COLLAPSIBLE PUSHDOWN GRAPHS OF LEVEL 2   ARE TREE-AUTOMATIC [*]

ALEXANDER KARTZOW

Universität Leipzig, Institut für Informatik, Augustusplatz 10, 04103 Leipzig, Germany
e-mail address: kartzow@informatik.uni-leipzig.de

Abstract. We show that graphs generated by collapsible pushdown systems of level 2 are tree-automatic. Even if we allow ε-contractions and reachability predicates (with regular constraints) for pairs of configurations, the structures remain tree-automatic whence their first-order logic theories are decidable. As a corollary we obtain the tree-automaticity of the second level of the Caucal-hierarchy.

Contents

1. Introduction

Higher-order pushdown systems were first introduced by Maslov [14, 15] as accepting devices for word languages. Later, Knapik et al. [13] studied them as generators for trees. They obtained an equi-expressivity result for higher-order pushdown systems and for higher-order recursion schemes that satisfy the constraint of *safety*, which is a rather unnatural syntactic condition. Hague et al. [10] introduced collapsible pushdown systems as extensions of higher-order pushdown systems and proved that these have exactly the same power as higher-order recursion schemes as methods for generating trees.

Both higher-order and collapsible pushdown systems also form interesting devices for generating graphs. Carayol and Wöhrle [6] showed that the graphs generated by higher-order pushdown systems of level l coincide with the graphs in the l-th level of the Caucal-hierarchy, a class of graphs introduced by Caucal [7]. Every level of this hierarchy is obtained from the preceding level by applying graph unfoldings and monadic second-order interpretations. Both operations preserve the decidability of the monadic second-order theory whence the Caucal-hierarchy forms a large class of graphs with decidable monadic second-order theories. If we use collapsible pushdown systems as generators for graphs we obtain a different situation. Hague et al. showed that even the second level of the hierarchy contains a graph with undecidable monadic second-order theory. Furthermore, they showed the decidability of the modal μ-calculus theories of all graphs in the hierarchy. These results turn graphs generated by collapsible pushdown systems into an interesting class. The author only knows one further natural class of graphs which shares these two properties, viz. the class of nested pushdown trees (cf. [1]). Moreover this class can be seen as a subclass of that of collapsible pushdown graphs (cf. [11]).

This paper is the long version of [12] and studies the first-order model-checking problem on collapsible pushdown graphs. We show that the graphs in the second level of the collapsible pushdown hierarchy are tree-automatic. Tree-automatic structures were introduced by Blumensath [2]. These structures enjoy decidable first-order theories due to the good closure properties of finite automata. Since the translation from collapsible pushdown systems into tree-automata presentations of the generated graphs is uniform, our result implies that first-order model-checking on collapsible pushdown graphs of level 2 is decidable:

given a pushdown system, first compute the tree-automata representing its graph, then apply classical model-checking for tree-automatic structures.

Moreover, the result still holds if regular reachability predicates are added to the graphs.

1.1. **Main Result.**

Theorem 1.1. *Let $\mathcal{S}$ be a collapsible pushdown system of level 2 with configuration graph $\mathfrak{G}$. Let $\mathfrak{G}/\varepsilon$ be the ε-contraction of $\mathfrak{G}$. Any expansion of $\mathfrak{G}/\varepsilon$ by regular reachability relations is tree-automatic.*[1]

A regular reachability relation is of the form Reach_L for some regular language L. For nodes a, b of some graph $\mathfrak{G}$ with labelled edges, $\mathfrak{G} \models \mathrm{Reach}_L(a, b)$ if there is a path from a to b which is labelled by some word $w \in L$. The translation from collapsible pushdown systems to tree-automatic presentations is uniform, i.e., there is a uniform way of computing, given a collapsible pushdown system (and finite automata representing regular languages over the edge-alphabet of the system), the tree-automata presentation of the ε-contraction of the generated graph (expanded by the regular reachability predicates). Once we have obtained tree-automata representing some graph, first-order model-checking on this graph is decidable. Combining these results we obtain the following corollary.

Corollary 1.2. *The following problem is decidable:*
Input: *a collapsible pushdown system $\mathcal{S}$ (of level 2), finite automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ representing regular languages $L_1, \ldots, L_n$, and a formula φ in first-order logic extended by the relations $\mathrm{Reach}_{L_1}, \ldots, \mathrm{Reach}_{L_n}$*
Output: *$\mathfrak{G}/\varepsilon \models \varphi$? ($\mathfrak{G}/\varepsilon$ denotes the ε-contraction of the graph generated by $\mathcal{S}$.)*

We also show that the decision procedure is necessarily nonelementary in the size of the formula.

1.2. **Outline of the Paper.**
Sections 2.1 and 2.2 introduce basic notation. In Section 2.3 we introduce collapsible pushdown graphs (of level 2) and we show basic properties of these graphs. We recall the notion of tree-automaticity in Section 2.4. We present our encoding of configurations of collapsible pushdown graphs as trees in Section 3. We also show that, once we have proved that regular reachability is tree-automatic via this encoding, collapsible pushdown graphs are tree-automatic. The final part of that Section discusses the optimality of the first-order model-checking algorithm obtained from this tree-automata approach. Sections 4 and 5 complete the proof by showing that regular reachability is actually tree-automatic via our encoding. In Section 4 we develop the technical machinery for proving the regularity of the reachability relation. We analyse how arbitrary runs of collapsible pushdown systems decompose as sequences of simpler runs. Afterwards, in Section 5 we apply these results and show that the encoding turns the regular reachability relations into tree-automatic relations. Finally, Section 6 contains concluding remarks.

2. Preliminaries and Basic Definitions

For a function f, we denote by $\mathrm{dom}(f)$ its domain.

---

[1] Due to Broadbent et al. [4], the result still holds if we expand the graph by $L\mu$ definable predicates.

2.1. **Logics.** We denote by FO first-order logic. Given some graph $\mathfrak{G} = (V, E_1, E_2, \ldots, E_n)$ with labelled edge relations $E_1, \ldots, E_n \subseteq V \times V$ we denote by Reach the *reachability predicate*, defined by

$$\text{Reach} := \left\{ (a, b) : \text{ there is a } (\bigcup_{i=1}^{n} E_i)\text{-path from } a \text{ to } b \right\}.$$

Given some regular language L, we denote by

$$\text{Reach}_L := \left\{ (a, b) : \text{ there is a } (\bigcup_{i=1}^{n} E_i)\text{-path from } a \text{ to } b \text{ labelled by some word } w \in L \right\}$$

the *reachability predicate with respect to L*.

2.2. **Words and Trees.** For words $w_1, w_2 \in \Sigma^*$ we write $w_1 \leq w_2$ if w_1 is a prefix of w_2. $w_1 \sqcap w_2$ denotes the greatest common prefix of w_1 and w_2. The concatenation of w_1 and w_2 is denoted by $w_1 w_2$.

We call a finite set $D \subseteq \{0, 1\}^*$ a *tree domain*, if D is prefix closed. A Σ-*labelled tree* is a mapping $T : D \to \Sigma$ for D some tree domain. For $d \in D$ we denote the *subtree rooted at* d by T_d. This is the tree defined by $T_d(e) := T(de)$. We will usually write $d \in T$ instead of $d \in \text{dom}(T)$. We denote the *depth* of the tree T by $\text{dpt}(T) := \max\{|t| : t \in \text{dom}(T)\}$.

For T some tree with domain D, let D_+ denote the set of minimal elements of the complement of D, i.e.,

$$D_+ = \{e \in \{0, 1\}^* \setminus D : \text{ all proper ancestors of } e \text{ are contained in } D\}.$$

We write $D^\oplus$ for $D \cup D_+$. Note that $D^\oplus$ is the extension of the tree domain D by one layer.

Sometimes it is useful to define trees inductively by describing the subtrees rooted at 0 and 1. For this purpose we fix the following notation. Let $\hat{T}_0$ and $\hat{T}_1$ be Σ-labelled trees and $\sigma \in \Sigma$. Then we write $T := \sigma \left\langle \hat{T}_0; \hat{T}_1 \right\rangle$ for the Σ-labelled tree T with the following three properties

1. $T(\varepsilon) = \sigma$,   2. $T_0 = \hat{T}_0$, and   3. $T_1 = \hat{T}_1$.

