A Pumping Lemma for Collapsible Pushdown Graphs of Level 2*

Alexander Kartzow

Universität Leipzig, Institut für Informatik
Johannisgasse 26, 04103 Leipzig

Abstract

We present a pumping lemma for the class of collapsible pushdown graphs of level 2. This pumping lemma even applies to the ε-contractions of level 2 collapsible pushdown graphs. Our pumping lemma also improves the bounds of Hayashi's pumping lemma for indexed languages.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Collapsible pushdown graph, ε-contraction, pumping lemma

Digital Object Identifier 10.4230/LIPIcs.CSL.2011.322

1 Introduction

Recently, generalisations of pushdown systems have gained attention for the verification of higher-order functional programs. This stems from the fact that collapsible higher-order pushdown systems generate exactly the same trees as higher-order recursion schemes [6]. Recursion Schemes can be fruitfully applied in the verification of functional programs [11]. The correspondence between collapsible pushdown trees and higher-order recursion schemes improves a result of Knapik et al. [10] showing that higher-order pushdown systems generate the same trees as *safe* higher-order recursion schemes. Safety is a syntactic condition whose semantical status is still open: it is conjectured that there is a recursion scheme that generates a tree which is not generated by any safe scheme.[1]

Also from a model-theoretic point of view, the classes of collapsible and higher-order pushdown systems are interesting. Carayol and Wöhrle [3] proved that the ε-contractions of graphs generated by higher-order pushdown systems are exactly the graphs in the Caucal-hierarchy. Thus, all these graphs have decidable monadic second-order theories. Collapsible pushdown graphs display a rather different behaviour: even on the second level of the hierarchy, there is a graph with undecidable monadic second-order theory. Nevertheless, the first-order model checking on level 2 collapsible pushdown graphs is decidable because all these graphs are tree-automatic [8]. Recently, Broadbent [2] proved that even first-order logic is undecidable on the collapsible pushdown graphs of level 3. Moreover, Hague et al. [6] showed that the modal μ-calculus is decidable on all collapsible pushdown graphs. These decidability results give collapsible pushdown graphs a unique status among natural classes of graphs. There is (almost[2]) no other known natural class of graphs with decidable modal μ-calculus model checking but undecidable monadic second-order theories.

* This research was carried out while the author was working at TU Darmstadt and funded by the DFG grant OT 147/5-1

[1] Recently, Parys [13] showed that the uniform safety conjecture is true: there is a level 2 recursion scheme which generates a tree that is not generated by any safe level 2 scheme

[2] Nested pushdown trees share the same status. But these form in some sense a subclass of collapsible pushdown graphs (cf. [9])

© Alexander Kartzow;
licensed under Creative Commons License NC-ND

Even though (collapsible) pushdown systems generate important classes of graphs, useful characterisations of their structure are still rare. We [8] showed that the second-level of the collapsible pushdown graph hierarchy is tree-automatic. Nevertheless, we still miss techniques for disproving membership in the collapsible pushdown hierarchy. In classical automata-theory, pumping lemmas form a good tool for disproving membership in languages defined by finite automata or pushdown systems. On higher levels, similar results are still missing. The only results in this direction that are known to the author are a pumping lemma of Hayashi [7] and a shrinking lemma of Gilman [5], both for indexed languages. Since the class of string-languages accepted by the second level of the collapsible higher-order pushdown hierarchy is the class of indexed languages, this is a first step towards pumping on higher-order pushdown systems. Blumensath [1] published an extension of this pumping lemma to all levels of the higher-order pushdown hierarchy. Unfortunately, there is an irrecoverable error in his proof (cf. [12]). Thus, the question for pumping lemmas for higher-order pushdown systems of level at least 3 is still open. Moreover, even for the second level of collapsible pushdown graphs no pumping lemma was known so far.

In this paper, we close the latter gap. As already mentioned, collapsible pushdown graphs are tree-automatic, i.e., there is an encoding of configurations in trees such that a (finite tree-)automaton accepts the encodings of configurations reachable form the initial one. Of course, the regular pumping lemma applies to this finite automaton. Since accepting runs of this automaton encode runs of the collapsible pushdown system, the trees obtained by regular pumping can be turned into runs of the collapsible pushdown system. Thus, the existence of a large configuration in the collapsible pushdown graph implies the existence of infinitely many runs. A refinement of this argument yields a pumping lemma for all ε-contractions of collapsible pushdown graphs of level 2.

1.1 Outline of the paper

In the next section, we recall the standard notion of trees, finite tree-automata and the pumping lemma for finite tree-automata. At the end of that section we introduce the general approach how tree-automaticity of the reachability predicate can be turned into a pumping lemma. In Section 3 we present the notion of level 2 collapsible pushdown graphs. Furthermore, we recall the main result from [8], i.e., we present an encoding of configurations of 2-CPG in trees that turns the set of reachable configurations into a regular set of trees. We also recall what an automaton determining the reachable configurations looks like. In Section 4 we compute the specific bounds of the pumping lemma for level 2 collapsible pushdown graphs obtained from our general approach. We then refine this result in Section 5 to ε-contractions of such graphs. In Section 6 we apply our pumping lemma and prove that certain trees are not ε-contractions of any 2-CPG. Section 7 contains concluding remarks and points to open questions.

2 A Pumping Lemma for Structures with Automatic Reachability Relation

A Σ-*labelled tree* is a function $T : D \to \Sigma$ for a finite set $D \subseteq \{0,1\}^*$ which is closed under prefixes. For $d \in D$ we denote by T_d the *subtree rooted at* d. It is useful to define trees inductively by describing their left and right subtrees. For this purpose we fix the following notation. Let $\hat{T}$ and T' be Σ-labelled trees and $\sigma \in \Sigma$. Then we write $T := \sigma(\hat{T}, T')$ for the

Σ-labelled tree T with the following three properties

1. $T(\varepsilon) = \sigma$, 2. $T_0 = \hat{T}$, and 3. $T_1 = T'$.

Finally, let $\mathrm{dp}(T) := \max\{|d| : d \in \mathrm{dom}(T)\} + 1$ be the *depth* of T. A straightforward induction gives a bound on the number of nodes of a tree of a fixed depth.

▶ **Lemma 1.** *If T is a tree of depth d, then $|dom(T)| \leq 2^d - 1$.*

▶ **Corollary 2.** *There are at most $(|\Sigma| + 1)^{2^d - 1}$ many Σ-labelled trees of depth d.*

We briefly recall the notion of a (finite tree-) automaton and the pumping lemma for these automata.

▶ **Definition 3.** A *(finite tree-)automaton* is a tuple $\mathcal{A} = (\Sigma, Q, \Delta, q_I)$ where Σ is a finite alphabet, Q a finite set of states with a distinguished symbol $\bot \in Q$, $\Delta \subseteq Q \times \Sigma \times Q \times Q$ a transition relation and $q_I \in Q$ the initial state. An *accepting run* of $\mathcal{A}$ on a Σ-labelled tree T is a map $\rho : \{0,1\}^* \to Q$ such that
1. $(\rho(d), T(d), \rho(d0), \rho(d1)) \in \Delta$ for all $d \in \mathrm{dom}(T)$ and
2. $\rho(\varepsilon) = q_I$ and $\rho(d) = \bot$ for all $d \in \{0,1\}^* \setminus \mathrm{dom}(T)$.

The *language* accepted by $\mathcal{A}$ is $L(\mathcal{A}) := \{T : \exists$ accepting run of $\mathcal{A}$ on $T\}$.

In order to develop a pumping lemma for structures with automatic reachability relation, we will use the pumping lemma for regular languages.

