigma) \to^* (r, \tau)$ of $P_1$.*

*Example 6.* $m_0 \overset{b_0}{\rightsquigarrow} q_0$ and $m_0 \overset{b_0ad}{\Rightarrow} q_2$ in the NFA $N$ from Exm. 1. We have $(q_0, b_0) \to (q_1, ab_0) \to (q_2, dab_0)$ in the corresponding PDA $P_1 = P_0$.

The following proposition is a corner stone in the correctness proof.

**Proposition 7.** *There is a path $m_0 \overset{\sigma^R}{\Rightarrow} r$ in $N$ if, and only if, $(r, \sigma)$ is reachable in $P_0$.*

*Example 8.* In the NFA $N$ from Exm. 1, the paths from $m_0$ to a final state are $m_0 \overset{b_0}{\Rightarrow} q_0$, $m_0 \overset{b_0a}{\Rightarrow} q_1$, $m_0 \overset{b_0b}{\Rightarrow} q_1$, $m_0 \overset{b_0ac}{\Rightarrow} q_2$, $m_0 \overset{b_0ad}{\Rightarrow} q_2$, $m_0 \overset{b_0bc}{\Rightarrow} q_2$, $m_0 \overset{b_0bd}{\Rightarrow} q_2$, $m_0 \overset{b_0ad}{\Rightarrow} q_2$, $m_0 \overset{b_0}{\Rightarrow} q_3$, and $m_0 \overset{\varepsilon}{\Rightarrow} q_f$. In the corresponding PDA $P_0$ the reachable configurations are $(q_0, b_0)$, $(q_1, ab_0)$, $(q_1, bb_0)$, $(q_2, cab_0)$, $(q_2, dab_0)$, $(q_2, cbb_0)$, $(q_2, dbb_0)$, $(q_2, dab_0)$, $(q_3, b_0)$, and $(q_f, \varepsilon)$.

**Theorem 9.** *The returned set $U_1$ consists of the unreachable transitions in $P_0$.*

*Proof.* Suppose $\theta = q \overset{\sigma/\tau}{\to} r$ in $P_0$ is not in $U_1$. Then during the construction of $N$, $\theta$ was used in the creation of a path $y \overset{\tau^R}{\rightsquigarrow} r$, together with one or more transitions $x \overset{\varepsilon}{\to} y$. We choose one such $x$. The construction requires that there is a path $x \overset{\sigma^R}{\Rightarrow} q$ in $N$. And by Lem. 2, $m_0 \overset{\upsilon^R}{\Rightarrow} x$ for some $\upsilon$. So by Prop. 7, $(q, \sigma\upsilon)$ is reachable in $P_0$. In this configuration, $\theta$ can be applied, to reach $(r, \tau\upsilon)$. So $\theta$ is reachable in $P_0$.

Vice versa, suppose $\theta$ is reachable in $P_0$. Then a configuration $(q, \sigma\rho)$ is reachable in $P_0$, for some $\rho$. So by Prop. 7 there is a path $m_0 \overset{(\sigma\rho)^R}{\Rightarrow} q$ in $N$. This path splits into $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\sigma^R}{\Rightarrow} q$ in $N$, where we choose $x$ such that $x \overset{\sigma^R}{\Rightarrow} q$ does not start with an $\varepsilon$-transition. In view of $\theta$, a path $y \overset{\tau^R}{\rightsquigarrow} r$ and a transition $x \overset{\varepsilon}{\to} y$ were added to $N$. And as a result, $\theta$ was deleted from $U_1$. □

*Example 10.* In Exm. 1, the fact that $U_1 = \emptyset$ means that the PDA $P_0$ does not contain unreachable transitions.

*Complexity analysis* Let $Q$ be the number of states and $T$ the number of transitions in the PDA $P_0$. For simplicity, in the analysis of the worst-case time complexity of the algorithm we assume that the number of elements popped from and pushed onto the stack in one transition, as well as the size of the stack alphabet, are bounded by some constant. Then the NFA $N$ contains at most $O(Q)$ states.

The worst-case time complexity of the *forward* procedure is $O(Q^4T)$. Firstly, the body of this procedure is run at most $O(Q^2)$ times (because $N$ contains at most $O(Q^2)$ transitions). Secondly, in each run at most $T$ times (once for each transition of $P_0$), in step 2 a backward scan over $\varepsilon$-transitions is performed, which takes at most $O(Q^2)$ (because there are at most $O(Q^2)$ $\varepsilon$-transitions).

### 3.2 Detecting which transitions can lead to the final state

If the transition $m_0 \xrightarrow{\varepsilon} q_f$ is not in $N$, then by Prop. 7 the language accepted by $P_0$ is empty. So then all transitions in $P_0$ can be reported as useless and we are done.

So we can suppose that the transition $m_0 \xrightarrow{\varepsilon} q_f$ is in $N$. Recall that the PDA $P_1$ is obtained by culling the set $U_1$ of unreachable transitions, computed by the *forward* procedure, from the input PDA $P_0$. The set $U_2$ of useless transitions in $P_1$ is constructed by running the following *backward* procedure over $\varepsilon$-transitions in $N$ that are in a set $E\backslash G$; at the start of such a run an $\varepsilon$-transition from $E\backslash G$ is copied to the set $G$, while on the other hand during the run $\varepsilon$-transitions from $N$ may be added to $E$.

Initially $U_2$, as an overapproximation, contains all transitions in $P_1$, $E = \{m_0 \xrightarrow{\varepsilon} q_f\}$ and $G = \emptyset$. We recall from step 2 of the *forward* procedure that the set $S_{q,\sigma}$ equals $\{q\}$ if $\sigma = \varepsilon$, or the states $n$ for which there exists a path $n \xrightarrow{a} y \overset{\sigma'^R}{\Longrightarrow} q$ in $N$ if $\sigma = \sigma' a$. These sets have already been computed in (the last run of) the *forward* procedure.

**Algorithm 3** Procedure *backward* to compute the transitions in the PDA $P_1$ via which the final state cannot be reached.