We denote by $\mathbb{T}_\Sigma$ the set of all Σ-labelled trees.

2.3. **Collapsible Pushdown Graphs.** Before we introduce collapsible pushdown graphs (CPG) in detail, we fix some notation. Then we informally explain collapsible pushdown systems. Afterwards we formally introduce these systems and the graphs generated by them. We conclude this section with some basic results on runs of collapsible pushdown systems.

We set $\Sigma^{+2} := (\Sigma^+)^+$ and $\Sigma^{*2} := (\Sigma^*)^*$. Each element of Σ^{*2} is called a 2-word. Stacks of a collapsible pushdown system are certain 2-words from Σ^{+2} over a special alphabet. In analogy, we will call words also 1-words.

Let us fix a 2-word $s \in \Sigma^{*2}$. s consists of an ordered list $w_1, w_2, \ldots, w_m$ of words. If we want to state this list of words explicitly, we write $s = w_1 : w_2 : \cdots : w_m$. Let $|s| := m$ denote the *width* of s. The *height* of s is $\text{hgt}(s) := \max\{|w_i| : 1 \leq i \leq m\}$ which is the length of the longest word occurring in s.

Let s' be another 2-word with $s' = w_1' : w_2' : \cdots : w_l'$. We write $s : s'$ for the concatenation $w_1 : w_2 : \cdots : w_m : w_1' : w_2' : \cdots : w_l'$.

For some word w, let $[w]$ be the 2-word that only consists of w. We regularly omit the brackets if no confusion arises.

A level 2 stack s over some alphabet Σ is a 2-word over Σ where each letter additionally carries a link to some i-word for $1 \leq i \leq 2$. We call i the level of the link. The idea is that the linked i-word contains some information about what the stack looked like when the letter was created.

We first define the initial level 2 stack; afterwards we describe the stack operations that are used to generate all level 2 stacks from the initial one.

Definition 2.1. Let Σ be some finite alphabet with a distinguished bottom-of-stack symbol $\bot \in \Sigma$. The *initial stack* of level 1 is the word $\bot_1 := \bot$. The initial stack of level 2 is the 2-word $\bot_2 := [\bot_1]$.

We informally describe the stack operations that can be applied to a level 2 stack.

- The push operation of level 1, denoted by $\text{push}_{\sigma,k}$ for $\sigma \in \Sigma$ and $1 \leq k \leq 2$, writes the symbol σ onto the topmost level 1 stack and attaches a link of level k. This link points to a copy of the topmost k-word of the resulting stack without the topmost $k-1$ stack. For $k = 2$ this means that the link points to the current stack where the topmost word is removed. For $k = 1$ the link points to the topmost word of the stack before the push operation was performed.
- The push operation of level 2 is denoted by clone_2. It duplicates the topmost word. Since this also copies the values of the links stored in the topmost word, the copy of each symbol in the newly created word still contains information what did the stack look like when the corresponding original symbol was pushed onto the stack.
- The level i pop operation pop_i for $1 \leq i \leq 2$ removes the topmost entry of the topmost i-word.
- The last operation is collapse. The result of collapse is determined by the link attached to the topmost letter of the stack. If we apply collapse to a stack s where the link level of the topmost letter is i, then collapse replaces the topmost level i stack of s by the level i stack to which the link points. Due to how push operations create these links, the application of a collapse is equivalent to the application of a sequence of pop_i operations where the link of the topmost letter controls how long this sequence is.

In the following, we formally introduce collapsible pushdown stacks and the stack operations. We represent such a stack of letters with links as 2-words over the alphabet $(\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$. We consider elements from Σ as elements with a link of level 1 and elements $(\sigma, 2, k)$ as letters with a link of level 2. In the latter case, the third component specifies the width of the substack to which the link points. For letters with link of level 1, the position of this letter within the stack already determines the stack to which the link points. Thus, we need not explicitly specify the link in this case.

Remark 2.2. Other equivalent definitions, for instance in [10], use a different way of storing the links: they also store symbols (σ, i, n) on the stack, but here n denotes the number of pop_i transitions that are equivalent to performing the collapse operation at a stack with topmost element (σ, i, n). The disadvantage of that approach is that the clone_i operation cannot copy stacks. Instead, it can only copy the symbols stored in the topmost stack and has to alter the links in the new copy. A clone of level i must replace all links (σ, i, n) by $(\sigma, i, n + 1)$ in order to preserve the links stored in the stack.

We introduce some auxiliary functions which are useful for defining the stack operations.

Definition 2.3. For $s = w_1 : w_2 : \cdots : w_n \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$, and $w_n = a_1 a_2 \ldots a_m$ we define the following auxiliary functions:

- $\text{top}_2(s) := w_n$ and $\text{top}_1(s) := a_m$.
- The *topmost symbol* is $\text{Sym}(s) := \sigma$ for $\text{top}_1(s) = \sigma \in \Sigma$ or $\text{top}_1(s) = (\sigma, 2, k) \in \Sigma \times \{2\} \times \mathbb{N}$.
- The *collapse level of the topmost element* is $\text{CLvl}(s) := \begin{cases} 1 & \text{if } \text{top}_1(s) \in \Sigma, \\ 2 & \text{otherwise.} \end{cases}$
- Set $\text{CLnk}(s) := \begin{cases} j & \text{if } \text{top}_1(s) \in \Sigma \times \{2\} \times \{j\}, \\ m - 1 & \text{if } \text{top}_1(s) \in \Sigma. \end{cases}$

  $\text{CLnk}(s)$ is called the *collapse link of the topmost element.*
- Set $\text{p}_{\sigma,2,k}(w_m) := w_m(\sigma, 2, k)$ and $\text{p}_{\sigma,2,k}(s) := w_1 : w_2 : \cdots : w_{n-1} : \text{p}_{\sigma,2,k}(w_n)$ for all $k \in \mathbb{N}$.

These auxiliary function are useful for the formalisation of the stack operations.

Definition 2.4. For $s = w_1 : w_2 : \cdots : w_n \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$ and $w_n = a_1 a_2 \ldots a_m$, for $\sigma \in \Sigma \setminus \{\bot\}$ and for $1 \le k \le 2$, we define the stack operations

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

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

$$\text{pop}_k(s) := \begin{cases} w_1 : w_2 : \cdots : w_{n-1} & \text{if } k = 2, n > 1, \\ w_1 : w_2 : \cdots : w_{n-1} : [a_1 a_2 \ldots a_{m-1}] & \text{if } k = 1, m > 1, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

$$\text{collapse}(s) := \begin{cases} w_1 : w_2 : \cdots : w_k & \text{if } \text{CLvl}(s) = 2, \text{CLnk}(s) = k > 0, \\ \text{pop}_1(s) & \text{if } \text{CLvl}(s) = 1, m > 1, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The *set of level 2 operations* is

$$\text{OP} := \{(\text{push}_{\sigma,k})_{\sigma \in \Sigma, 1 \le k \le 2}, \text{clone}_2, (\text{pop}_k)_{1 \le k \le 2}, \text{collapse}\}.$$

The *set of (level 2) stacks* $\text{Stck}(\Sigma)$ is the smallest set that contains $\bot_2$ and is closed under application of operations from OP.

Note that we defined collapse and pop_k in such a way that the the resulting stack is always nonempty and does not contain empty words. This avoids the special treatment of empty stacks. Note that there is no clone_1 operation. Thus, any collapse that works on level 1 is equivalent to one pop_1 operation because level 1 links always point to the preceding letter. Every collapse that works on level 2 is equivalent to a sequence of pop_2 operations. Next, we introduce the substack relation.

Definition 2.5. Let $s, s' \in \text{Stck}(\Sigma)$. We say that s' is a *substack* of s (denoted as $s' \le s$) if there are $n, m \in \mathbb{N}$ such that $s' = \text{pop}_1{}^n(\text{pop}_2{}^m(s))$.

Having concluded the definitions concerning stacks and stack operations, it is time to introduce collapsible pushdown systems.