▶ **Lemma 4** (see [4]). *Let $\mathcal{A} = (\Sigma, Q, \Delta, q_I)$ be an automaton recognising the (tree-)language L, let $T \in L$ and let ρ be an accepting run of $\mathcal{A}$ on T. If $d \in dom(T)$ with $\mathrm{dp}(T_d) > |Q|$, then there are nodes $d \leq d_1 \leq d_2 \in dom(T)$ such that the following holds. If we replace in t the subtree rooted at d_1 by the subtree rooted at d_2, the tree T_0 resulting from this replacement satisfies $T_0 \in L$. Furthermore, let $T_1, T_2, T_3, \ldots$ be the infinite sequence of trees where $T_1 = T$ and T_{i+1} arises from T_i by replacing the subtree rooted at d_2 in T_i by the subtree rooted at d_1 in T_i, then $T_i \in L$ for all $i \in \mathbb{N}$. Furthermore, for each $i \in \mathbb{N}$ there is an accepting run ρ_i on T_i that coincides with ρ on all positions except for those in the subtree induced by d.*

In order to define the notion of an automatic reachability relation, we recall the definition of the *convolution* of two Σ-labelled trees T and T'. This is the tree

$$T \otimes T' : \mathrm{dom}(T) \cup \mathrm{dom}(T') \to (\Sigma \cup \{\Box\})^2$$

$$(T \otimes T')(d) := \begin{cases} (T(d), T'(d)) & \text{if } d \in \mathrm{dom}(T) \cap \mathrm{dom}(T') \\ (T(d), \Box) & \text{if } d \in \mathrm{dom}(T) \setminus \mathrm{dom}(T') \\ (\Box, T'(d)) & \text{if } d \in \mathrm{dom}(T') \setminus \mathrm{dom}(T) \end{cases}$$

where $\Box$ is a new symbol for padding. The convolution of trees is the standard concept if one wants to use a finite tree automaton for recognising some relation on trees. We say $\mathcal{A}$ recognises some relation R if $L(\mathcal{A}) := \{T_1 \otimes T_2 \otimes \cdots \otimes T_n : (T_1, T_2, \ldots, T_n) \in R\}$. In this case we say that R is automatic.

Using the regular pumping lemma, we obtain the following two pumping lemmas for structures with automatic reachability relation.

▶ **Lemma 5.** *Let $\mathfrak{G} = (D, \vdash)$ be some graph such that D is a regular set of trees over the alphabet Σ. Assume that the reachability relation R on $\mathfrak{G}$ is recognised by some tree-automaton with q states. If there starts a finite path p at $d \in D$ of length $(|\Sigma| + 1)^{2^{(\mathrm{dp}(d)+q)}}$ then there start infinitely many paths at d.*

Proof. If p visits some vertex $d' \in D$ twice then there is obviously an infinite path starting at d and passing d' infinitely many times.

Otherwise, p passes $(|\Sigma|+1)^{2^{(\mathrm{dp}(d)+q)}}$ many different trees. Due to Corollary 2, p passes some d' of depth $\mathrm{dp}(d') > \mathrm{dp}(d)+q$. Since $\mathrm{dp}(d') - \mathrm{dp}(d) > q$, we may apply the regular pumping lemma to $d \otimes d'$ in such a way that we obtain infinitely many trees $d_1, d_2, \ldots$ such that $(d, d_i) \in R$. This means that for each i there is a path from d to d_i. ◀

▶ **Lemma 6.** *Let $\mathfrak{G} = (D, \vdash)$ be some graph such that D is a regular set of trees over the alphabet Σ. Assume that the reachability relation R on $\mathfrak{G}$ is recognised by some tree-automaton with q states. If $|\{x \in D : (d,x) \in R\}| > (|\Sigma|+1)^{2^{(\mathrm{dp}(d)+q)}}$, then $\{x \in D : (d,x) \in R\}$ is an infinite set.*

Proof. Assume that $|\{x \in D : (d,x) \in R\}| > (|\Sigma|+1)^{2^{(\mathrm{dp}(d)+q)}}$ There must be some $d' \in D$ such that $(d,d') \in R$ and $\mathrm{dp}(d') > \mathrm{dp}(d)+q$. We conclude as in the previous lemma. ◀

In the rest of this paper, we develop an adaptation of these lemmas to collapsible pushdown graphs of level 2.

3 Collapsible Pushdown Systems and Graphs

In this section we define our notation of collapsible pushdown systems (of level 2). For a more detailed introduction, we refer the reader to [6] or [9]. Afterwards, we present those results from [8] that are relevant for the results of this paper.

3.1 Collapsible Pushdown Stacks of Level 2

First, we provide some terminology concerning stacks of (collapsible) higher-order pushdown systems. We write Σ^{*2} for $(\Sigma^*)^*$ and Σ^{+2} for $(\Sigma^+)^+$. We call an $s \in \Sigma^{*2}$ a 2-*word*.

Let us fix a 2-word $s \in \Sigma^{*2}$ which consists of an ordered list $w_1, w_2, \ldots, w_m \in \Sigma^*$. We separate the words of this list by colons writing $s = w_1 : w_2 : \ldots : w_m$. We write $\mathrm{wdt}(s) := m$ for the *width* of s and $\mathrm{hgt}(s) := \max\{|w_i| : 1 \leq i \leq m\}$ for the *height* of s. For a second word $s' = w'_1 : w'_2 : \ldots : w'_n \in \Sigma^{*2}$, we write $s : s'$ for the concatenation $w_1 : w_2 : \ldots : w_m : w'_1 : w'_2 : \ldots : w'_n$. If $w \in \Sigma^*$, we write $[w]$ for the 2-word that consists of a list of one word which is w.

A level 2 collapsible pushdown stack is a special element of $(\Sigma \times \{1,2\} \times \mathbb{N})^{+2}$ that is generated by certain stack operations from an initial stack. We introduce these in the following definitions. The natural numbers following the stack symbol represent a pointer: every element in a collapsible pushdown stack has a pointer to some substack and applying the collapse operation returns the substack to which the topmost symbol of the stack points. Here, the first number denotes the *collapse level.* If it is 1 the collapse pointer always points to the symbol below the topmost symbol and the collapse operations just removes the topmost symbol. The more interesting case is when the collapse level of the topmost symbol of the stack s is 2. Then the stack obtained by the collapse contains the first n words of s where n is the second number in the topmost element of s.

The initial level 1 stack is $\bot_1 := (\bot, 1, 0)$ and the initial level 2 stack is $\bot_2 := [\bot_1]$.

Let $k \in \{1,2\}$ and let $s = w_1 : w_2 : \ldots : w_n \in (\Sigma \times \{1,2\} \times \mathbb{N})^{+2}$ be a 2-word such that $w_n = a_1 a_2 \ldots a_m$ with $a_i \in \Sigma \times \{1,2\} \times \mathbb{N}$ for all $1 \leq i \leq m$.

- We define the *topmost $(k-1)$-word of s* as $\mathrm{top}_k(s) := \begin{cases} w_n & \text{if } k = 2 \\ a_m & \text{if } k = 1. \end{cases}$

- For $\mathrm{top}_1(s) = (\sigma, i, j) \in \Sigma \times \{1,2\} \times \mathbb{N}$, we define the *topmost symbol* $\mathrm{Sym}(s) := \sigma$, the *collapse-level of the topmost element* $\mathrm{CLvl}(s) := i$, and the *collapse-link of the topmost element* $\mathrm{CLnk}(s) := j$.