While $E\backslash G \neq \emptyset$ do:

1. Pick an $x \xrightarrow{\varepsilon} y \in E\backslash G$ and add it to $G$.
2. Find the path $y \overset{\tau^R}{\rightsquigarrow} r$ in $N$ (for some $\tau$, $r$); since $r$ denotes a final state, according to Lem. 3, exactly one such path exists.
3. For each transition $\theta = q \xrightarrow{\sigma/\tau} r$ in $P_1$ (for any $q$, $\sigma$) do:

3.1 If $x \notin S_{q,\sigma}$, then stop this iteration step (i.e., return to step 3).

3.2 If $\theta \in U_2$, then delete $\theta$ from $U_2$.

3.3 If $\sigma = \sigma' a$ (i.e., $\sigma \neq \varepsilon$), then add to $E$ the $\varepsilon$-transitions that occur in any path $x \xrightarrow{a} z \overset{\sigma'^R}{\Rightarrow} q$ in $N$.

Stop, and return the set $U_2$ of useless transitions in $P_1$.

The transitions in $U_1 \cup U_2$ that stem from the original PDA $P$ (i.e., not those from the preprocessing step in which $q_e$ and $q_f$ were added) are the useless transitions in $P$, so can be culled without changing the associated language.

The idea behind the *backward* procedure is that a transition $x \xrightarrow{\varepsilon} y$ in $N$ is added to $E$ when we are certain that, for some $\rho$, $\upsilon$ and $s$, there is a path $m_0 \overset{\rho^R}{\Rightarrow} x \xrightarrow{\varepsilon} y \overset{\upsilon^R}{\Rightarrow} s$ in $N$ and a run $(s, \upsilon\rho) \to^* (q_f, \varepsilon)$ of $P_1$. Each transition $x \xrightarrow{\varepsilon} y$ in $E$ may in turn give rise to adding $\varepsilon$-transitions in $N$ to $E$, and removing transitions from $U_2$. Namely, by Lem. 3 there is exactly one path $y \overset{\tau^R}{\rightsquigarrow} r$ in $N$ (for some $\tau$, $r$). For each transition $\theta = q \xrightarrow{\sigma/\tau} r$ (for any $q$, $\sigma$) in $P_1$, all $\varepsilon$-transitions in any path $x \overset{\sigma^R}{\Rightarrow} q$ in $N$ can be added to $E$. The reason is that there is a path $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\sigma^R}{\Rightarrow} q$ in $N$, as well as a run $(q, \sigma\rho) \to^* (q_f, \varepsilon)$ of $P_1$. The transition $\theta$ gives rise to the move $(q, \sigma\rho) \to (r, \tau\rho)$; in view of the paths $y \overset{\tau^R}{\rightsquigarrow} r$ and $y \overset{\upsilon^R}{\Rightarrow} s$ in $N$, by Lem. 5 there is a run $(r, \tau\rho) \to^* (s, \upsilon\rho)$ of $P_1$; and by assumption there is a run $(s, \upsilon\rho) \to^* (q_f, \varepsilon)$ of $P_1$. So if there is a path $x \overset{\sigma^R}{\Rightarrow} q$ in $N$, then $\theta$ is useful: in view of the path $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\sigma^R}{\Rightarrow} q$ in $N$, by Prop. 7, the configuration $(q, \sigma\rho)$ is reachable in $P_0$; and we argued there is a run $(q, \sigma\rho) \to^* (q_f, \varepsilon)$ of $P_1$, which

starts with an application of $\theta$. Hence $\theta$ can be removed from $U_2$. To try and avoid adding the same $\varepsilon$-transition to $E$ more than once, we only consider those paths $x \overset{\sigma^R}{\Rightarrow} q$ in $N$ that do not start with an $\varepsilon$-transition.

The *backward* procedure is executed only a finite number of times, as it is performed at most once for each $\varepsilon$-transition of $N$. The set $U_2$ returned at the end contains exactly the useless transitions in $P_1$. The corresponding theorem is presented at the end of this section, together with two propositions needed in its proof.

*Example 11.* We perform the *backward* procedure on the NFA $N$ from Exm. 1.

- Initially $E = \{m_0 \overset{\varepsilon}{\to} q_f\}$ and $G = \emptyset$.
- $m_0 \overset{\varepsilon}{\to} q_f$ is added to $G$; then $\tau = \varepsilon$ and $r = q_f$. Since $S_{q_3,b_0} = \{m_0\}$, the transition $q_3 \overset{b_0/\varepsilon}{\to} q_f$ is deleted from $U_2$, and the $\varepsilon$-transitions in paths $m_0 \overset{b_0}{\to} z \overset{\varepsilon}{\Rightarrow} q_3$ in $N$ are added to $E$: $q_0 \overset{\varepsilon}{\to} n_1 \overset{\varepsilon}{\to} q_3$ and $q_0 \overset{\varepsilon}{\to} n_2 \overset{\varepsilon}{\to} q_3$.
- $q_0 \overset{\varepsilon/a}{\to} n_1$ is added to $G$; then $\tau = a$ and $r = q_1$. Since $S_{q_0,\varepsilon} = \{q_0\}$, the transition $q_0 \overset{\varepsilon}{\to} q_1$ is deleted from $U_2$.
- $q_0 \overset{\varepsilon/b}{\to} n_2$ is added to $G$; then $\tau = b$ and $r = q_1$. Since $S_{q_0,\varepsilon} = \{q_0\}$, the transition $q_0 \overset{\varepsilon}{\to} q_1$ is deleted from $U_2$.
- $n_1 \overset{ca/\varepsilon}{\to} q_3$ is added to $G$; then $\tau = \varepsilon$ and $r = q_3$. Since $S_{q_2,ca} = \{n_1\}$, the transition $q_2 \overset{ca/\varepsilon}{\to} q_3$ is deleted from $U_2$, and the $\varepsilon$-transitions in paths $n_1 \overset{a}{\to} z \overset{c}{\Rightarrow} q_2$ in $N$ are added to $E$: $q_1 \overset{\varepsilon}{\to} n_5$.
- $n_2 \overset{db/\varepsilon}{\to} q_3$ is added to $G$; then $\tau = \varepsilon$ and $r = q_3$. Since $S_{q_2,db} = \{n_2\}$, the transition $q_2 \overset{db/\varepsilon}{\to} q_3$ is deleted from $U_2$, and the $\varepsilon$-transitions in paths $n_2 \overset{b}{\to} z \overset{d}{\Rightarrow} q_2$ in $N$ are added to $E$: $q_1 \overset{\varepsilon}{\to} n_4$.
- $q_1 \overset{\varepsilon/c}{\to} n_5$ is added to $G$; then $\tau = c$ and $r = q_2$. Since $S_{q_1,\varepsilon} = \{q_1\}$, the transition $q_1 \overset{\varepsilon}{\to} q_2$ is deleted from $U_2$.
- $q_1 \overset{\varepsilon/d}{\to} n_4$ is added to $G$; then $\tau = d$ and $r = q_2$. Since $S_{q_1,\varepsilon} = \{q_1\}$, the transition $q_1 \overset{\varepsilon}{\to} q_2$ is deleted from $U_2$.