Definition 2.6. A *collapsible pushdown system* of level 2 (CPS) is a tuple

$$\mathcal{S} = (Q, \Sigma, \Gamma, \Delta, q_0)$$

where Q is a finite set of states, Σ a finite stack alphabet with a distinguished bottom-of-stack symbol $\bot \in \Sigma$, Γ a finite input alphabet, $q_0 \in Q$ the initial state, and

$$\Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \mathrm{OP}$$

the transition relation.

Every $(q, s) \in \mathrm{Cnf}(Q, \Sigma) := Q \times \mathrm{Stck}(\Sigma)$ is called a *configuration* and $\mathrm{Cnf}(Q, \Sigma)$ is called the *set of configurations*.[2] We define γ-labelled *transitions* $\vdash^\gamma \subseteq \mathrm{Cnf} \times \mathrm{Cnf}$ as follows: $(q_1, s) \vdash^\gamma (q_2, t)$ if there is a $(q_1, \sigma, \gamma, q_2, op) \in \Delta$ such that $op(s) = t$ and $\mathrm{Sym}(s) = \sigma$.

We call $\vdash := \bigcup_{\gamma \in \Gamma} \vdash^\gamma$ the *transition relation* of $\mathcal{S}$. We set $C(\mathcal{S})$ to be the set of all configurations that are reachable from $(q_0, \bot_2)$ via $\vdash$. These configurations are called *reachable*. The *collapsible pushdown graph* (CPG) *generated by* $\mathcal{S}$ is

$$\mathrm{CPG}(\mathcal{S}) := \big(C(\mathcal{S}), (C(\mathcal{S})^2 \cap \vdash^\gamma)_{\gamma \in \Gamma}\big).$$

Example 2.7. The following example (Figure 1) of a collapsible pushdown graph $\mathfrak{G}$ of level 2 is taken from [10]. Let $Q := \{0, 1, 2\}, \Sigma := \{\bot, a\}, \Gamma := \{\mathrm{Cl}, A, A', P, \mathrm{Co}\}$. Δ is given by $(0, -, \mathrm{Cl}, 1, \mathrm{clone}_2), (1, -, A, 0, \mathrm{push}_{a,2}), (1, -, A', 2, \mathrm{push}_{a,2}), (2, a, P, 2, \mathrm{pop}_1)$, and $(2, a, \mathrm{Co}, 0, \mathrm{collapse})$, where $-$ denotes any letter from Σ.
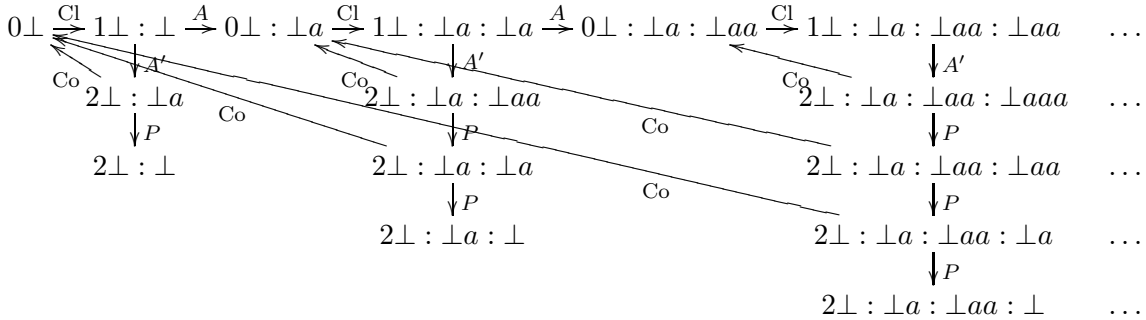
Figure 1: Example of the 2-CPG $\mathfrak{G}$ (the level 2 links of the letters a are omitted in the representation of the stacks).

Hague et al. [10] already noted that the previous example has undecidable MSO theory because the half grid $\{(n, m) \in \mathbb{N}^2 : n > m\}$ is MSO interpretable in this graph (note that the collapse edges of two vertices point to the same target if and only if the vertices are on the same diagonal of the grid).

Next we define the ε-contraction of a given collapsible pushdown graph. From now on, we always assume that the input alphabet Γ contains the symbol ε.

Definition 2.8. Let Γ be some alphabet. Let L and L_γ be the regular languages defined by the expressions $(\{\varepsilon\}^*(\Gamma \setminus \{\varepsilon\}))^*$ and $\{\varepsilon\}^*\gamma$, respectively. Given a collapsible pushdown graph $\mathfrak{G}$, the ε-contraction $\mathfrak{G}/\varepsilon$ of $\mathfrak{G}$ is the graph $(M, (\mathrm{Reach}_{L_\gamma})_{\gamma \in \Gamma \setminus \varepsilon})$ where $M := \{g \in \mathfrak{G} : \mathfrak{G} \models \mathrm{Reach}_L((q_0, \bot_2), g)\}$.

---

[2]We write Cnf instead of $\mathrm{Cnf}(Q, \Sigma)$ if Q and Σ are clear from the context.

Remark 2.9. This is the usual definition of ε-contraction. An edge in the new graph consists of a sequence of ε-edges followed by one non-ε-edge. The set of configurations is then restricted to those configurations that are reachable via the new edges from the initial configuration.

Now we come to the notion of a run of a collapsible pushdown system.

Definition 2.10. Let $\mathcal{S}$ be a collapsible pushdown system. A *run* ρ of $\mathcal{S}$ is a sequence of configurations that are connected by transitions, i.e., a sequence

$$c_0 \vdash^{\gamma_1} c_1 \vdash^{\gamma_2} c_2 \vdash^{\gamma_3} \cdots \vdash^{\gamma_n} c_n.$$

We denote by $\rho(i) := c_i$ the i-th configuration of ρ. Moreover, we denote by $\mathrm{length}(\rho) := n$ the *length* of ρ. For $0 \leq i \leq j \leq \mathrm{length}(\rho)$, we write $\rho\!\restriction_{[i,j]}$ for the subrun

$$c_i \vdash^{\gamma_{i+1}} c_{i+1} \vdash^{\gamma_{i+2}} \ldots \vdash^{\gamma_j} c_j.$$

We write $\mathrm{Runs}(c, c')$ for the set of runs starting at c and ending in c'. For s, s' stacks, we also write $\mathrm{Runs}(s, s') := \bigcup_{q,q' \in Q} \mathrm{Runs}((q, s), (q', s'))$.

Consider some configuration (q, s) of a CPS. If $|s| = n$ then a $\mathrm{push}_{\sigma,2}$ transition applied to (q, s) creates a letter with a link to the substack of width $n-1$. Thus, links to the substack of width $n - 1$ in some word above the n-th one are always created by a clone_2 operation. A direct consequence of this fact is the following lemma.

Lemma 2.11. *Let s be some level 2 stack with $\mathrm{top}_1(s) = (\sigma, 2, k)$. Let $\rho \in \mathrm{Runs}(s, s)$ be a run that passes $\mathrm{pop}_1(s)$. If $k < |s| - 1$ then ρ passes $\mathrm{pop}_2(s)$.*

The proof is left to the reader. Later we often use the contraposition: if $\mathrm{CLvl}(s) = 2$ and $\mathrm{CLnk}(s) < |s| - 1$, then any run not passing $\mathrm{pop}_2(s)$ does not pass $\mathrm{pop}_1(s)$.

Following the ideas of Blumensath [3] for higher-order pushdown systems, we introduce a prefix replacement for collapsible pushdown systems. This replacement allows to copy runs starting in one configuration into a run starting at another configuration.

Definition 2.12. For $t \in \mathrm{Stck}(\Sigma)$ and some substack $s \leq t$ we say that s is a *prefix* of t and write $s \trianglelefteq t$, if there are $n \leq m \in \mathbb{N}$ such that $s = w_1 : w_2 : \cdots : w_{n-1} : w_n$ and $t = w_1 : w_2 \cdots : w_{n-1} : v_n : v_{n+1} : \cdots : v_m$ such that $w_n \leq v_j$ for all $n \leq j \leq m$.