For s, w_n and k as before, $\sigma \in \Sigma \setminus \{\bot\}$, and $w'_n := a_1 \dots a_{m-1}$, we define the stack operations

$$\mathrm{pop}_k(s) := \begin{cases} w_1 : w_2 : \ldots : w_{n-1} & \text{if } k = 2, n \geq 2, \\ w_1 : w_2 : \ldots : w_{n-1} : w'_n & \text{if } k = 1, m \geq 2, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

$$\mathrm{clone}_2(s) := w_1 : w_2 : \ldots : w_{n-1} : w_n : w_n,$$

$$\mathrm{push}_{\sigma,k}(s) := \begin{cases} w_1 : w_2 : \ldots : w_n(\sigma, 2, n-1) & \text{if k=2,} \\ w_1 : w_2 : \ldots : w_n(\sigma, 1, m) & \text{if k=1,} \end{cases}$$

$$\mathrm{collapse}(s) := \begin{cases} w_1 : w_2 : \ldots : w_{\mathrm{CLnk}(s)} & \text{if } \mathrm{CLvl}(s) = 2, n \geq \mathrm{CLnk}(s) > 0, \\ \mathrm{pop}_1(s) & \text{if } \mathrm{CLvl}(s) = 1, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The *set of level* 2*-operations* is $\mathrm{OP} := \{\mathrm{push}_{\sigma,1}, \mathrm{push}_{\sigma,2}, \mathrm{clone}_2, \mathrm{pop}_1, \mathrm{pop}_2, \mathrm{collapse}\}$. The *set of level* 2 *stacks*, $\mathrm{Stck}(\Sigma)$, is the smallest set that contains $\bot_2$ and is closed under all operations from OP.

Note that collapse- and pop_k-operations are only allowed if the resulting stack is a nonempty list of nonempty words. This avoids the special treatment of empty words or stacks. Furthermore, a collapse on level 2 summarises a non-empty sequence of pop_2-operations. For example, starting from $\bot_2$, we can apply a clone_2, a $\mathrm{push}_{\sigma,2}$, a clone_2, and finally a collapse. This sequence first creates a level 2 stack that contains 3 words and then performs the collapse and ends in the initial stack again. This example shows that clone_2-operations are responsible for the fact that collapse-operations on level 2 may remove more than one word from the stack. Since there is no level 1 clone operation, a collapse of level 1 always simulates exactly one pop_1.

For $s, s' \in \mathrm{Stck}(\Sigma)$, we call s' a *substack* of s if there are $n_1, n_2 \in \mathbb{N}$ such that $s' = {\mathrm{pop}_1}^{n_1}({\mathrm{pop}_2}^{n_2}(s))$. We write $s' \leq s$ if s' is a substack of s.

3.2 Collapsible Pushdown Systems and Collapsible Pushdown Graphs of Level 2

We introduce collapsible pushdown systems (CPS) and graphs (CPG) which are analogues of pushdown systems and pushdown graphs using collapsible pushdown stacks instead of ordinary stacks.

▶ **Definition 7.** A *collapsible pushdown system* is a tuple $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$ where Σ is a finite stack alphabet with a special symbol $\bot \in \Sigma$, Q a finite set of states, $q_0 \in Q$ the initial state, and $\Delta \subseteq Q \times \Sigma \times Q \times \mathrm{OP}$ the transition relation.

For $q \in Q$ and $s \in \mathrm{Stck}(\Sigma)$ the pair (q, s) is called a *configuration.* We define labelled *transitions* on pairs of configurations by setting $(q_1, s) \vdash^\delta (q_2, t)$ if there is a $\delta = (q_1, \sigma, q_2, op) \in \Delta$ such that $\mathrm{Sym}(s) = \sigma$ and $op(s) = t$. The union of these relations is denoted by $\vdash := \bigcup_{\delta \in \Delta} \vdash^\delta$. We set $C(S)$ to be the set of all configurations that are reachable from $(q_0, \bot_2)$ via $\vdash$-paths. We call $C(S)$ the set of *reachable* configurations. The *collapsible pushdown graph generated by* S is $\mathrm{CPG}(S) := \big(C(S), C(S)^2 \cap \vdash\big)$

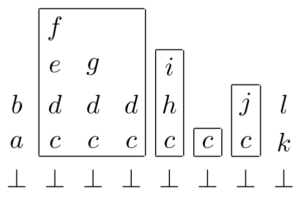

Figure 1 Example of blocks in a stack. These form a c-blockline.

▶ **Definition 8.** Let S be a CPS. A *run* ρ of S of length n is a function

$$\rho : \{0, 1, 2, \dots, n\} \to Q \times (\Sigma \times \{1, 2\} \times \mathbb{N})^{*2} \text{ such that } \rho(0) \vdash \rho(1) \vdash \cdots \vdash \rho(n).$$

We write $\mathrm{len}(\rho) := n$ and call ρ a run from $\rho(0)$ to $\rho(n)$. We say ρ visits a stack s at i if $\rho(i) = (q, s)$ for some $q \in Q$.

For runs ρ, π of length n and m, respectively, with $\rho(n) = \pi(0)$, we define the composition $\rho \circ \pi$ of ρ and π in the obvious manner.

▶ Remark. Note that we do not require runs to start in the initial configuration.

From now on, we consider a fixed set of states Q and a fixed stack alphabet Σ with bottom-of-stack symbol $\bot$.

3.3 Encoding of Configurations as Trees

In [8] we proved that collapsible pushdown graphs are tree-automatic via an encoding function CEnc. We recall this function in this section. The concept underlying the encoding is that of blocks and blocklines. A blockline is a list of words that start with the same letter and a block is a list of words that start with the same two letters. We encode a blockline as follows. The root is labelled by the first letter of all words; for each block of the blockline, we add one subtree encoding the corresponding block. We present the details after the formal introduction of blocks and blocklines. For $w \in \Sigma^*$ and $s = w_1 : w_2 : \dots : w_n \in \Sigma^{*2}$, we write $s' := w \setminus s$ for $s' = [ww_1] : [ww_2] : \dots : [ww_n]$.

▶ **Definition 9** (γ-block(line)). For Γ some set and $\gamma \in \Gamma$, we call $b \in \Gamma^{*2}$ a *γ-block* if $b = [\gamma]$ or $b = \gamma\tau \setminus s'$ for some $\tau \in \Gamma$ and $s' \in \Gamma^{*2}$. See Figure 1 for examples of blocks. If $b_1, b_2, \dots, b_n$ are γ-blocks, then we call $b_1 : b_2 : \dots : b_n$ a *γ-blockline*.

Note that every stack forms a $(\bot, 1, 0)$-blockline. Furthermore, every blockline l decomposes uniquely as $l = b_1 : b_2 : \dots : b_n$ of maximal blocks b_i. Another crucial observation is that a γ-block $b \in \Gamma^{*2} \setminus \Gamma$ decomposes as $b = \gamma \setminus l$ for some blockline l and we say l is the induced blockline of b. For $b \in \Gamma$ the induced blockline of $[b]$ is just the empty 2-word.

Now we encode a (σ, n, m)-blockline l in a tree by labelling the root with (σ, n), by encoding the blockline induced by the first block of l in the left subtree, and by encoding the rest of the blockline in the right subtree. In order to avoid repetitions, we do not repeat the symbol (σ, n) in the right subtree, but replace it by the default letter ε.

▶ **Definition 10.** Let $s = w_1 : w_2 : \dots : w_n \in (\Sigma \times \{1, 2\} \times \mathbb{N})^{+2}$ be a (σ, l, k)-blockline. Let w_i' be words such that $s = (\sigma, l, k) \setminus (w_1' : w_2' : \dots : w_n')$. Set $s' := w_1' : w_2' : \dots : w_n'$. As an abbreviation we write ${}_h s_i := w_h : w_{h+1} : \dots : w_i$. Furthermore, let $w_1 : w_2 : \dots : w_j$ be a maximal block of s. Note that $j > 1$ implies $w_{j'} = (\sigma, l, k)(\sigma', l', k')w''_{j'}$ for all $j' \leq j$, some

	$(c,2,1)$		$(e,1,3)$	
$(b,2,0)$	$(b,2,0)$	$(c,1,2)$	$(d,2,3)$	
$(a,2,0)$	$(a,2,0)$	$(a,2,2)$	$(a,2,2)$	$(a,2,2)$
$(\bot,1,0)$	$(\bot,1,0)$	$(\bot,1,0)$	$(\bot,1,0)$	$(\bot,1,0)$

$$\begin{tikzcd}
 & c,2 & & e,1 & \\
b,2 \arrow[r] & \varepsilon \arrow[u] & c,1 & d,2 \arrow[u] & \\
a,2 \arrow[u] & & a,2 \arrow[u] \arrow[r] & \varepsilon \arrow[u] \arrow[r] & \varepsilon \\
\bot,1 \arrow[u] \arrow[rr] & & \varepsilon \arrow[u] & &
\end{tikzcd}$$