At the end, $U_2$ consists of $q_0 \overset{\varepsilon/da}{\to} q_2$. So this transition is useless in $P_1$.

Exm. 11 shows that in step 2 of the *backward* procedure, the path $y \overset{\tau^R}{\rightsquigarrow} r$ cannot be replaced by all paths $y \overset{\tau^R}{\Rightarrow} r$ in $N$. Else $q_0 \overset{\varepsilon/da}{\to} q_2$ would be erroneously deleted from $U_2$, in view of the transition $q_0 \overset{\varepsilon}{\to} n_1$ in $E$ and the path $n_1 \overset{a}{\to} q_1 \overset{\varepsilon}{\to} n_4 \overset{d}{\to} q_2$ in $N$.

*Correctness proof* The following proposition is needed to show that only useful transitions in $P_1$ are deleted from $U_2$ during runs of the *backward* procedure.

**Proposition 12.** *Let $x \overset{\varepsilon}{\to} y$ be a transition in $E$, and $y \overset{\tau^R}{\rightsquigarrow} r$ a path in $N$. Then there is a path $m_0 \overset{\rho^R}{\Rightarrow} x$ in $N$, for some $\rho$, such that there is a run $(r, \tau\rho) \to^* (q_f, \varepsilon)$ of $P_1$.*

*Example 13.* We revisit Exm. 1. According to Exm. 11, $q_1 \overset{\varepsilon}{\to} n_5 \in E$. Moreover, $n_5 \overset{c}{\rightsquigarrow} q_2$ is a path in $N$. The only run of $P_1$ from $q_2$ with $c$ on the top of the stack to $(q_f, \varepsilon)$ is $(q_2, cab_0) \to^* (q_f, \varepsilon)$. Indeed, as predicted by Prop. 12, there is a path $m_0 \overset{b_0a}{\Rightarrow} q_1$ in $N$.

The following proposition is needed to show that all useful transitions in $P_1$ are eventually deleted from $U_2$.

**Proposition 14.** *Let* $(r, \tau\rho) \to^* (q_f, \varepsilon)$ *be a run of* $P_1$ *and* $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\varepsilon}{\to} y \overset{\tau^R}{\Rightarrow} r$ *a path in* $N$. *Then* $x \overset{\varepsilon}{\to} y$ *is in* $E$ *when the* backward *procedure terminates.*

*Example 15.* Again we revisit Exm. 1. Consider the run $(q_1, ab_0) \to^* (q_f, \varepsilon)$ of $P_1$ and the path $m_0 \overset{b_0}{\Rightarrow} q_0 \overset{\varepsilon}{\to} n_1 \overset{a}{\Rightarrow} q_1$ in $N$. As predicted by Prop. 14, $q_0 \overset{\varepsilon}{\to} n_1$ is in $E$ when the *backward* procedure terminates in Exm. 11.

**Theorem 16.** *The returned set* $U_2$ *consists of the useless transitions in* $P_1$.

*Proof.* Suppose the transition $\theta = q \overset{\sigma/\tau}{\to} r$ in $P_1$ is not in $U_2$. Since $\theta$ is in $P_1$, by the *forward* procedure, there is a path $y \overset{\tau^R}{\rightsquigarrow} r$ in $N$. Since $\theta \notin U_2$, while running the *backward* procedure, for some $x$, the transition $x \overset{\varepsilon}{\to} y$ was found to be in $E$, and a path $x \overset{\sigma^R}{\Rightarrow} q$ was found to be in $N$. In view of the transition $x \overset{\varepsilon}{\to} y$ in $E$ and the path $y \overset{\tau^R}{\rightsquigarrow} r$ in $N$, by Prop. 12, there is a path $m_0 \overset{\rho^R}{\Rightarrow} x$ in $N$, for some $\rho$, such that there is a run $(r, \tau\rho) \to^* (q_f, \varepsilon)$ of $P_1$. In view of the path $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\sigma^R}{\Rightarrow} q$ in $N$, by Prop. 7, $(q, \sigma\rho)$ is reachable in $P_1$. By applying $\theta$ to this configuration, $(r, \tau\rho)$ is reached. Since moreover there is a run $(r, \tau\rho) \to^* (q_f, \varepsilon)$ of $P_1$, $\theta$ is useful in $P_1$.

Vice versa, suppose $\theta$ is useful in $P_1$. Then there is a run $(q_0, b_0) \to^* (q, \sigma\rho) \to (r, \tau\rho) \to^* (q_f, \varepsilon)$ of $P_1$. In view of the run $(q_0, b_0) \to^* (q, \sigma\rho)$, by Prop. 7, there is a path $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\sigma^R}{\Rightarrow} q$ in $N$, where we choose $x$ such that $x \overset{\sigma^R}{\Rightarrow} q$ does not start with an $\varepsilon$-transition. Since $\theta$ is in $P_1$, by the *forward* procedure, there is a path $x \overset{\varepsilon}{\to} y \overset{\tau^R}{\rightsquigarrow} r$ in $N$. In view of the run $(r, \tau\rho) \to^* (q_f, \varepsilon)$, by Prop. 14, $x \overset{\varepsilon}{\to} y$ is eventually in $E$. In view of the paths $y \overset{\tau^R}{\rightsquigarrow} r$ and $x \overset{\sigma^R}{\Rightarrow} q$ in $N$, during the iteration of the *backward* procedure in which $x \overset{\varepsilon}{\to} y$ is added to $G$, $\theta$ is deleted from $U_2$. □

*Example 17.* In Exm. 11, the fact that $U_2 = \{q_0 \overset{\varepsilon/da}{\to} q_2\}$ means that this is the only useless transition of the PDA $P_1$.