For a configuration $c = (q, t)$, we write $s \trianglelefteq c$ as an abbreviation for $s \trianglelefteq t$. For some run ρ, we write $s \trianglelefteq \rho$ if $s \trianglelefteq \rho(i)$ for all $i \in \mathrm{dom}(\rho)$.

Definition 2.13. Let s, t, u be level 2 stacks such that $s \trianglelefteq t$. Assume that

$$s = w_1 : w_2 : \cdots : w_{n-1} : w_n,$$
$$t = w_1 : w_2 \cdots : w_{n-1} : v_n : v_{n+1} : \cdots : v_m, \text{ and}$$
$$u = x_1 : x_2 : \cdots : x_{n-1} : x_n$$

for numbers $n, m \in \mathbb{N}$ such that $n \leq m$. For each $n \leq i \leq m$, let $\hat{v}_i$ be the unique word such that $v_i = w_n \hat{v}_i$. We define

$$t[s/u] := x_1 : x_2 : \cdots : x_{n-1} : (x_n \hat{v}_n) : (x_n \hat{v}_{n+1}) : \cdots : (x_n \hat{v}_m)$$

and call $t[s/u]$ *the stack obtained from t by replacing the prefix s by u*.

Remark 2.14. Note that for t some stack with level 2 links, the resulting object $t[s/u]$ may be no stack. Take for example the stacks

$$s = \bot(a, 2, 0) : \bot,$$
$$t = \bot(a, 2, 0) : \bot(a, 2, 0) \text{ and}$$
$$u = \bot : \bot.$$

Then $t[s/u] = \bot : \bot(a, 2, 0)$. This list of words cannot be created from the initial stack using the stack operation because an element $(a, 2, 0)$ in the second word has to be a clone of some element in the first one. But $(a, 2, 0)$ does not occur in the first word. Note that $t[s/u]$ is always a stack if $s = \text{pop}_2{}^k(t)$ for some $k \in \mathbb{N}$.

Lemma 2.15. *Let ρ be a run of some collapsible pushdown system $\mathcal{S}$ of level 2 and let s and u be stacks such that the following conditions are satisfied:*

(1) $s \trianglelefteq \rho$,
(2) $\text{top}_1(u) = \text{top}_1(s)$,
(3) $|s| = |u|$, *and*
(4) *for $\rho(0) = (q, t)$, $t[s/u]$ is a stack.*

Under these conditions the function $\rho[s/u]$ defined by $\rho[s/u](i) := \rho(i)[s/u]$ is a run of $\mathcal{S}$.

Proof (sketch). The proof is by induction on the length of ρ. It is tedious but straightforward to prove that $\rho(i)[s/u]$ and $\rho(i)$ share the same topmost element. Thus, the transition δ connecting $\rho(i)$ with $\rho(i+1)$ is also applicable to $\rho(i)[s/u]$. By case distinction on the stack operation one concludes that δ connects $\rho(i)[s/u]$ with $\rho(i+1)[s/u]$. $\qquad\square$

2.4. **Finite Automata and Automatic Structures.** In this section, we present the basic theory of finite bottom-up tree-automata and tree-automatic structures. For a more detailed introduction, we refer the reader to [8].

Definition 2.16. A *(finite tree-)automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_I, F, \Delta)$ where Q is a finite nonempty set of states, Σ is a finite alphabet, $q_I \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\Delta \subseteq Q \times \Sigma \times Q \times Q$ is the transition relation.

We next define the concept of a run of an automaton on a tree.

Definition 2.17. A *run* of $\mathcal{A}$ on a binary Σ-labelled tree t is a map $\rho : \text{dom}(t)^{\oplus} \to Q$ such that

• $\rho(d) = q_I$ for all $d \in \text{dom}(t)_+$, and
• $\big(\rho(d), t(d), \rho(d0), \rho(d1)\big) \in \Delta$ for all $d \in \text{dom}(t)$.

ρ is *accepting* if $\rho(\varepsilon) \in F$. We say t is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on t. With each automaton $\mathcal{A}$, we associate the *language*

$$L(\mathcal{A}) := \{t : t \text{ is accepted by } \mathcal{A}\}$$

recognised (or accepted) by $\mathcal{A}$. The class of languages accepted by automata is called the class of *regular* languages.

Automata can be used to represent infinite structures. Such representations have good computational behaviour. In the following we recall the definitions and important results on tree-automatic structures.

We first introduce the convolution of trees. This is a tool for representing an n-tuple of Σ-trees as a single tree over the alphabet $(\Sigma \cup \{\Box\})^n$ where $\Box$ is a padding symbol satisfying $\Box \notin \Sigma$.

Definition 2.18. The *convolution* of two Σ-labelled trees t and s is given by a function

$$t \otimes s : \mathrm{dom}(t) \cup \mathrm{dom}(s) \to (\Sigma \cup \{\Box\})^2$$

where $\Box$ is some new padding symbol, and

$$(t \otimes s)(d) := \begin{cases} (t(d), s(d)) & \text{if } d \in \mathrm{dom}(t) \cap \mathrm{dom}(s), \\ (t(d), \Box) & \text{if } d \in \mathrm{dom}(t) \setminus \mathrm{dom}(s), \\ (\Box, s(d)) & \text{if } d \in \mathrm{dom}(s) \setminus \mathrm{dom}(t). \end{cases}$$

We also use the notation $\bigotimes(t_1, t_2, \ldots, t_n)$ for $t_1 \otimes t_2 \otimes \cdots \otimes t_n$.

Using convolutions of trees we can use a single automaton for defining n-ary relations on a set of trees. Thus, we can then use automata to represent a set and a tuple of n-ary relations on this set. If we can represent the domain of some structure and all its relations by automata, we call the structure automatic.

Definition 2.19. We say a relation $R \subseteq \mathbb{T}_\Sigma{}^n$ is automatic if there is an automaton $\mathcal{A}$ such that $L(\mathcal{A}) = \{\bigotimes(t_1, t_2, \ldots, t_n) \in \mathbb{T}_\Sigma{}^n : (t_1, t_2, \ldots, t_n) \in R\}$.

A structure $\mathfrak{B} = (B, E_1, E_2, \ldots, E_n)$ with relations E_i is *automatic* if there are automata $\mathcal{A}_B, \mathcal{A}_{E_1}, \mathcal{A}_{E_2}, \ldots, \mathcal{A}_{E_n}$ and a bijection $f : L(\mathcal{A}_B) \to B$ such that for $c_1, c_2, \ldots, c_n \in L(\mathcal{A}_B)$, the automaton $\mathcal{A}_{E_i}$ accepts $\bigotimes(c_1, c_2, \ldots, c_n)$ if and only if $(f(c_1), f(c_2), \ldots, f(c_n)) \in E_i$.

In other words, f is a bijection between $L(\mathcal{A}_B)$ and B and the automata $\mathcal{A}_{E_i}$ witness that the relations E_i are automatic via f. We call f a tree presentation of $\mathfrak{B}$.

Automatic structures form a nice class because automata theoretic techniques may be used to decide first-order formulas on these structures:

Theorem 2.20 ([2], [16], [11]). *If $\mathfrak{B}$ is automatic, then its* $\mathrm{FO}(\exists^{\mathrm{mod}}, \mathrm{Ram})$*-theory is decidable.*[3]

3. Collapsible Pushdown Graphs are Tree-Automatic

In Section 3.1 we present a bijection Enc between Cnf and a regular set of trees. Moreover, Enc translates the reachability predicates $\mathrm{Reach}_L \subseteq \mathrm{Cnf} \times \mathrm{Cnf}$ for each regular language L into a tree-automatic relation. The proof of this claim which is the technical core of this paper is developed in Sections 4 and 5. Before we present this proof, we show in Section 3.2 how this result can be used to prove our main theorem. Moreover, in Section 3.3 we discuss the optimality of the first-order model-checking algorithm derived from this construction.

Regularity of the regular reachability predicates implies that $\mathrm{Enc}{\restriction}_{\mathrm{dom}(\mathrm{CPG}(\mathcal{S})/\varepsilon)}$ is an automatic presentation of $\mathrm{CPG}(\mathcal{S})/\varepsilon$ because its domain and its transition relation can be defined as reachability relations Reach_L for certain regular languages L. Note that for the definition of the domain, we need the encoding of the initial configuration as parameter. This parameter can be hard-coded because its encoding is a fixed tree.