Figure 2 A stack s and its encoding $\mathrm{Enc}(s)$: right arrows lead to 1-successors (right successors), upward arrows lead to 0-successors (left successors).

fixed $(\sigma', l', k') \in \Sigma \times \{1,2\} \times \mathbb{N}$, and appropriate $w''_{j'} \in \Sigma^*$. For $\rho \in \big(\Sigma \times \{1,2\}\big) \cup \{\varepsilon\}$, we define recursively the $\big(\Sigma \times \{1,2\}\big) \cup \{\varepsilon\}$-labelled tree $\mathrm{Enc}(s,\rho)$ via

$$\mathrm{Enc}(s,\rho) := \begin{cases} \rho & \text{if } |w_1| = 1, n = 1 \\ \rho(\emptyset, \mathrm{Enc}({}_2s_n, \varepsilon)) & \text{if } |w_1| = 1, n > 1 \\ \rho(\mathrm{Enc}({}_1s'_n, (\sigma', l')), \emptyset) & \text{if } j = n, |w_1| > 1 \\ \rho(\mathrm{Enc}({}_1s'_j, (\sigma', l')), \mathrm{Enc}({}_{j+1}s_n, \varepsilon)) & \text{otherwise.} \end{cases}$$

$\mathrm{Enc}(s) := \mathrm{Enc}(s, (\bot, 1))$ is called the (tree-)encoding of the stack $s \in \mathrm{Stck}(\Sigma)$.

Figure 2 shows a configuration and its encoding.

▶ Remark. In this encoding, the first block of a (σ, l, k)-blockline is encoded in a subtree whose root d is labelled (σ, l). For every node labelled by some element in $\Sigma \times \{1,2\}$, i.e., for every $d \in \mathrm{Enc}(s) \cap \{0,1\}^*0$, we can restore k from the position of d in $\mathrm{Enc}(s)$ as follows. If $l = 1$ then $k = |d|_0$, i.e., the number of occurrences of 0 in d. This is due to the fact that level 1 links always point to the preceding letter and that we always introduce a left-successor tree in order to encode letters that are higher in the stack. If $l = 2$ then

$$k = |\{d' \in \mathrm{dom}(\mathrm{Enc}(s)) \cap \{0,1\}^*1 : d' \leq_{\mathrm{lex}} d\}|,$$

where $\leq_{\mathrm{lex}}$ is the lexicographic order. This is due to the fact that every right-successor corresponds to the separation of some block from another block further left.

Having defined the encoding of stacks, we define the encoding of configurations.

▶ **Definition 11.** For $q \in Q$ and s some stack, we define $\mathrm{CEnc}(q,s) := q(\mathrm{Enc}(s), \emptyset)$.

The image of CEnc contains only trees of a very specific type. We call this class $\mathbb{T}_{\mathrm{Enc}}$.

▶ **Definition 12.** Let $\mathbb{T}_{\mathrm{Enc}}$ be the class of all trees T that satisfy the following conditions.

1. The root of T is labelled by some element of Q ($T(\varepsilon) \in Q$).
2. Every element of the form $\{0,1\}^*0$ is labelled by some $(\sigma, l) \in \Sigma \times \{1,2\}$; especially, $T(0) = (\bot, 1)$ and there are no other occurrences of $(\bot, 1)$ or $(\bot, 2)$.
3. Every element of the form $\{0,1\}^*1$ is labelled by ε.
4. $1 \notin \mathrm{dom}(T)$, $0 \in \mathrm{dom}(T)$.
5. For all $t \in T$, if $T(t0) = (\sigma, 1)$ then $T(t10) \neq (\sigma, 1)$.

▶ **Lemma 13** ([8]). *The image of* CEnc *is* $\mathbb{T}_{\mathrm{Enc}}$.

The following lemma shows that $\mathbb{T}_{\mathrm{Enc}}$ is a regular set.

▶ **Lemma 14.** *There is a finite automaton $\mathcal{A}_{\mathbb{T}_{\mathrm{Enc}}}$ with $f_0(\Sigma) := 2 + 3|\Sigma|$ many states that recognises $\mathbb{T}_{\mathrm{Enc}}$.*

Proof. Set $\mathcal{A}_{\mathbb{T}_{\mathrm{Enc}}} := (Q \cup (\Sigma \times \{1,2\}) \cup \{\varepsilon\}, Q_{\mathcal{A}}, \Delta_{\mathcal{A}}, q_I)$ where $Q_{\mathcal{A}}$ and $\Delta_{\mathcal{A}}$ are defined as follows. Let $Q_{\mathcal{A}} := \{\bot, q_I\} \cup (\Sigma \times \{1,2\}) \cup \{P_\sigma : \sigma \in \Sigma\}$. The states of the form (σ, i) are used to guess that a node of the tree is labelled by (σ, i) while the states P_σ are used to prohibit that the left successor of a certain node is labelled by $(\sigma, 1)$. The transitions ensure that whenever we guess that $d0$ is labelled by $(\sigma, 1)$ then $d1$ is reached in state P_σ ensuring that $d10$ cannot be labelled by $(\sigma, 1)$. $\Delta_{\mathcal{A}}$ is defined as follows.

- $(q_I, q, (\bot, 1), \bot) \in \Delta_{\mathcal{A}}$ for all $q \in Q$,
- $((\sigma, i), (\sigma, i), (\tau, 1), P_\tau) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$, $\tau \in \Sigma \setminus \{\bot\}, i \in \{1, 2\}$,
- $((\sigma, i), (\sigma, i), (\tau, 2), P_\bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$, $\tau \in \Sigma \setminus \{\bot\}$, and $i \in \{1, 2\}$,
- $((\sigma, i), (\sigma, i), (\tau, j), \bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$, $\tau \in \Sigma \setminus \{\bot\}, i, j \in \{1, 2\}$,
- $((\sigma, i), (\sigma, i), \bot, P_\bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$, and $i \in \{1, 2\}$,
- $((\sigma, i), (\sigma, i), \bot, \bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$, and $i \in \{1, 2\}$,
- $(P_\sigma, \varepsilon, (\tau, 1), P_\tau) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$ and $\tau \in \Sigma \setminus \{\sigma, \bot\}$
- $(P_\sigma, \varepsilon, (\tau, 2), P_\bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$ and $\tau \in \Sigma \setminus \{\bot\}$
- $(P_\sigma, \varepsilon, (\tau, i), \bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$ and $(\tau, i) \in (\Sigma \times \{1, 2\}) \setminus \{(\sigma, 1), (\bot, 1), (\bot, 2)\}$,
- $(P_\sigma, \varepsilon, \bot, P_\bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$
- $(P_\sigma, \varepsilon, \bot, \bot) \in \Delta_{\mathcal{A}}$ for all $\sigma \in \Sigma$ ◀

The next lemma is a straightforward observation concerning the depth of the encoding of a configuration in terms of the width and height of the stack.

▶ **Lemma 15.** *Given a stack s and a state q, such that (q, s) is reachable from the initial configuration by some path of length l then* $\mathrm{dp}(\mathrm{CEnc}(q, s)) < \mathrm{hgt}(s) + \mathrm{wdt}(s) \leq l + 2$.