*Complexity analysis* Computing $U_2$ takes at most $O(Q^4T)$: for each of the at most $O(Q^2)$ $\varepsilon$-transitions in $E$, and for at most $T$ transitions in $P_1$, in step 3.3 a forward scan is performed over the $\varepsilon$-transitions in $N$, which takes at most $O(Q^2)$.

## 4 Alternative approaches

We discuss three alternative approaches to detect useless transitions in PDAs.

One approach is to transform the PDA under consideration into an equivalent context-free grammar, and then determine the useless productions. Disadvantage here is that the resulting grammar tends to be much larger than the original PDA.

A second approach is to check for each transition separately whether it is useless: provide the transition with a special input symbol $\xi$, all other transitions in the PDA with empty input $\varepsilon$, and check whether the language accepted by the resulting PDA intersected with the regular language $\xi^+$ is empty. Emptiness of a PDA can be checked by getting rid of $\varepsilon$-transitions, determining an upper bound on the length of the shortest

string in the language (if any), and checking all strings up to this length. This approach however is much more expensive than the one put forward in the current paper.

The most interesting alternative is to use the algorithms from [2, 4] to compute $post*(C)$ and $pre*(C)$, with $C$ a set of configurations of the PDA under consideration: $post*(C)$ contains the configurations reachable in the PDA from a configuration in $C$, while $pre*(C)$ contains the configurations from which a configuration in $C$ can be reached in the PDA. The idea is as follows. To check whether a transition $\theta = q \stackrel{\sigma/\tau}{\rightarrow} r$ of the PDA is useless, introduce a fresh state $q_\theta$, and replace $\theta$ in the PDA by two transitions: $q \stackrel{\varepsilon/\varepsilon}{\rightarrow} q_\theta$ and $q_\theta \stackrel{\sigma/\tau}{\rightarrow} r$. Then $\theta$ is useless if and only if

$$post*(\{(q_0, \varepsilon)\}) \cap (\{q_\theta\} \times \Gamma^*) \cap pre*(F \times \Gamma^*) = \emptyset .$$

The $post*$ set captures the reachable configurations, while the $pre*$ set captures the configurations from which a final state can be reached. The set $\{q_\theta\} \times \Gamma^*$ checks whether $\theta$ is used on a path from initial configuration to final state.

## 5 Implementation and performance comparison

We assessed an implementation of our algorithm with a test suite of randomly generated PDAs. The only real-world PDA, with 294 transitions, was obtained from the grammar of the programming language C. This resulted in an NFA with 339 states and 1030 transitions, of which 695 $\varepsilon$-transitions, and took 3.56 seconds on a 2GHz processor.

Achieving this performance required two optimizations, both limiting the influence of $\varepsilon$-transitions. The first concerns determining the set of states leading to $q$ in step 2 of the *forward* procedure. This set is constructed by following paths backwards from $q$, which may lead through webs of $\varepsilon$-transitions, causing a considerable slow-down. These $\varepsilon$-transitions were created in step 5 of the *forward* procedure. The optimization consists of computing for each state $s$, in step 5, the set $B(s)$ of states that can reach $s$ through $\varepsilon$-transitions only. If more $\varepsilon$-transitions are added in step 5 during a next iteration, $B$ is updated. The second optimization concerns memoization of $\varepsilon$-transitions as they are encountered on paths to $q$ in step 3.3 of the *backward* procedure.

Furthermore, the third alternative approach described in Section 4 was implemented, and its performance was compared with the implementation of our algorithm. Actually four versions of that approach were implemented. In the first version, transitions in the input PDA that pop more than one element from the stack or push more than two elements onto the stack are first broken up into multiple transitions that pop one element from the stack and push at most two elements onto the stack. In the second version, the computations of $pre*$ and $post*$ have been redesigned to avoid this preprocessing step; and unlike the first version, the second version can cope with PDA transitions that pop no elements from the stack. The third and fourth versions make use of the fact that the redesigned $pre*$ and $post*$ procedures are each other's duals and can be defined in terms of each other. The first three versions exhibit a worst-case time complexity of $O(S^3T^4)$ and the fourth version $O(S^2T^4)$, with $S$ the maximum stack string length in the PDA transitions. (We take $S$ into account here because of this small distinction.)

It should be noted that the last three versions of the alternative approach do not terminate for certain input PDAs. This is due to the fact that the redesigned $pre*$ and $post*$ procedures, which eliminate the need for input preprocessing, result in an infinite

loop whenever the input PDA contains certain loops. PDAs on which at least one of these algorithms did not terminate have been excluded from the comparison in Table 1.

The prototype implementations of the alternative approach have certain drawbacks, which are described in [3]. The most relevant to the performance comparison is the maximum allowed set size (*L_SIZE*). For each transition in the PDA, the alternative approach computes the intersection of two NFAs in order to determine whether the transition is useless. As a result, the set of states overflows fairly quickly even for small inputs. *L_SIZE* has to be adjusted for each input PDA at compile time. This posed a limit on the size of the PDAs that could be tested. An additional parameter, *M_SIZE*, exists and serves a similar purpose to that of *L_SIZE*. For certain inputs this parameter has to be adjusted as well. To keep the comparison fair, all implementations were tested using the same *M_SIZE* and *L_SIZE* for a particular input. Some indicative results are presented in Table 1. $Q$ denotes the number of states, $T$ the number of transitions, and $S$ the maximum stack string length in the PDA. The PDAs were generated using the *PDAgen* tool. The number of useless transitions they contain, which ranges from none to all transitions, appears to have no impact on the running time of the algorithms.