---

[3] $\mathrm{FO}(\exists^{\mathrm{mod}}, \mathrm{Ram})$ is the extension of FO by modulo counting quantifiers and by Ramsey-Quantifiers.
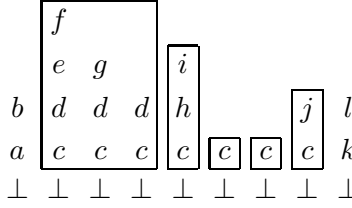
Figure 2: A stack with blocks forming a c-blockline.

3.1. **Encoding of Level 2 Stacks in Trees.** In this section we present an encoding of level 2 stacks in trees. The idea is to divide a stack into blocks and to encode different blocks in different subtrees. The crucial observation is that every stack is a list of words that share the same first letter. A block is a maximal list of words occurring in the stack which share the same two first letters. If we remove the first letter of every word of such a block, the resulting 2-word decomposes again as a list of blocks. Thus, we can inductively carry on to decompose parts of a stack into blocks and encode every block in a different subtree. The roots of these subtrees are labelled with the first letter of the block. This results in a tree where every initial left-closed path in the tree represents one word of the stack. A path of a tree is left-closed if its last element has no left successor (i.e., no 0-successor).

The following notation is useful for the formal definition of blocks. Let $w \in \Sigma^*$ be some word and $s = w_1 : w_2 : \cdots : w_n \in \Sigma^{*2}$ some 2-word. We write $s' := w \setminus s$ for $s' = ww_1 : ww_2 : \cdots : ww_n$. Note that $[w] \trianglelefteq (w \setminus s)$, i.e., $[w]$ is a prefix of s'. We say that s' is s *prefixed by w*.

Definition 3.1. Let $\sigma \in \Sigma$ and $b \in \Sigma^{+2}$. We call b a σ-*block* if $b = [\sigma]$ or $b = \sigma\tau \setminus s'$ for some $\tau \in \Sigma$ and some $s' \in \Sigma^{*2}$. If $b_1, b_2, \ldots, b_n$ are σ-blocks, then we call $b_1 : b_2 : \cdots : b_n$ a σ-*blockline*.

Note that every stack in $\mathrm{Stck}(\Sigma)$ forms a $\bot$-blockline. Furthermore, every blockline l decomposes uniquely as $l = b_1 : b_2 : \cdots : b_n$ of maximal blocks b_i in l.

Another crucial observation is that a σ-block $b \in \Sigma^{*2} \setminus \Sigma$ decomposes as $b = \sigma \setminus l$ for some blockline l and we call l the blockline *induced* by b. For a block of the form $[b]$ with $b \in \Sigma$, we define the blockline induced by $[b]$ to be ε.

Definition 3.2. Let l be a σ-blockline such that $l = b_1 : b_2 : \cdots : b_n$ is its decomposition into maximal blocks. Let $i_1, i_2, \ldots, i_m$ be those indices such that for all $1 \leq j \leq n$ we have $b_j \neq [\sigma]$ if and only if $j = i_k$ for some $1 \leq k \leq m$. For $1 \leq k \leq m$, let b'_{i_k} be the 2-word such that $b_{i_k} = \sigma \setminus b'_{i_k}$. We recursively define the *blocks of l* to be the minimal set containing $b_1, b_2, \ldots, b_n$ and the blocks of each of the b_{i_k} $(1 \leq k \leq m)$ seen as τ-blockline for some letter τ.

See Figure 2 for an example of a stack with one of its blocklines.

Recall that the symbols of a collapsible pushdown stack (of level 2) come from the set $\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ where Σ is the stack alphabet. For $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$, we encode a τ-blockline l in a tree as follows. The root of the tree is labelled by $(\mathrm{Sym}(\tau), \mathrm{CLvl}(\tau))$. The blockline induced by the first maximal block of l is encoded in the left subtree and the rest of l is encoded in the right subtree. This means that we only encode explicitly the symbol and the collapse level of each element of the stack, but not the collapse link. We will later see how to decode the collapse links from the encoding of a stack. When we encode a part of
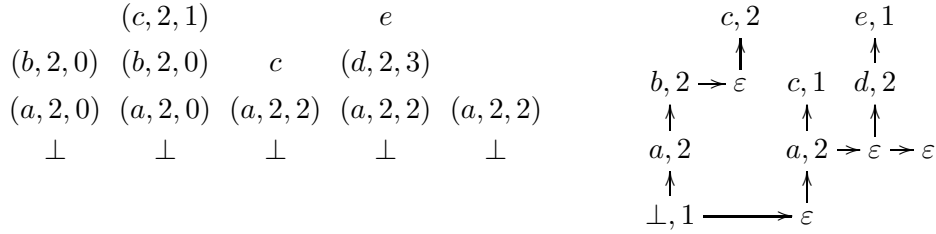
$$
\begin{array}{ccccc}
(c,2,1) & & e & & \\
(b,2,0) & (b,2,0) & c & (d,2,3) & \\
(a,2,0) & (a,2,0) & (a,2,2) & (a,2,2) & (a,2,2) \\
\bot & \bot & \bot & \bot & \bot
\end{array}
$$

Figure 3: 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).

a blockline in the right subtree, we do not repeat the label $(\mathrm{Sym}(\tau), \mathrm{CLvl}(\tau))$, but replace it by the empty word ε.

Definition 3.3. Let $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Furthermore, let

$$
s = w_1 : w_2 : \cdots : w_n \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}
$$

be some τ-blockline. Let w_i' be a word for each $1 \le i \le n$ such that $s = \tau \setminus [w_1' : w_2' : \cdots : w_n']$ and set $s' := w_1' : w_2' : \cdots : w_n'$. As an abbreviation we write $_i s_k := w_i : w_{i+1} : \cdots : w_k$. Let $_1 s_j$ be a maximal block of s. Note that $j > 1$ implies that there is some $\tau' \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ and there are words $w_{j'}''$ for each $j' \le j$ such that $w_{j'} = \tau \tau' w_{j'}''$.

For arbitrary $\sigma \in (\Sigma \times \{1,2\}) \cup \{\varepsilon\}$, we define recursively the $(\Sigma \times \{1,2\}) \cup \{\varepsilon\}$-labelled tree $\mathrm{Enc}(s, \sigma)$ via

$$
\mathrm{Enc}(s, \sigma) := \begin{cases}
\sigma & \text{if } |w_1| = 1, n = 1 \\
\sigma \langle \emptyset; \mathrm{Enc}(_2 s_n, \varepsilon) \rangle & \text{if } |w_1| = 1, n > 1 \\
\sigma \langle \mathrm{Enc}(_1 s_n', (\mathrm{Sym}(\tau'), \mathrm{CLvl}(\tau'))); \emptyset \rangle & \text{if } |w_1| > 1, j = n \\
\sigma \left\langle \mathrm{Enc}(_1 s_j', (\mathrm{Sym}(\tau'), \mathrm{CLvl}(\tau'))); \mathrm{Enc}(_{j+1} s_n, \varepsilon) \right\rangle & \text{otherwise}
\end{cases}
$$

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

Figure 3 shows a configuration and its encoding.

Remark 3.4. Fix some stack s. For $\sigma \in \Sigma$ and $k \in \mathbb{N}$, every $(\sigma, 2, k)$-block of s is encoded in a subtree whose root d is labelled $(\sigma, 2)$. We can restore k from the position of $d \in \{0,1\}^* 0$ in the tree $\mathrm{Enc}(s)$ as follows.

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

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

This correspondence can be seen as a bijection. Let $s = w_1 : w_2 : \cdots : w_n$ be some stack. We define the set $R := \mathrm{dom}(\mathrm{Enc}(s)) \cap (\{\varepsilon\} \cup \{0,1\}^* 1)$. Then there is a bijection $f : \{1, 2, 3, \ldots, n\} \to R$ such that i is mapped to the i-th element of R in lexicographic order. Each $1 \le i \le n$ represents the i-th word of s. f maps the first word of s to the root of $\mathrm{Enc}(s)$ and every other word in s to the element of $\mathrm{Enc}(s)$ that separates this word from its left neighbour in s.