Proof. We have seen that successors to the right correspond to the separation of different words of s. More precisely, $\mathrm{wdt}(s) = |\{d \in \{0,1\}^*\{1\} : d \in \mathrm{CEnc}(q, s)\}| + 1$. Furthermore, we have seen that every element $d \in \mathrm{CEnc}(q, s)$ encodes some word of length $|d|_0$. Thus, $\mathrm{hgt}(s) = \max\{|d|_0 : d \in \mathrm{CEnc}(q, s)\}$. We immediately conclude that $|d| < \mathrm{hgt}(s) + \mathrm{wdt}(s)$ for all $d \in \mathrm{CEnc}(q, s)$.

The second part of the claim is proved by induction. Note that the initial configuration is encoded by a tree of depth 2. Any stack operation increases the width or height of the stack by at most 1 and no operation increases the height and the width at the same time. ◀

3.4 Milestones

We now recall the concept of milestones from [8]. The milestones of a stack s are those substacks that every run to s has to pass. Thus, the concept of a milestone forms an essential key to understanding our pumping arguments in the next section.

▶ **Definition 16** (Milestone). A substack s' of $s = w_1 : w_2 : \ldots : w_n$ is a *milestone* if $s' = w_1 : w_2 : \ldots : w_i : w'$ such that $0 \leq i < n$ and $w_i \sqcap w_{i+1} \leq w' \leq w_{i+1}$.[3] We denote by $\mathrm{MS}(s)$ the set of milestones of s.

▶ **Lemma 17** ([8]). *If s, t, m are stacks with $m \in \mathrm{MS}(t)$ but $m \not\leq s$, then every run from s to t visits m.*

[3] $\sqcap$ is the greatest common prefix operator.

▶ **Corollary 18.** *If ρ is a run from a stack s to a stack t, then ρ visits some stack $m \in \mathrm{MS}(t)$ with $\mathrm{dp}(m) \leq \mathrm{dp}(s) + 1$.*

Proof. ρ has to pass some common substack u of s and t. It is an easy exercise to show that pop_1 and pop_2-operations do not increase the depth of the encoding of a stack. If $u \in \mathrm{MS}(t)$ we are done. Otherwise, there is a minimal sequence of level 1 push operations that generate a milestone $m \in \mathrm{MS}(t)$ from u. For $c := \mathrm{wdt}(u)$, $\mathrm{top}_2(u) < w_{c-1} \sqcap w_c$ where w_i denotes the i-th word of t and $\mathrm{top}_2(m) = w_{c-1} \sqcap w_c$. Since w_{c-1} is also the $(c-1)$-st word of s, the height of m is bounded by $\mathrm{hgt}(s)$. It is straightforward to show that the encodings of m and u differ in exactly two nodes. There is some $d \in \mathrm{dom}(\mathrm{Enc}(u))$ such that $\mathrm{Enc}(m)$ is $\mathrm{Enc}(u)$ where we delete the node $d1$ and add some node $d01^{k_1}01^{k_2}\dots01^{k_l+1}$ such that $d01^{k_1}01^{k_2}\dots01^{k_l} \in \mathrm{dom}(\mathrm{Enc}(u))$. This operation increases the depth by at most 1. ◀

Milestones form an effectively regular set. This stems from the close correspondence between milestones of a stack s and the elements of $\mathrm{CEnc}(s)$ as follows.

▶ **Definition 19.** Let $c = (q, s)$ be some configuration. For $T := \mathrm{CEnc}(c)$ the encoding of c, let $d \in T \setminus \{\varepsilon\}$. Then the *left and downward closed tree induced by d* is $LT(d,T) := T{\restriction}_D$ where $D := \{d' \in T : d' \leq_{lex} d\} \setminus \{\varepsilon\}$. Then we denote by $\mathrm{LStck}(d,T)$ the unique stack s' such that $\mathrm{CEnc}(q, s') = LT(d,T)$. We call $\mathrm{LStck}(d,T)$ the *left stack induced by d.*

▶ Remark. $\mathrm{LStck}(d, \mathrm{CEnc}(q, s))$ is a substack of s for all $d \in \mathrm{dom}(\mathrm{Enc}(s))$. This observation follows from the fact that the left stack is induced by a lexicographically downward closed subset. In fact, $\mathrm{LStck}(d, \mathrm{Enc}(q, s))$ is a milestone of s. See [8] for more details.

▶ **Lemma 20.** *[8] The map given by $g : d \mapsto \mathrm{LStck}(d, \mathrm{CEnc}(q, s))$ is an order isomorphism between $(dom(\mathrm{CEnc}(q, s)) \setminus \{\varepsilon\}, \leq_{lex})$ and $(\mathrm{MS}(s), \leq)$.*

▶ **Lemma 21.** *There is an automaton $\mathcal{A}$ with 5 states such that for all configurations (q, s) and (q', m) the automaton $\mathcal{A}$ accepts $\mathrm{CEnc}(q', m) \otimes \mathrm{CEnc}(q, s)$ if and only if $m \in \mathrm{MS}(s)$.*

Proof. $\mathcal{A}$ has to check that $\mathrm{CEnc}(q', m)$ is a left and downward closed subtree of $\mathrm{CEnc}(q, s)$ (except for the label at the root). The states of $\mathcal{A}$ are $\{q_I, \bot, =, \neq, =^*\}$. The transition relation Δ consists of the following transitions:

1. $(q_I, (q_1, q_2), =, \bot)$ for $q_1, q_2 \in Q$
2. $(=, (X, X), =^*, =)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
3. $(=, (X, X), =, \bot)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
4. $(=, (X, X), \neq, \bot)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
5. $(=, (X, X), =, \neq)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
6. $(=, (X, X), \bot, =)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
7. $(=, (X, X), \bot, \neq)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
8. $(=, (X, X), \neq, \neq)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
9. $(=, (X, X), \bot, \bot)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
10. $(=^*, (X, X), =^*, =^*)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
11. $(=^*, (X, X), =^*, \bot)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
12. $(=^*, (X, X), \bot, =^*)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
13. $(=^*, (X, X), \bot, \bot)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
14. $(\neq, (\bot, X), \bot, \bot)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
15. $(\neq, (\bot, X), \neq, \bot)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
16. $(\neq, (\bot, X), \bot, \neq)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$,
17. $(\neq, (\bot, X), \neq, \neq)$ for $X \in \{\varepsilon\} \cup (\Sigma \times \{1, 2\})$.

◀

4 Pumping on Encodings of Configurations

The main result of [8] is that there is an automaton that accepts the encoding of a configuration if and only if this configuration is reachable from the initial one. This automaton guesses this run by labelling each node of the encoding with the initial and final state corresponding to the subrun starting at the corresponding milestone. In the following, we will use variants of this automaton in order to develop a pumping lemma on collapsible pushdown systems.

▶ **Theorem 22** ([8]). *For each collapsible pushdown system $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$, there is a finite tree automaton $\mathcal{A}$ with*

$$f_1(Q, \Sigma) := 2 \cdot (2^{|Q\times Q|})^2 \cdot |Q|^2 \cdot |\Sigma \times \{1, 2\}| \cdot |\Sigma|$$

many states that accepts a tree $\mathrm{CEnc}(q, s)$ *if and only if* $(q, s) \in \mathrm{CPG}(\mathcal{S})$*, i.e., if there is a run of $\mathcal{S}$ from the initial configuration to (q, s).*

▶ Remark. In [8], we did non state the explicit bound on the number of states. This bound is extracted as follows. $\mathcal{A}$ guesses at each node $d \in \mathrm{CEnc}(q, s)$ states q_1, q_2 such that there is a run from $(q_1, \mathrm{LStck}(d, \mathrm{CEnc}(q, s)))$ to some configuration (q_2, t) where the definition of t depends on whether d is in the rightmost branch.