| **Input PDA** | | | | | **Algorithm (sec)** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $Q$ | $T$ | $S$ | *M_SIZE* | *L_SIZE* | ours | alt. 1 | alt. 2 | alt. 3 | alt. 4 |
| 4 | 5 | 5 | 600 | 1200 | 0.38 | 1.33 | 1.08 | 1.29 | 1.00 |
| 6 | 6 | 9 | 600 | 2000 | 0.66 | 3.14 | 2.16 | 2.32 | 2.12 |
| 11 | 10 | 5 | 600 | 1200 | 0.32 | 2.82 | 2.13 | 2.55 | 1.89 |
| 7 | 10 | 7 | 600 | 1200 | 0.38 | 3.59 | 2.36 | 2.56 | 2.18 |
| 8 | 10 | 8 | 600 | 2000 | 0.66 | 4.84 | 3.36 | 3.77 | 3.23 |
| 7 | 10 | 12 | 600 | 2800 | 1.02 | 16.86 | 11.57 | 6.29 | 11.17 |
| 16 | 20 | 6 | 600 | 2000 | 0.65 | 18.21 | 12.68 | 11.58 | 7.69 |
| 6 | 10 | 17 | 600 | 5000 | 2.06 | 34.48 | 25.56 | 11.15 | 26.14 |
| 22 | 31 | 5 | 600 | 3000 | 1.04 | 26.35 | 22.86 | 26.61 | 19.20 |
| 9 | 20 | 10 | 600 | 6000 | 2.50 | 78.33 | 159.49 | 45.00 | 99.24 |
| 8 | 20 | 13 | 600 | 8000 | 3.83 | 156.01 | 108.35 | 90.24 | 116.95 |
| 8 | 20 | 13 | 600 | 10000 | 5.96 | 274.46 | 325.94 | 141.99 | 376.41 |

**Table 1.** Performance of our algorithm against the alternative approach

In Table 1, entries have been sorted on the basis of the $S^3T^4$ size of the input PDA. This performance analysis shows that the algorithm presented in the current paper clearly outperforms the four implementations based on the third approach in Section 4. Our algorithm's running time shows a favorable running time when the size of the input PDA is increased, owing to the fact that its worst-case time complexity $O(Q^4T)$ only occurs in the unlikely case that the NFA is constructed over a large number of iterations, is saturated with $\varepsilon$-transitions, and contains a lot of backward nondeterminism. Rather its efficiency is affected by an increase of *L_SIZE*, while increasing this parameter is actually only needed for the implementation of the alternative approach in the above test cases. To illustrate this and indicate how our algorithm performs when PDA size increases, Table 2 shows the performance of our algorithm on different-sized inputs. For larger input PDAs *M_SIZE* and *L_SIZE* have been adjusted as necessary.

A detailed account of the implementation of the alternative approach and the performance comparison is presented in [3].

| Input PDA | | | | Algorithm (sec) |
|---|---|---|---|---|
| $Q$ | $T$ | *M_SIZE* | *L_SIZE* | our alg. |
| 16 | 20 | 600 | 1200 | 0.38 |
| 26 | 40 | 600 | 1200 | 0.45 |
| 30 | 50 | 600 | 1200 | 0.47 |
| 29 | 60 | 600 | 1200 | 0.50 |
| 33 | 70 | 600 | 1200 | 0.68 |
| 44 | 80 | 600 | 1200 | 0.68 |
| 60 | 120 | 600 | 1200 | 1.39 |
| 86 | 140 | 600 | 1200 | 1.33 |
| 3 | 294 | 600 | 1200 | 3.56 |
| 107 | 200 | 1200 | 2000 | 5.54 |
| 128 | 300 | 1200 | 2000 | 5.58 |
| 155 | 400 | 2000 | 3000 | 31.02 |
| 170 | 500 | 2000 | 3000 | 39.12 |
| 185 | 600 | 3000 | 4000 | 75.26 |
| 250 | 600 | 3000 | 4000 | 74.77 |

**Table 2.** Scalability of our algorithm

*Acknowledgments* Javier Esparza proposed the alternative approach using $pre*$ and $post*$. Jörg Endrullis provided a useful suggestion for the efficient implementation of this alternative approach.

## References

1. E. Bertsch and M.-J. Nederhof. Regular closure of deterministic languages. *SIAM Journal on Computing*, 29(1):81–102, 1999.
2. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR'97*, *LNCS* 1243, pp. 135–150. Springer, 1997.
3. E. Chatzikalymnios. Comparison of two algorithms for detecting useless transitions in pushdown automata. BSc thesis, Vrije Universiteit Amsterdam, 2016. `http://www.cs.vu.nl/~tcs/chatzikalymnios-bscthesis.pdf`
4. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *CAV'00*, *LNCS* 1855, pp. 232–247. Springer, 2000.
5. A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. In *INFINITY'97*, *ENTCS* 9, pp. 27–37. Elsevier, 1997.
6. J. Goldstine, J.K. Price, and D. Wotschke. A pushdown automaton or a context-free grammar - which is more economical? *Theoretical Computer Science*, 18:33–40, 1982.
7. C. Griffin. A note on deciding controllability in pushdown systems. *IEEE Transactions on Automatic Control*, 51(2):334–337, 2006.
8. D. Grune and C. Jacobs. *Parsing Techniques - A Practical Guide* (2nd ed.). Springer, 2008.
9. P. Linz. *An Introduction to Formal Languages and Automata*. Jones & Bartlett, 2011.
10. M.-J. Nederhof and E. Bertsch. Linear-time suffix parsing for deterministic languages. *Journal of the ACM*, 43(3):524–554, 1996.
11. I. Walukiewicz. Pushdown processes: Games and model checking. In *CAV'96*, *LNCS* 1102, pp. 62–74. Springer, 1996.

# A Proofs of propositions and lemmas

This appendix contains the proofs of the propositions and lemmas in the paper.

First we prove Lem. 2 and 3 in one go: For each state $x$ in $N$, there is a path $m_0 \overset{\sigma}{\Rightarrow} x$ in $N$, for some $\sigma$, and there is exactly one path $x \overset{\sigma}{\rightsquigarrow} q$ in $N$, for some $\sigma$, $q$.

*Proof.* We apply by induction on the construction of $N$.

Initially the desired properties hold trivially, because then $N$ only consists of the transition $m_0 \overset{b_0}{\to} q_0$.

When a path $y \overset{\tau}{\rightsquigarrow} q$ is added to $N$, then also a transition $x \overset{\varepsilon}{\to} y$ is added, where $x$ was already present in $N$. Since by induction $m_0 \overset{\rho}{\Rightarrow} x$ for some $\rho$, clearly Lem. 2 still holds in the extended NFA $N$. Moreover, since only (part of) the path $y \overset{\tau}{\rightsquigarrow} q$ is added to $N$, together with some $\varepsilon$-transitions to the first state in this path, clearly Lem. 3 still holds in the extended NFA $N$. □

We proceed with the proof of Lem. 5: If there are paths $x \overset{\sigma^R}{\rightsquigarrow} q$ and $x \overset{\tau^R}{\Rightarrow} r$ in $N$, then there is a run $(q, \sigma) \to^* (r, \tau)$ of $P_1$.