If we interpret ε as empty word, the word from the root to $f(i)$ in $\mathrm{Enc}(s)$ is the greatest common prefix of w_{i-1} and w_i. More precisely, the word read along this path is the projection onto the letters and collapse levels of $w_{i-1} \sqcap w_i$.

Furthermore, set $f'(i) := f(i)0^m \in \text{Enc}(s)$ such that m is maximal with this property, i.e., $f'(i)$ is the leftmost descendant of $f(i)$. Then the path from $f(i)$ to $f'(i)$ is the suffix w'_i such that $w_i = (w_{i-1} \sqcap w_i)w'_i$ (here we set $w_0 := \varepsilon$). More precisely, the word read along this path is the projection onto the symbols and collapse levels of w'_i.

Having defined the encoding of a stack, we want to encode whole configurations, i.e., a stack together with a state. To this end, we just add the state as a new root of the tree and attach the encoding of the stack as left subtree, i.e., for some configuration (q, s) we set

$$\text{Enc}(q, s) := q \langle \text{Enc}(s); \emptyset \rangle .$$

The image of this encoding function contains only trees of a very specific type. We call this class $\mathbb{T}^{\text{Enc}}$. In the next definition we state the characterising properties of $\mathbb{T}^{\text{Enc}}$.

Definition 3.5. Let $\mathbb{T}^{\text{Enc}}$ be the class of trees T that satisfy the following conditions.
(1) The root of T is labelled by some element of Q ($T(\varepsilon) \in Q$).
(2) $1 \notin \text{dom}(T)$, $0 \in \text{dom}(T)$.
(3) $T(0) = (\bot, 1)$.
(4) Every element of the form $0\{0, 1\}^*0$ is labelled by some $(\sigma, l) \in (\Sigma \setminus \{\bot\}) \times \{1, 2\}$,
(5) Every element of the form $\{0, 1\}^*1$ is labelled by ε.
(6) There is no $t \in T$ such that $T(t0) = (\sigma, 1)$ and $T(t10) = (\sigma, 1)$.

Remark 3.6. Note that all trees in the image of Enc satisfy condition 6 due to the following. $T(t0) = T(t10) = (\sigma, 1)$ would imply that the subtree rooted at t encodes a blockline l such that the first block b_1 of l induces a σ-blockline and the second block b_2 induces also a σ-blockline. This contradicts the maximality of the blocks used in the encoding because all words of b_1 and b_2 have σ as second letter whence $b_1 : b_2$ forms a larger block. Note that for letters with links of level 2 the analogous restriction does not hold. In Figure 3 one sees the encoding of a stack s where $\text{Enc}(s)(0) = \text{Enc}(s)(10) = (a, 2)$. Here, the label $(a, 2)$ represents two different letters. $\text{Enc}(s)(0)$ encodes the element $(a, 2, 0)$, while $\text{Enc}(s)(10)$ encodes the element $(a, 2, 2)$, i.e., the first element encodes a letter a with undefined link and the second encodes the letter a with a link to the substack of width 2.

Lemma 3.7. *There is a finite automaton $\mathcal{A}_{\mathbb{T}\text{Enc}}$ with $2 + 3|\Sigma|$ many states that recognises $\mathbb{T}^{\text{Enc}}$.*

Proof. Set $\mathcal{A}_{\mathbb{T}\text{Enc}} := (Q_\mathcal{A}, Q \cup (\Sigma \times \{1, 2\}) \cup \{\varepsilon\}, \bot, \{q_I\}, \Delta_\mathcal{A})$ 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 node is labelled by $(\sigma, 1)$ ($P_\bot$ is used if no restriction applies). 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 $(\sigma, 1)$. For the definition of $\Delta_\mathcal{A}$ we use the following conventions. q ranges over Q, i, j range over $\{1, 2\}$, σ over Σ, τ over $\Sigma \setminus \{\bot\}$ and τ_σ over $\Sigma \setminus \{\bot, \sigma\}$ whenever σ is fixed. Set $\Delta_\mathcal{A} := \{(q_I, q, (\bot, 1), \bot), ((\sigma, i), (\sigma, i), (\tau, 1), P_\tau), ((\sigma, i), (\sigma, i), (\tau, 2), P_\bot), ((\sigma, i), (\sigma, i), (\tau, j), \bot), ((\sigma, i), (\sigma, i), \bot, P_\bot), ((\sigma, i), (\sigma, i), \bot, \bot), (P_\sigma, \varepsilon, (\tau_\sigma, 1), P_\tau), (P_\sigma, \varepsilon, (\tau, 2), P_\bot), (P_\sigma, \varepsilon, (\tau_\sigma, 1), \bot), (P_\sigma, \varepsilon, (\tau, 2), \bot), (P_\sigma, \varepsilon, \bot, P_\bot), (P_\sigma, \varepsilon, \bot, \bot)\}$. $\quad\square$

Lemma 3.8. Enc : $Q \times \text{Stck}(\Sigma) \to \mathbb{T}^{\text{Enc}}$ *is a bijection. We denote its inverse by* Dec.

The proof of this lemma is tedious. It can be found in Appendix A.

3.2. **Tree-automaticity of Collapsible Pushdown Graphs.** Our main technical contribution in this paper is stated in the next proposition. It concerns the regularity of the regular reachability predicates Reach_L with respect to our encoding of configurations. We postpone the proof of this proposition to Section 5.

Proposition 3.9. *There are polynomials p_1 and p_2 such that the following holds. Let $\mathcal{S} = (Q, \Sigma, \Gamma, q_0, \Delta)$ be some collapsible pushdown system of level 2 and let L be some regular language over Γ recognised by some nondeterministic finite automaton with state set P. Reach_L is tree-automatic via Enc and there is a nondeterministic finite tree-automaton with $p_1(|\Sigma|) \cdot \exp(p_2(|Q| \cdot |P|))$ many states recognising Reach_L in this encoding.*

Remark 3.10. In the proposition, Reach_L has to be understood with respect to all possible configurations of a level 2 collapsible pushdown system as opposed to those occurring in the configuration graph of $\mathcal{S}$, i.e., those reachable via the transitions of $\mathcal{S}$ from the initial configuration of $\mathcal{S}$.

We obtain the automaticity of the ε-contractions of all level 2 collapsible pushdown graphs as a direct corollary of the previous result.

Corollary 3.11. *There are polynomials p and q such that the following holds. Given a CPS $\mathcal{S} = (Q, \Sigma, \Gamma, q_0, \Delta)$, the ε-contraction $\mathrm{CPG}(\mathcal{S})/\varepsilon$ is regular via Enc. Moreover, there is a presentation such that each automaton in the presentation of $\mathrm{CPG}(\mathcal{S})/\varepsilon$ has at most $p(|\Sigma|) \cdot \exp(q(|Q|))$ many states.*

Proof. The domain of $\mathrm{CPG}(\mathcal{S})/\varepsilon$ is $\big\{c : \mathrm{CPG}(\mathcal{S}) \models \mathrm{Reach}_{(\Gamma^*(\Gamma \setminus \{\varepsilon\}))^*}((q_0, \perp_2), c)\big\}$. Note that $(\{\varepsilon\}^*(\Gamma \setminus \{\varepsilon\}))^*$ is accepted by an automaton with 2 states. Furthermore, hard-coding $\mathrm{Enc}(q_0, \perp_2)$ as first argument to the automaton from Proposition 3.9 increases the number of states by a at most a factor 3 (because $\mathrm{dom}(\mathrm{Enc}(q_0, \perp_2)) = \{\varepsilon, 0\}$). Thus, the corresponding automaton has $3 \cdot p_1(|\Sigma|) \cdot \exp(p_2(2 \cdot |Q|))$ many states where p_1 and p_2 are the polynomials from Proposition 3.9.