Let t' be the block encoded in the subtree rooted at d. If d is in the rightmost path, then $t = \mathrm{LStck}(d, \mathrm{CEnc}(q, s)) \setminus t'$.[4] At all other positions $t = (\mathrm{LStck}(d, \mathrm{CEnc}(q, s)) \setminus t') : \mathrm{top}_2(\mathrm{LStck}(d, \mathrm{CEnc}(q, s)))$. Thus, we obtain a factor 2 for keeping track of the rightmost branch. Furthermore, in order to verify the guesses q_1, q_2 at each node d, $\mathcal{A}$ stores the values of $\mathrm{Sym}(\mathrm{LStck}(d, \mathrm{CEnc}(q, s)))$, $\mathrm{CLvl}(\mathrm{LStck}(d, \mathrm{CEnc}(q, s)))$, $\mathrm{Sym}(\mathrm{pop}_1(\mathrm{LStck}(d, \mathrm{CEnc}(q, s))))$ and the existence of *loops* and *returns* starting at $\mathrm{LStck}(d, \mathrm{CEnc}(q, s))$. A loop starting in a stack s is a run from (q_1, s) to (q_2, s) for $q_1, q_2 \in Q$ that does not visit substacks of $\mathrm{pop}_2(s)$ and a return is a run from (q_1, s) to $(q_2, \mathrm{pop}_2(s))$ that does not visit any substack of $\mathrm{pop}_2(s)$ before its final configuration. At each node d, the automaton $\mathcal{A}$ has to keep track of the sets

$$\{(q_1, q_2) : \exists \text{ a loop from } (q_1, t) \text{ to } (q_2, t)\} \text{ and}$$
$$\{(q_1, q_2) : \exists \text{ a return from } (q_1, t) \text{ to } (q_2, \mathrm{pop}_2(t))\}$$

where $t = \mathrm{LStck}(d, \mathrm{CEnc}(q, s))$.

▶ **Corollary 23.** *For each collapsible pushdown system $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$, there is a finite tree automaton $\mathcal{A}_\mathcal{S}$ such that $\mathcal{A}_\mathcal{S}$ accepts a tree T if and only if there is some configuration c such that $T = \mathrm{CEnc}(c)$ and c is contained in $\mathrm{CPG}(\mathcal{S})$. Moreover, $\mathcal{A}_\mathcal{S}$ has $f_2(\Sigma, Q) := f_0(\Sigma) \cdot f_1(Q, \Sigma)$ many states.*

Proof. $\mathcal{A}$ is the product of the automaton from Theorem 22 and that from Lemma 14. ◀

In fact, every run from the initial configuration to some configuration c induces an accepting run of $\mathcal{A}$ on $\mathrm{CEnc}(c)$. There is a close correspondence between the states of the run of $\mathcal{A}$ at positions $d \in \mathrm{CEnc}(c)$ and the states in which the run to c visits $\mathrm{LStck}(d, \mathrm{CEnc}(c))$. We state this correspondence in the following lemma.

▶ **Lemma 24** ([8]). *For each stack t and each state $q \in Q$ there is a state $q_{q,t}$ of $\mathcal{A}_\mathcal{S}$ such that the following holds. Given an arbitrary configuration $c = (q, s)$ such that $t \in \mathrm{MS}(s)$,*

[4] $s_1 \setminus s_2$ is an abbreviation for $\mathrm{pop}_2(s_1) : (\mathrm{top}_2(s_1) \setminus s_2)$.

there is a run ρ from the initial configuration to c that passes t in state q for the last time if and only if there is an accepting run $\rho_{\mathcal{A}_\mathcal{S}}$ of $\mathcal{A}_\mathcal{S}$ on $\mathrm{CEnc}(c)$ *such that $\rho_{\mathcal{A}_\mathcal{S}}(d) = q_{q,t}$ for d the unique node such that $t = \mathrm{LStck}(d, \mathrm{CEnc}(c))$.*

In analogy to the general pumping lemma 5, we now prove a specific pumping lemma for collapsible pushdown graphs. In order to obtain better bounds, we do not use the automaticity of the reachability predicate. Instead, we use reachability only restricted to pairs $(q_1, s_1), (q_2, s_2)$ where s_1 is a milestone of s_2.

▶ **Theorem 25.** *Let $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$ be a* CPS*. Let ρ_0 be a run from the initial configuration to some configuration c with $\mathrm{len}(\rho_0) = l$. If ρ is a run starting at c of length*

$$\mathrm{len}(\rho) > f_3(Q, \Sigma, l) := |Q| \cdot (2|\Sigma| + 1)^{2^{l+2+f_2(Q,\Sigma)}}$$

then there are infinitely many runs starting at c.

Proof. If there are $i < j \leq \mathrm{len}(\rho)$ such that $\rho(i) = \rho(j)$ then $\pi_i := \rho{\restriction}_{[0,i]} \circ (\rho{\restriction}_{[i,j]})^i \circ \rho{\restriction}_{[j,\mathrm{len}(\rho)]}$ is an infinite sequence of runs starting at c.

Otherwise, the run visits more configurations than there are configurations whose encoding has depth at most $l + 2 + f_2(Q, \Sigma)$. Thus, there is some $i \leq \mathrm{len}(\rho)$ such that

$$\mathrm{dp}(\mathrm{CEnc}(\rho(i))) > l + 2 + f_2(Q, \Sigma)$$

Set $(q, s) := \rho(i)$. Due to Lemma 15, $\mathrm{dp}(\mathrm{CEnc}(c)) < l + 2$. Due to Corollary 18, there is a maximal $0 \leq j < i$ such that $\rho(j) = (\hat{q}, m)$ for some milestone $m \in \mathrm{MS}(s)$ and some state $\hat{q} \in Q$ such that $\mathrm{dp}(\mathrm{CEnc}(\hat{q}, m)) \leq l + 2$. Since m is a milestone of s, there is some node $d_m \in \mathrm{CEnc}(q, s)$ such that $\mathrm{LStck}(d_m, \mathrm{CEnc}(q, s)) = m$. This implies that the left and downward closed subtree induced by d_m is $\mathrm{Enc}(m)$.

Using Lemma 24, $\hat{q}$ and m define a state $q_{\hat{q},m}$ of $\mathcal{A}$ such that there is an accepting run of $\mathcal{A}$ on $\mathrm{CEnc}(q, s)$ that labels d_m with $q_{\hat{q},m}$.

Note that $LT(d_m, \mathrm{CEnc}(q, s))$ is a tree of depth at most $l + 2$. Hence, $\mathrm{CEnc}(q, s)$ contain a subtree of depth greater than $f_2(Q, \Sigma)$ that does not intersect with $LT(d_m, \mathrm{CEnc}(q, s))$. Since $f_2(Q, \Sigma)$ is a bound on the number of states of $\mathcal{A}$, Lemma 4 gives an infinite set of configurations $(q, s_1), (q, s_2), \dots, (q, s_i), \dots$ that are accepted by $\mathcal{A}$. Since the pumping does not affect $LT(d_m, \mathrm{CEnc}(q, s))$, we have $LT(d_m, \mathrm{CEnc}(q, s)) = LT(d_m, \mathrm{CEnc}(q, s_1)) = \cdots = LT(d_m, \mathrm{CEnc}(q, s_i)) = \dots$ and the accepting run of $\mathcal{A}$ on $\mathrm{CEnc}(q, s_i)$ labels d by $q_{\hat{q},m}$. Using Lemma 24, we conclude that for each $1 \leq i$ there is a run π_i from the initial configuration to (q, s_i) passing $(\hat{q}, m)$ at position k_i. Recall that $\rho(j) = (\hat{q}, m)$. Thus, the composition $\rho{\restriction}_{[0,j]} \circ \pi_i{\restriction}_{[k_i,\mathrm{len}(\pi_i)]}$ is a run from c to (q, s_i). Hence, we constructed infinitely many runs starting at configuration c. ◀

5 ε-Contractions of Collapsible Pushdown Graphs of Level 2

In this section we lift the pumping lemma from the previous section to ε-contractions of collapsible pushdown systems. Let $\mathfrak{G}$ be a graph with labelled edges where the labels come from the set $\Gamma \cup \{\varepsilon\}$. The ε-contraction of $\mathfrak{G}$ is then the graph $\mathfrak{G}/\varepsilon$ that consists of the vertices of $\mathfrak{G}$ where v and v' are connected by a γ-labelled edge (for $\gamma \in \Gamma$) if there is a path from v to v' in $\mathfrak{G}$ such that all edges of this path are labelled by ε except for the last edge, which is labelled by γ. We denote by $\vdash^\gamma$ the relation induced by the γ-labelled edges in the ε-contraction. From now on, we consider the transitions of a collapsible pushdown system to be labelled with elements from some finite set $\Gamma \cup \{\varepsilon\}$.