*Proof.* We apply induction on the path $x \overset{\tau^R}{\Rightarrow} r$. A well-founded partial ordering on paths in $N$ is defined as follows. Suppose that during the construction of $N$, each $\varepsilon$-transition is at its creation provided with a sequence number, being one higher than the previously created $\varepsilon$-transition (the first created $\varepsilon$-transition gets sequence number 0). Now $y_1 \overset{\rho_1}{\Rightarrow} s_1$ is defined to be smaller than $y_2 \overset{\rho_2}{\Rightarrow} s_2$ if:

(i) either $y_2 \overset{\rho_2}{\Rightarrow} s_2$ contains an $\varepsilon$-transition with a higher sequence number than any of the $\varepsilon$-transitions in $y_1 \overset{\rho_1}{\Rightarrow} s_1$;
(ii) or $y_2 \overset{\rho_2}{\Rightarrow} s_2$ contains $y_1 \overset{\rho_1}{\Rightarrow} s_1$ as a strict subsequence.

In the base case of the induction, the path $x \overset{\tau^R}{\Rightarrow} r$ consists of zero transitions, so that $\tau = \varepsilon$ and $x = r$. By Lem. 3, the path $r \overset{\sigma^R}{\rightsquigarrow} q$ in $N$ implies $\sigma = \varepsilon$ and $r = q$. And trivially there is a run $(q, \varepsilon) \to^* (q, \varepsilon)$ of $P_1$.

In the inductive case, the path $x \overset{\tau^R}{\Rightarrow} r$ consists of one or more transitions. We distinguish two cases, depending on whether the path $x \overset{\tau^R}{\Rightarrow} r$ starts with an $\varepsilon$-transition.

CASE 1: The path $x \overset{\tau^R}{\Rightarrow} r$ is of the form $x \overset{a}{\to} y \overset{\tau'^R}{\Rightarrow} r$, with $\tau = \tau' a$. By Lem. 3, $x$ is non-final, and $x \overset{\sigma^R}{\rightsquigarrow} q$ is of the form $x \overset{a}{\to} y \overset{\sigma'^R}{\rightsquigarrow} q$, with $\sigma = \sigma' a$. By (ii), $y \overset{\tau'^R}{\Rightarrow} r$ is smaller than $x \overset{\tau^R}{\Rightarrow} r$. Since there are paths $y \overset{\sigma'^R}{\rightsquigarrow} q$ and $y \overset{\tau'^R}{\Rightarrow} r$ in $N$, by induction, there is a run $(q, \sigma') \to^* (r, \tau')$ of $P_1$. So there is a run $(q, \sigma' a) \to^* (r, \tau' a)$ of $P_1$.

CASE 2: The path $x \overset{\tau^R}{\Rightarrow} r$ is of the form $x \overset{\varepsilon}{\to} y \overset{\tau^R}{\Rightarrow} r$. Suppose the transition $x \overset{\varepsilon}{\to} y$ was created in $N$ due to a transition $s \overset{\upsilon/\zeta}{\to} t$ in $P_1$. Then there must be paths $x \overset{\upsilon^R}{\Rightarrow} s$ and $y \overset{\zeta^R}{\rightsquigarrow} t$ in $N$. By (i), $x \overset{\upsilon^R}{\Rightarrow} s$ is smaller than $x \overset{\tau^R}{\Rightarrow} r$, because $x \overset{\varepsilon}{\to} y$ has a higher sequence number than any of the $\varepsilon$-transitions in the path $x \overset{\upsilon^R}{\Rightarrow} s$. Since $x \overset{\sigma^R}{\rightsquigarrow} q$ and $x \overset{\upsilon^R}{\Rightarrow} s$, by induction, there is a run $(q, \sigma) \to^* (s, \upsilon)$ of $P_1$. The transition $s \overset{\upsilon/\zeta}{\to} t$ in $P_1$ gives rise

to the move $(s, \upsilon) \to (t, \zeta)$. By (ii), $y \overset{\tau^R}{\Rightarrow} r$ is smaller than $x \overset{\tau^R}{\Rightarrow} r$. Since $y \overset{\zeta^R}{\rightsquigarrow} t$ and $y \overset{\tau^R}{\Rightarrow} r$, by induction, there is a run $(t, \zeta) \to^* (r, \tau)$ of $P_1$. Concluding, there is a run $(q, \sigma) \to^* (r, \tau)$ of $P_1$. □

We now present the proof of Prop. 7: There is a path $m_0 \overset{\sigma^R}{\Rightarrow} r$ in $N$ if, and only if, $(r, \sigma)$ is reachable in $P_0$.

*Proof.* ($\Rightarrow$) The transition $m_0 \overset{b_0}{\to} q_0$ is in $N$, and by assumption there is a path $m_0 \overset{\sigma^R}{\Rightarrow} r$ in $N$. So by Lem. 5, there is a run $(q_0, b_0) \to^* (r, \sigma)$ of $P_1$, and so of $P_0$.

($\Leftarrow$) By induction on the number of moves in a run $(q_0, b_0) \to^* (r, \sigma)$ of $P_0$. In the base case, no transition is applied, so $r = q_0$ and $\sigma = b_0$. The transition $m_0 \overset{b_0}{\to} q_0$ is in $N$.