Similarly, $\vdash^\gamma$ in $\mathrm{CPG}(\mathcal{S})/\varepsilon$ is exactly the same as $\mathrm{Reach}_{\{\varepsilon\}^*\gamma}$. Again 2 states suffice to recognise $\{\varepsilon\}^*\gamma$. $\qquad\square$

3.3. **Lower Bound for FO Model-Checking.** Since CPG are tree-automatic, their FO model-checking problem is decidable. The algorithm obtained this way has nonelementary complexity. In this section we prove that we cannot do better: there is a fixed collapsible pushdown graph of level 2 whose FO theory has nonelementary complexity. We present a reduction of FO model-checking on the full infinite binary tree to FO model-checking on this collapsible pushdown graph. Recall that FO model-checking on the full infinite binary tree $\mathfrak{T} := (T, \preceq, S_1, S_2)$ with prefix order $\preceq$ and successor relations S_1, S_2 has a nonelementary lower bound (cf. Example 8.3 in [9]).

Theorem 3.12. *The expression complexity of any FO model-checking algorithm for level 2 collapsible pushdown graphs is nonelementary.*

Remark 3.13. Note that this is a statement about plain collapsible pushdown graphs and not about the ε-contractions. In contrast to the theorem, the first-order model-checking problem on non-ε-contracted level 1 pushdown graphs is complete for alternating exponential time [17].

Proof. We modify the CPS of Example 2.7. We add the transition $(2, a, \mathrm{P}_2, \mathrm{pop}_2, 0)$. Note that the ordinal $(\omega, \preceq)$ is first order definable in this graph: restrict the domain to all elements with state 0. The order $\preceq$ is then defined via $\varphi_{\preceq}(x, y) := \exists z \; z \vdash^{\mathrm{P}_2} y \wedge z \vdash^{\mathrm{Co}} x$.

Now, we obtain the binary tree $(T, \preceq)$ by use of the stack alphabet $\{\bot, a, b\}$. For each occurrence of a in a transition δ, we make a copy of δ where we replace a by b. Then, each configuration c with state 0 is determined by $\mathrm{top}_2(c)$ and these are in bijection to the set $\{a, b\}^*$. Furthermore, $\varphi_{\preceq}$ defines the prefix relation on this set. Thus, $(T, \preceq, S_1, S_2)$ is FO-interpretable in this graph whence its FO theory has nonelementary complexity. $\qquad\square$

4. DECOMPOSITION OF RUNS

In this section we develop the technical background for the proof that the regular reachability predicates are tree-automatic via Enc. We investigate the structure of runs of CPS. We prove that any run is composed from subruns which can be classified as *returns*, *loops*, or 1-*loops*. Forgetting about technical details, one can say that returns are runs from some stack s to $\mathrm{pop}_2(s)$, loops are runs that start and end in the same stack and 1-loops are runs from some stack s to a stack s' such that s and s' share the same topmost word and s is a substack of s'. Every run decomposes as a sequence of the form $\lambda_0 \circ \rho_1 \circ \lambda_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \lambda_n$ where the ρ_i only perform one operation each and the λ_i are returns, loops or 1-loops of maximal length in a certain sense. Let us explain this idea precisely in the case of a run ρ from some stack s_1 to a stack s_2 such that $s_1 = \mathrm{pop}_2{}^k(s_2)$. In this special case, the λ_i are all loops and the sequence of operations induced by $\rho_1, \ldots, \rho_n$ is a sequence of minimal length transforming s_1 into s_2. This sequence of minimal length is in fact unique up to replacement of pop_1 and collapse operations of level 1. As a direct consequence, the loops λ_i occurring in the decomposition cover the largest possible part of ρ in terms of loops, returns and 1-loops. This is also the key to understanding our decomposition result for general runs: we identify maximal subruns of an arbitrary run which are returns, loops and 1-loops and we prove that the parts not contained in one of these subruns form a short sequence of operations.

Hence, understanding the existence of returns, loops and 1-loops allows to clarify whether runs between certain configurations exist. It turns out that our decomposition is very suitable for the analysis with finite automata because such automata can be used to decide whether returns, loops and 1-loops starting in a given stack exist.

We next start with a general decomposition of any run into four parts. Afterwards we prove decomposition results for each of the parts where returns, loops and 1-loops are the central pieces of the decomposition. Finally, we show how finite automata acting on the topmost word of a stack can be used to compute the existence of returns, loops and 1-loops starting at this stack.

4.1. **Decomposition of General Runs.** We introduce a decomposition of an arbitrary run ρ into four parts. The idea is that every run from a stack s_1 to a stack s_2 passes a minimal common substack t of s_1 and s_2. Any run from s_1 to t decomposes into a first part from s_1 to a stack of the form $t_1 := \mathrm{pop}_2{}^k(s_1)$ such that $|t_1| = |t|$ and a second part from t_1 to t. Similarly, for the unique stack $t_2 := \mathrm{pop}_2{}^j(s_2)$ such that $|t_2| = |t|$ the run from t to s_2 decomposes into a run from t to t_2 and a run from t_2 to s_2. In the following sections

we prove that every part of this decomposition again decomposes into returns, loops, and 1-loops.

Lemma 4.1. *Let $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$ be configurations and $\rho \in \mathrm{Runs}(c_1, c_2)$. Let t be the minimal substack of s_1 such that ρ visits t. Furthermore, let $m_1 := \mathrm{pop}_2^{|s_1|-|t|}(s_1)$ and $m_2 := \mathrm{pop}_2^{|s_2|-|t|}(s_2)$. ρ decomposes as $\rho = \rho_1 \circ \rho_2 \circ \rho_3 \circ \rho_4$ where*

- *$\rho_1 \in \mathrm{Runs}(s_1, m_1)$ does not visit any substack of m_1 before its final configuration,*
- *$\rho_2 \in \mathrm{Runs}(m_1, t)$ does not visit any substack of t before its final configuration,*
- *$\rho_3 \in \mathrm{Runs}(t, m_2)$ does not visit a substack of $\mathrm{pop}_1(t)$, and*
- *$\rho_4 \in \mathrm{Runs}(m_2, s_2)$ does not visit any substack of m_2 after its initial configuration.*

Proof. Let $i_2 \in \mathrm{dom}(\rho)$ be minimal such that $\rho(i_2) = t$. If $t = m_1$ then set $i_1 := i_2$. Otherwise there is some minimal $i_1 < i_2$ such that $\rho(i_1) = m_1$: note that a stack operation alters either the width of the stack or the content of the topmost word. Thus, before reaching t, ρ must visit some stack $\hat{m}$ of width at most $|t|$ and of the form $\hat{m} = \mathrm{pop}_2^k(s_1)$ for some $k \in \mathbb{N}$. Since $\hat{m}$ cannot be a substack of t, $\hat{m} = m_1$. Thus, we set $\rho_1 := \rho\!\restriction_{[0,i_1]}$ and $\rho_2 := \rho\!\restriction_{[i_1,i_2]}$.

For the definition of ρ_3 and ρ_4, note that $|m_2| = |t|$. Let $i_2 < i_3 \in \mathrm{dom}(\rho)$ be maximal such that $|\rho(i_3)| = |t|$. Since the first $|t|$ words of the stack are not changed by ρ after i_3, $\rho(i_3) = m_2$ and i_3 is the last occurrence of m_2. Setting $\rho_3 := \rho\!\restriction_{[i_2,i_3]}$ and $\rho_4 := \rho\!\restriction_{[i_3,\mathrm{length}(\rho)]}$ we are done. $\qquad\square$

This decomposition motivates the following definition.