We first prove a slight variation of Theorem 22. Then we show that in every finitely branching ε-contraction of some CPS the stack size of connected configurations cannot differ too much. Finally, we develop the analogue of Theorem 25 for ε-contractions of CPG.

▶ **Lemma 26.** *For each collapsible pushdown system $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$ and each subset $\Delta' \subseteq \Delta$, there is a finite tree automaton $\mathcal{A}^{\Delta'}$ with $f_4 := 2 \cdot f_1(Q, \Sigma)$ many states that accepts a tree $\mathrm{CEnc}(q', t) \otimes \mathrm{CEnc}(q, s)$ for $t \in \mathrm{MS}(s)$ if and only if there is a run ρ from the initial configuration to (q, s) passing t at position i for the last time such that $\rho(i) = (q', t)$ and $\rho{\restriction}_{[i,\mathrm{len}(\rho)]}$ only uses transitions from Δ'.*

Proof. The automaton nondeterministically guesses the rightmost path of $\mathrm{CEnc}(q', t)$. After this path, i.e. on $\mathrm{dom}(\mathrm{CEnc}(q, s)) \setminus \mathrm{dom}(\mathrm{CEnc}(q', t))$ and at the lexicographically greatest node of $\mathrm{dom}(\mathrm{CEnc}(q', t))$, it simulates the automaton $\mathcal{A}$ from Theorem 22 but with respect to the transition relation Δ'. Along the rightmost path of $\mathrm{CEnc}(q', t)$ (except for the maximal node of this path), it simulates $\mathcal{A}$ in guessing final states for the corresponding subtrees. But it keeps the initial state fixed to q'. Thus, the automaton looks for runs to (q, s) that only use transitions from Δ', but it is forced to pass t' in state q'. ◀

▶ **Corollary 27.** *For each collapsible pushdown system $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$ and each subset $\Delta' \subseteq \Delta$, there is a finite tree automaton $\mathcal{A}$ with*

$$f_5(Q, \Sigma) := 5 \cdot f_0(\Sigma) \cdot f_0(\Sigma) \cdot f_4(Q, \Sigma)$$

many states that accepts a tree T if and only if $T = \mathrm{CEnc}(q', t) \otimes \mathrm{CEnc}(q, s)$ for some configurations $(q', t), (q, s)$ such that $t \in \mathrm{MS}(s)$ and such that there is a run ρ from the initial configuration to (q, s) passing t at position i for the last time such that $\rho(i) = (q', t)$ and $\rho{\restriction}_{[i,\mathrm{len}(\rho)]}$ only uses transitions from Δ'.

Proof. T has to consists of two components, each one from $\mathbb{T}_{\mathrm{Enc}}$. Taking a product of two adaptations of the Automaton from Lemma 14 we can check that $T = \mathrm{CEnc}(q', t) \otimes \mathrm{CEnc}(q, s)$ for some configurations (q', t) and (q, s). Furthermore, taking the product with the automaton from Lemma 21, we can ensure that $t \in \mathrm{MS}(s)$. Finally, taking the product with the automaton from Lemma 26 yields the automaton $\mathcal{A}$. ◀

Completely analogously to the proof of Theorem 25 we now derive a bound of the size of stacks of configurations connected by an edge in a finitely branching ε-contraction of collapsible pushdown graphs.

▶ **Lemma 28.** *For each transition $\delta \in \Delta$ there is an automaton $\mathcal{A}_\delta$ with 10 states that accepts $\mathrm{CEnc}(q, s) \otimes \mathrm{CEnc}(q', s')$ if and only if $(q, s) \vdash^{\delta} (q', s')$.*

The proof of this lemma can be found in the long version of this article. It is obtained by explicit construction of the automaton informally described in [8].

▶ **Corollary 29.** *For each collapsible pushdown system $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$ and each transition $\delta \in \Delta$, there is a finite tree automaton with $f_7(\Sigma) := 10 f_0(\Sigma) \cdot f_0(\Sigma)$ states that accepts a tree T, if and only if $T = \mathrm{CEnc}(q, s) \otimes \mathrm{CEnc}(q', s')$ and $(q, s) \vdash^{\delta} (q', s')$.*

Proof. T has to consists of two components, each one from $\mathbb{T}_{\mathrm{Enc}}$. Taking a product of two adaptations of the Automaton from Lemma 14 (one for each component) and of $\mathcal{A}_\delta$ from the previous lemma does the job. ◀

We now give a bound on the branching degree of ε-contractions of collapsible pushdown graphs.

▶ **Lemma 30.** *Let $\mathcal{S}$ be some collapsible pushdown system with stack alphabet Σ and state set Q. Set $\mathfrak{G} := \mathrm{CPG}(\mathcal{S})/\varepsilon$. If there are configurations $c', c \in \mathfrak{G}$ such that $\mathfrak{G} \models c' \vdash^{\gamma} c$ and*

$$\mathrm{dp}(\mathrm{CEnc}(c)) > 1 + \mathrm{dp}(\mathrm{CEnc}(c')) + f_7(\Sigma) \cdot f_5(Q, \Sigma)$$

then $\mathfrak{G}$ is infinitely branching.

Proof. Assume that $c' \vdash^{\gamma} c$ in $\mathfrak{G}$. Let $\Delta' \subseteq \Delta$ be the ε-labelled transitions of the CPS $\mathcal{S}$ generating $\mathfrak{G}$. Let $\mathcal{A}'$ denote the automaton of Corollary 27 with respect to $\mathcal{S}$ and Δ'. Furthermore, let $\mathcal{A}_{\gamma}$ denote the automaton of Corollary 29. There is an automaton $\hat{\mathcal{A}}$ that accepts a tree T if and only if $T = \mathrm{CEnc}(q_1, s_1) \otimes \mathrm{CEnc}(q_2, s_2) \otimes \mathrm{CEnc}(q_3, s_3)$ such that

1. $s_1 \in \mathrm{MS}(s_2)$,
2. $(q_1, s_1) \in \mathrm{CPG}(\mathcal{S})$,
3. there is a run from (q_1, s_1) to (q_2, s_2) that only uses transitions from Δ', and
4. $(q_2, s_2) \vdash^{\gamma} (q_3, s_3)$.

$\mathcal{A}$ is basically a product of $\mathcal{A}'$ and $\mathcal{A}_{\gamma}$. Thus, this automaton can be realised with $f_7(\Sigma) \cdot f_5(Q, \Sigma)$ many states.