In the inductive case, suppose $q \overset{\tau/\upsilon}{\to} r$ is the last transition applied in the run $(q_0, b_0) \to^* (r, \sigma)$. Then $\sigma = \upsilon\rho$ for some $\rho$, and there is a run $(q_0, b_0) \to^* (q, \tau\rho)$ of $P_0$. Since this run takes one move less than the one to $(r, \sigma)$, by induction there is a path $m_0 \overset{(\tau\rho)^R}{\Rightarrow} q$ in $N$. This path splits into $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\tau^R}{\Rightarrow} q$ in $N$, where we choose $x$ such that $x \overset{\tau^R}{\Rightarrow} q$ does not start with an $\varepsilon$-transition. In view of the transition $q \overset{\tau/\upsilon}{\to} r$ in $P_0$ and the path $x \overset{\tau^R}{\Rightarrow} q$ in $N$, during the construction of $N$, a path $y \overset{\upsilon^R}{\rightsquigarrow} r$ was created, together with the transition $x \overset{\varepsilon}{\to} y$. Concluding, there is a path $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\varepsilon}{\to} y \overset{\upsilon^R}{\rightsquigarrow} r$, so $m_0 \overset{\sigma^R}{\Rightarrow} r$, in $N$. □

We proceed with the proof of Prop. 12: Let $x \overset{\varepsilon}{\to} y$ be a transition in $E$, and $y \overset{\tau^R}{\rightsquigarrow} r$ a path in $N$. Then there is a path $m_0 \overset{\rho^R}{\Rightarrow} x$ in $N$, for some $\rho$, such that there is a run $(r, \tau\rho) \to^* (q_f, \varepsilon)$ of $P_1$.

*Proof.* By induction on the construction of $E$. In the base case, $E = \{m_0 \overset{\varepsilon}{\to} q_f\}$. So $x = m_0$, $y = q_f$, $\tau = \varepsilon$ and $r = q_f$. So we can take $\rho = \varepsilon$.

In the inductive case, suppose that due to a transition $z \overset{\varepsilon}{\to} w$ in $E$, a path $w \overset{\upsilon^R}{\rightsquigarrow} s$ in $N$, and a transition $\theta = q \overset{\sigma/\upsilon}{\to} s$ in $P_1$, $\varepsilon$-transitions in paths $z \overset{\sigma^R}{\Rightarrow} q$ in $N$ are added to $E$. We show that the proposition holds for any transition $x \overset{\varepsilon}{\to} y$ in any path $z \overset{\sigma^R}{\Rightarrow} q$; say $z \overset{\sigma_2^R}{\Rightarrow} x \overset{\varepsilon}{\to} y \overset{\sigma_1^R}{\Rightarrow} q$ with $\sigma = \sigma_1\sigma_2$. Since $y \overset{\tau^R}{\rightsquigarrow} r$ and $y \overset{\sigma_1^R}{\Rightarrow} q$ are paths in $N$, by Lemma 5, there is a run $(r, \tau) \to^* (q, \sigma_1)$ of $P_1$. And by $\theta$, $(q, \sigma) \to (s, \upsilon)$ in $P_1$. The transition $z \overset{\varepsilon}{\to} w$ was already present in $E$ before the current extension of $E$. Since $w \overset{\upsilon^R}{\rightsquigarrow} s$ is a path in $N$, by induction there is a path $m_0 \overset{\rho^R}{\Rightarrow} z$ in $N$, for some $\rho$, such that there is a run $(s, \upsilon\rho) \to^* (q_f, \varepsilon)$ of $P_1$. Concluding, there is a path $m_0 \overset{\rho^R}{\Rightarrow} z \overset{\sigma_2^R}{\Rightarrow} x$ in $N$, and a run $(r, \tau\sigma_2\rho) \to^* (q, \sigma_1\sigma_2\rho) \to (s, \upsilon\rho) \to^* (q_f, \varepsilon)$ of $P_1$. □

We finish with the proof of Prop. 14. First we formulate and prove a lemma needed in the proof of this proposition. The idea behind this lemma is that if $(r, \tau\rho)$ is a reachable configuration of $P_1$, and $\rho \neq \varepsilon$, then any run $(r, \tau\rho) \to^* (q_f, \varepsilon)$ of $P_1$ contains a

move $(q, \sigma\rho) \to (s, \upsilon\rho_2)$ in which a non-empty part $\rho_1$ of $\rho = \rho_1\rho_2$ is removed from the stack.

**Lemma 18.** *Let $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\tau^R}{\Rightarrow} r$ be a path in $N$, and $(r, \tau\rho) \to^* (q_f, \varepsilon)$ a run of $P_1$, with $\rho \neq \varepsilon$. Then there is a path $x \overset{\sigma^R}{\Rightarrow} q$ in $N$ for some $\sigma$ and $q$, and the path $m_0 \overset{\rho^R}{\Rightarrow} x$ in $N$ splits into $m_0 \overset{\rho_2^R}{\Rightarrow} y \overset{\rho_1^R}{\Rightarrow} x$ for some $\rho_2$, $y$ and $\rho_1 \neq \varepsilon$, where $y \overset{\rho_1^R}{\Rightarrow} x$ does not start with an $\varepsilon$-transition, and there is a path $y \overset{\varepsilon}{\to} z \overset{\upsilon^R}{\rightsquigarrow} s$ in $N$ for some $z$, $\upsilon$ and $s$, where $q \overset{\sigma\rho_1/\upsilon}{\to} s$ is a transition of $P_1$ that is applied to the configuration $(q, \sigma\rho)$ in the run $(r, \tau\rho) \to^* (q_f, \varepsilon)$.*

*Proof.* By induction on the number of moves in the run $(r, \tau\rho) \to^* (q_f, \varepsilon)$. Note that $\rho \neq \varepsilon$ implies $r \neq q_f$. Let $\theta = r \overset{\zeta/\pi}{\to} t$ in $P_1$ be the first transition that is applied in the run $(r, \tau\rho) \to^* (q_f, \varepsilon)$. We distinguish two cases:

CASE 1: $\zeta = \tau\rho_1$ with $\rho = \rho_1\rho_2$, for some $\rho_1 \neq \varepsilon$ and $\rho_2$. We take $\sigma = \tau$ and $q = r$. The path $m_0 \overset{\rho^R}{\Rightarrow} x$ in $N$ splits into $m_0 \overset{\rho_2^R}{\Rightarrow} y \overset{\rho_1^R}{\Rightarrow} x$, where we choose $y$ such that $y \overset{\rho_1^R}{\Rightarrow} x$ does not start with an $\varepsilon$-transition. Since there is a path $y \overset{\rho_1^R}{\Rightarrow} x \overset{\tau^R}{\Rightarrow} r$ in $N$, $\zeta = \tau\rho_1$, and $\theta$ is in $P_1$, by the *forward* procedure, there is a path $y \overset{\varepsilon}{\to} z \overset{\pi^R}{\rightsquigarrow} t$ in $N$. So if we take $s = t$ and $\upsilon = \pi$, we are done.