Fix a run ρ witnessing that $c' \vdash^{\gamma} c$ in $\mathfrak{G}$. Writing $(q, s) := c$ and $\hat{c} := \rho(\mathrm{len}(\rho) - 1)$, Corollary 18 gives us a milestone m such that for some $q_m \in Q$ the automaton $\mathcal{A}$ accepts $\mathrm{CEnc}(q_m, m) \otimes \mathrm{CEnc}(\hat{c}) \otimes \mathrm{CEnc}(c)$. Furthermore,

$$\mathrm{dp}(\mathrm{CEnc}(c)) > \mathrm{dp}(\mathrm{CEnc}(q_m, m)) + f_7(\Sigma) \cdot f_5(Q, \Sigma).$$

Thus, we can apply the regular pumping argument to some subtree of $\mathrm{CEnc}(q_m, m) \otimes \mathrm{CEnc}(\hat{c}) \otimes \mathrm{CEnc}(c)$ where the first component is undefined. This yields infinitely many configurations $c_1, c_2, c_3, \dots$ such that $(q_m, m) \vdash^{\gamma} c_j$ for each $j \in \mathbb{N}$. Since $\rho{\restriction}_{[0,i]}$ is an ε-path from c' to (q_m, m), this implies $c' \vdash^{\gamma} c_j$ for each $j \in \mathbb{N}$ in $\mathfrak{G}$. Hence, $\vdash^{\gamma}$ in $\mathfrak{G}$ is infinitely branching at c'. ◀

▶ **Corollary 31.** *Let $\mathfrak{G}$ be the ε-contraction of some collapsible pushdown system with stack alphabet Σ and state set Q. If $\mathfrak{G}$ is finitely branching and if a path of length n connects the initial configuration with $c \in \mathfrak{G}$, then*

$$\mathrm{dp}(\mathrm{CEnc}(c)) \leq 2 + n\,(1 + f_7(\Sigma) \cdot f_5(Q, \Sigma))$$

The proof is by induction using the previous lemma. The straightforward adaptation of Theorem 25 yields the following pumping lemma for ε-contractions of collapsible pushdown graphs of level 2.

▶ **Theorem 32.** *Let $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$ be a CPS. Let $\mathfrak{G} := \mathrm{CPG}(\mathcal{S})/\varepsilon$ be finitely branching. Let ρ_0 be a path from the initial configuration to some configuration c of length l in $\mathfrak{G}$.*

If there is a path ρ starting in c such that

$$\mathrm{len}(\rho) > f_6(Q, \Sigma, l) := |Q| \cdot (2|\Sigma| + 1)^{2^{L+K}}$$

for $L := 2 + l\,(1 + f_7(\Sigma) \cdot f_5(Q, \Sigma))$

and for $K := 1 + f_7(\Sigma) \cdot f_5(Q, \Sigma)$

then there are infinitely many paths in $\mathfrak{G}$ starting at c.

Proof. There may be $i < j$ such that $\rho(i) = \rho(j)$ and we can iterate $\rho{\restriction}_{[i,j]}$ arbitrarily many times. Otherwise, due to the length of ρ, there is some i such that

$$\mathrm{dp}(\mathrm{CEnc}(\rho(i))) > L + K \geq \mathrm{dp}(\mathrm{CEnc}(\rho(0))) + K.$$

Analogous to the proof of Lemma 30, pumping yields runs $\rho_1, \rho_2, \dots$ starting at c and ending in pairwise different configurations of $\mathfrak{G}$. ◀

Using the same bounds, the second general pumping lemma 6 has a collapsible pushdown version:

▶ **Theorem 33.** *Let $\mathcal{S} = (\Sigma, Q, \Delta, q_0)$ be a* CPS*. Let $\mathfrak{G} := \mathrm{CPG}(\mathcal{S})/\varepsilon$ be finitely branching. Let ρ_0 be a path from the initial configuration to some configuration c of length l in $\mathfrak{G}$.*

If there are more than $f_6(Q, \Sigma, l)$ many configurations reachable from c then there are infinitely many paths in $\mathfrak{G}$ starting at c.

Again, one of the configurations reachable form c must be encoded by a tree of depth $\mathrm{dp}(\mathrm{CEnc}(\rho(i))) > \mathrm{dp}(\mathrm{CEnc}(\rho(0))) + K$ and we may apply the pumping argument from the previous proof.

▶ Remark. There is some function f_8 such that $f_6(Q, \Sigma, l) \leq 2^{2^{f_8(Q,\Sigma)\cdot l}}$.

6 Applications

▶ **Corollary 34.** *Let $\mathcal{S}$ be some* CPS *and $\mathcal{G} := \mathrm{CPG}(\mathcal{S})/\varepsilon$. It is decidable whether $\mathfrak{G}$ is finite.*

We can also use the pumping lemma in order to prove that certain graphs are not ε-contractions of CPG.

▶ **Example 35.** Let $\varphi : \mathbb{N} \to \mathbb{N}$ be an unbounded monotone function. The tree

$$\mathcal{T} = \left\{0^{i-1}1^j \in \{0,1\}^* : j \leq 2^{2^{\varphi(i)\cdot i}} + 1\right\}$$

(with left and right successor relation) is not the ε-contraction of any CPG of level 2.

Heading for a contradiction, assume there was such a CPS $\mathcal{S}$. Choose $k \in \mathbb{N}$ such that $\varphi(k) \geq f_8(Q, \Sigma)$. Thus, $2^{2^{\varphi(k)\cdot k}} \geq f_6(Q, \Sigma, k)$ whence we may apply Theorem 32 to the path connecting $0^{k-1}1$ with $0^{k-1}1^{\varphi(k)k+1}$ and obtain infinitely many paths starting in $0^{k-1}1$. But this contradicts the definition of $\mathcal{T}$.

Using the second pumping lemma, one proves analogously that the tree

$$\mathcal{T} = \{0^n : n \in \mathbb{N}\} \cup \{0^n 1 : n \in \mathbb{N}\} \cup \left\{0^{i-1}1j : j \leq 2^{2^{\varphi(i)\cdot i}} + 1\right\}$$

is not the ε-contraction of any CPS of level 2.

7 Conclusions

To our knowledge, we presented the first pumping lemma for collapsible pushdown graphs of level 2. Moreover, the result also improves Hayashi's pumping lemma for indexed languages [7]. An analysis of his proof shows that his pumping lemma applies to runs of level 2 pushdown systems that have length three-fold exponential in the size of the pushdown system. Our lemma already applies to runs that have length doubly exponential in the size of the system.

Unfortunately, our approach does not extend directly to higher levels of the collapsible pushdown hierarchy. Higher levels are not tree-automatic. But perhaps it is possible to represent the reachability relations of higher-order collapsible pushdown graphs by other types of automata for which pumping lemmas exist. These could then be turned into pumping lemmas for higher-order collapsible pushdown graphs. Another approach towards pumping on higher-order graphs stems from the the technical tools of milestones and loops developed in [8]. It is an interesting question whether these notions can be adapted to higher levels in order to obtain pumping lemmas for all higher-order (collapsible) pushdown graphs.

Acknowledgments

I thank the anonymous referees for their very useful comments.

References

1 A. Blumensath. On the structure of graphs in the Caucal hierarchy. *Theoretical Computer Science*, 400:19–45, 2008.

2 C. H. Broadbent. Private communication. September 2010.

3 A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2003*, volume 2914 of *LNCS*, pages 112–123. Springer, 2003.

4 H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007. release October, 12th 2007.

5 Robert H. Gilman. A shrinking lemma for indexed languages. *Theor. Comput. Sci.*, 163(1&2):277–281, 1996.

6 M. Hague, A. S. Murawski, C-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In *LICS '08: Proceedings of the 2008 23rd Annual IEEE Symposium on Logic in Computer Science*, pages 452–461, 2008.

7 Takeshi Hayashi. On derivation trees of indexed grammars. *Publ. RIMS, Kyoto Univ.*, 9:61–92, 1973.

8 A. Kartzow. Collapsible pushdown graphs of level 2 are tree-automatic. In *STACS 10*, volume 5 of *LIPIcs*, pages 501–512. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

9 A. Kartzow. *First-Order Model Checking On Generalisations of Pushdown Graphs.* PhD thesis, Technische Universität Darmstadt, 2011. Unpublished, submitted in December 2010.

10 T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *FOSSACS'02*, volume 2303 of *LNCS*, pages 205–222. Springer, 2002.

11 Naoki Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. *SIGPLAN Not.*, 44:416–428, January 2009.

12 Pawel Parys. The pumping lemma is incorrect? unpublished, June 2010.

13 Pawel Parys. Collapse Operation Increases Expressive Power of Deterministic Higher Order Pushdown Automata. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 603–614, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.