CASE 2: $\tau = \zeta\tau'$ for some $\tau'$. The path $x \overset{\tau^R}{\Rightarrow} r$ in $N$ splits into $x \overset{\tau'^R}{\Rightarrow} w \overset{\zeta^R}{\Rightarrow} r$, where we choose $w$ such that $w \overset{\zeta^R}{\Rightarrow} r$ does not start with an $\varepsilon$-transition. Since $\theta$ is in $P_1$, by the *forward* procedure, there is a path $w \overset{\varepsilon}{\to} v \overset{\pi^R}{\rightsquigarrow} t$ in $N$. So there is a path $m_0 \overset{\rho^R}{\Rightarrow} x \overset{(\pi\tau')^R}{\Rightarrow} t$ in $N$. Moreover, the run $(r, \tau\rho) \to^* (q_f, \varepsilon)$ is of the form $(r, \tau\rho) \to (t, \pi\tau'\rho) \to^* (q_f, \varepsilon)$. By induction, there is a path $x \overset{\sigma^R}{\Rightarrow} q$ in $N$, and the path $m_0 \overset{\rho^R}{\Rightarrow} x$ in $N$ splits into $m_0 \overset{\rho_2^R}{\Rightarrow} y \overset{\rho_1^R}{\Rightarrow} x$ with $\rho_1 \neq \varepsilon$, where $y \overset{\rho_1^R}{\Rightarrow} x$ does not start with an $\varepsilon$-transition, and there is a path $y \overset{\varepsilon}{\to} z \overset{\upsilon^R}{\rightsquigarrow} s$ in $N$, where $q \overset{\sigma\rho_1/\upsilon}{\to} s$ is a transition of $P_1$ that is applied to the configuration $(q, \sigma\rho)$ in the run $(t, \pi\tau'\rho) \to^* (q_f, \varepsilon)$. □

*Example 19.* Again we revisit Exm. 1. Consider the path $m_0 \overset{b_0a}{\Rightarrow} n_5 \overset{c}{\Rightarrow} q_2$ in $N$ and the run $(q_2, cab_0) \to^* (q_f, \varepsilon)$ of $P_1$. We select the path $n_5 \overset{c}{\Rightarrow} q_2$ in $N$. Moreover, we split the path $m_0 \overset{b_0a}{\Rightarrow} n_5$ in $N$ into $m_0 \overset{\varepsilon}{\Rightarrow} n_1 \overset{a}{\Rightarrow} n_5$; note that $n_1 \overset{a}{\Rightarrow} n_5$ does not start with an $\varepsilon$-transition. We select the path $n_1 \overset{\varepsilon}{\to} q_3 \overset{\varepsilon}{\rightsquigarrow} q_3$ in $N$. The transition $q_2 \overset{ca/\varepsilon}{\to} q_3$ of $P_1$ is applied in the run $(q_2, cab_0) \to^* (q_f, \varepsilon)$.

Finally we prove Prop. 14: Let $(r, \tau\rho) \to^* (q_f, \varepsilon)$ be a run of $P_1$ and $m_0 \overset{\rho^R}{\Rightarrow} x \overset{\varepsilon}{\to} y \overset{\tau^R}{\Rightarrow} r$ a path in $N$. Then $x \overset{\varepsilon}{\to} y$ is in $E$ when the *backward* procedure terminates.

*Proof.* We apply induction on the length of $\rho$. In the base case, $\rho = \varepsilon$. Since $m_0 \overset{b_0}{\to} q_0$ is a transition of $N$, and $q_e \overset{b_0/\varepsilon}{\to} q_f$ is the only transition of $P_0$ with an occurrence of $b_0$, by the *forward* procedure, the only possible outgoing $\varepsilon$-transition of $m_0$ in $N$ is

$m_0 \xrightarrow{\varepsilon} q_f$. Hence the path $m_0 \overset{\varepsilon}{\Rightarrow} x \xrightarrow{\varepsilon} y \overset{\tau^R}{\Rightarrow} r$ implies that $x = m_0$, $y = q_f$, $\tau = \varepsilon$ and $r = q_f$. Since $m_0 \xrightarrow{\varepsilon} q_f$ is in $N$, initially $E = \{m_0 \xrightarrow{\varepsilon} q_f\}$.

In the inductive case, $\rho \neq \varepsilon$. Since there is a path $m_0 \overset{\rho^R}{\Rightarrow} x \xrightarrow{\varepsilon} y \overset{\tau^R}{\Rightarrow} r$ in $N$ and a run $(r, \tau\rho) \rightarrow^* (q_f, \varepsilon)$ of $P_1$, by Lem. 18, there is a path $y \overset{\sigma^R}{\Rightarrow} q$ in $N$ for some $\sigma$ and $q$, and the path $m_0 \overset{\rho^R}{\Rightarrow} x$ in $N$ splits into $m_0 \overset{\rho_2^R}{\Rightarrow} z \overset{\rho_1^R}{\Rightarrow} x$ for some $\rho_2$, $z$ and $\rho_1 \neq \varepsilon$, where $z \overset{\rho_1^R}{\Rightarrow} x$ does not start with an $\varepsilon$-transition, and there is a path $z \xrightarrow{\varepsilon} w \overset{\upsilon^R}{\rightsquigarrow} s$ in $N$ for some $w$, $\upsilon$ and $s$, where $q \overset{\sigma\rho_1/\upsilon}{\longrightarrow} s$ is a transition of $P_1$ that is applied to the configuration $(q, \sigma\rho)$ in the run $(r, \tau\rho) \rightarrow^* (q_f, \varepsilon)$. Since $\rho_2$ is shorter than $\rho$, by induction, $z \xrightarrow{\varepsilon} w$ is eventually in $E$. During the iteration of the *backward* procedure in which $z \xrightarrow{\varepsilon} w$ is added to $G$, in view of the transition $q \overset{\sigma\rho_1/\upsilon}{\longrightarrow} s$ in $P_1$ and the paths $w \overset{\upsilon^R}{\rightsquigarrow} s$ and $z \overset{\rho_1^R}{\Rightarrow} x \xrightarrow{\varepsilon} y \overset{\sigma^R}{\Rightarrow} q$ in $N$, the $\varepsilon$-transitions in the latter path are added to $E$. So in particular, $x \xrightarrow{\varepsilon} y$ is added to $E$. □