Definition 4.2. Given a collapsible pushdown system $\mathcal{S}$, we define the following four relations on the configurations of $\mathcal{S}$:

$$R^{\Leftarrow} := \{(c_1, c_2) \in \mathrm{Cnf}^2 : c_2 = \mathrm{pop}_2^k(c_1) \text{ and } \exists \rho \in \mathrm{Runs}(c_1, c_2) \forall i < \mathrm{length}(\rho) \quad \rho(i) \not\leq c_2\}$$

$$R^{\Downarrow} := \{(c_1, c_2) \in \mathrm{Cnf}^2 : c_2 = \mathrm{pop}_1^k(c_1) \text{ and } \exists \rho \in \mathrm{Runs}(c_1, c_2) \forall i < \mathrm{length}(\rho) \quad \rho(i) \not\leq c_2\}$$

$$R^{\Uparrow} := \{(c_1, c_2) \in \mathrm{Cnf}^2 : c_1 = \mathrm{pop}_1^k(c_2) \text{ and } \exists \rho \in \mathrm{Runs}(c_1, c_2) \forall i \leq \mathrm{length}(\rho) \quad \rho(i) \not< c_1\}$$

$$R^{\Rightarrow} := \{(c_1, c_2) \in \mathrm{Cnf}^2 : c_1 = \mathrm{pop}_2^k(c_2) \text{ and } \exists \rho \in \mathrm{Runs}(c_1, c_2) \forall i > 0 \quad \rho(i) \not\leq c_1\}.$$

Remark 4.3. Since we allow runs of length 0, the relations $R^{\Leftarrow}$, $R^{\Downarrow}$, $R^{\Uparrow}$ and $R^{\Rightarrow}$ are reflexive. Lemma 4.1 states that

$$(c_1, c_2) \in \mathrm{Reach} \Leftrightarrow \exists d, e, f \quad (c_1, d) \in R^{\Leftarrow} \wedge (d, e) \in R^{\Downarrow} \wedge (e, f) \in R^{\Uparrow} \wedge (f, c_2) \in R^{\Rightarrow}.$$

In Section 5.2 we show that the relations $R^{\Leftarrow}, R^{\Downarrow}, R^{\Uparrow}$ and $R^{\Rightarrow}$ are automatic whence Reach is also automatic. In the next section, we prove a decomposition result that especially applies to all runs in $R^{\Rightarrow}$. Afterwards, in Section 4.3 we provide a corresponding decomposition result for all runs in $R^{\Leftarrow}$. Finally, we provide (much simpler) decompositions for $R^{\Downarrow}$ and $R^{\Uparrow}$ in Section 4.4.

4.2. **Milestones, Loops and Increasing Runs.** In this section we aim at a decomposition result for all runs in $R^{\Rightarrow}$. For this purpose we first introduce the notion of *generalised milestones* of some stack s. The underlying idea is as follows. Some stack m is a generalised milestone of s if any run from the initial configuration to s of any collapsible pushdown system also passes m. Moreover, if some run ending in s passes some generalised milestone m of s, then it passes all generalised milestones of s that are not generalised milestones of

m. From the definition of generalised milestones it will be obvious that every run ρ in $R^{\Rightarrow}$ starts at a generalised milestone of its final stack. Thus, a run in $R^{\Rightarrow}$ can be decomposed into parts that connect one generalised milestone of its final stack with the next generalised milestone. After introducing the precise notion of a loop we will see that each of these parts consists of such a loop plus one further transition. At a first glance, the formal definition of a generalised milestone has nothing to do with our informal description. The connection between the intended meaning and the formal definition is that in order to create some stack s from the initial stack $\perp_2$, we have to create it word-by-word and each word letter-by-letter in the following sense. If we want to create $s = w_1 : w_2 : \cdots : w_k$, we have to use push operations in order to create the first word, i.e., the stack $[w_1]$. Then we have to apply clone$_2$ and obtain $w_1 : w_1$. In order to generate s from this stack, we first have to generate $w_1 : w_2$ and then we can proceed generating the other words of s. But for this purpose, we first have to remove every letter from the second copy of w_1 until we reach the greatest common prefix of w_1 and w_2. This can only be done by iteratively applying pop$_1$ or collapse operations of level 1. Having reached $w_1 : (w_1 \sqcap w_2)$, we again start to create $w_1 : w_2$ by using push operations of level 1.

This way of creating s from $\perp_2$ is the shortest method to create s which is unique up to replacements of pop$_1$ operations by collapse of level 1 and vice versa. At the same time any other method contains this pattern as a (scattered) subsequence (again up to replacement of pop$_1$ by collapse of level 1 and vice versa). If we deviate from the described way of creating s, then we just insert some loops where we first create some different stack and then return to the position where we started to deviate. At the end of this section, Corollary 4.10 will show that our intuition is correct. Let us now formally define generalised milestones.

Definition 4.4. Let $s = w_1 : w_2 : \cdots : w_k$ be a stack and let $w_0 = \perp$. We call a stack m a *generalised milestone of* s if m is of the form

$$m = w_1 : w_2 : \cdots : w_i : v_{i+1} \text{ where } 0 \leq i < k,$$
$$w_i \sqcap w_{i+1} \leq v_{i+1} \text{ and}$$
$$v_{i+1} \leq w_i \text{ or } v_{i+1} \leq w_{i+1}.$$

We denote by $\mathrm{GMS}(s)$ the set of all generalised milestones of s.

For a generalised milestone m of s, we call m a *milestone of* s if m is a substack of s, i.e., if $v_{i+1} \leq w_{i+1}$ in the above definition. We write $\mathrm{MS}(s)$ for the set of all milestones of s.

We next define a partial order that turns out to be linear when restricted to the set $\mathrm{GMS}(s)$.

Definition 4.5. We define a partial order $\ll$ on all stacks as follows. $\ll$ is the smallest reflexive and transitive relation that satisfies the following conditions. Let $s = w_1 : w_2 : \cdots : w_k$ and $t = v_1 : v_2 : \cdots : v_l$ be stacks. $s \ll t$ holds if

(1) $|s| < |t|$, or
(2) $l = k$, $w_i = v_i$ for $i < k$ and $v_k < w_k \leq w_{k-1} = v_{k-1}$, or
(3) $l = k$, $w_i = v_i$ for $i < k$ and $w_k < v_k \nleq w_{k-1} = v_{k-1}$.

For each stack s, we now characterise $\ll$ restricted to $\mathrm{GMS}(s)$. The straightforward proofs of the following lemmas are left to the reader.

Lemma 4.6. *Let $s = w_1 : w_2 : \cdots : w_k$ be a stack and $m_1, m_2 \in \mathrm{GMS}(s)$. Then $m_1 \ll m_2$ if one of the following holds:*

(1) $|m_1| < |m_2|$,
(2) $|m_1| = |m_2| = i$, $w_{i-1} \sqcap w_i < \mathrm{top}_2(m_1) \leq w_{i-1}$ *and* $w_{i-1} \sqcap w_i \leq \mathrm{top}_2(m_2) \leq w_i$,
(3) $|m_1| = |m_2| = i$ *and* $w_{i-1} \sqcap w_i < \mathrm{top}_2(m_2) \leq \mathrm{top}_2(m_1) \leq w_{i-1}$, *or*
(4) $|m_1| = |m_2| = i$ *and* $w_{i-1} \sqcap w_i \leq \mathrm{top}_2(m_1) \leq \mathrm{top}_2(m_2) \leq w_i$.

Lemma 4.7. *For each stack s, $\ll$ induces a finite linear order on $\mathrm{GMS}(s)$. Moreover, if $m \in \mathrm{GMS}(s)$ then $(\mathrm{GMS}(m), \ll)$ is the initial segment of $(\mathrm{GMS}(s), \ll)$ up to m.*

We call some $m \in \mathrm{GMS}(s)$ the i-th generalised milestone if it is the i-th element of $\mathrm{GMS}(s)$ with respect to $\ll$. Later we will see that $\ll$ corresponds to the order in which the generalised milestones appear in any run from the initial configuration to s. Note that the restriction of $\ll$ to $\mathrm{MS}(s)$ coincides with the substack relation $\leq$.

Now we introduce loops formally and characterise runs connecting generalised milestones in terms of loops. Later we will see that there is a close correspondence between the milestones of some stack s and the nodes of our encoding $\mathrm{Enc}(s)$. This correspondence is one of the key observation in proving that $\mathrm{Enc}(s)$ yields a tree-automatic encoding of the relation $R^{\Rightarrow}$.

Definition 4.8. Let s be a stack and q, q' states. A *loop* from (q, s) to (q', s) is a run $\lambda \in \mathrm{Runs}((q, s), (q', s))$ that does not pass a substack of $\mathrm{pop}_2(s)$ and that may pass $\mathrm{pop}_1{}^k(s)$ only if the k topmost elements of $\mathrm{top}_2(s)$ are letters with links of level 1. This means that for all $i \in \mathrm{dom}(\lambda)$, if $\lambda(i) = (q_i, \mathrm{pop}_1{}^k(s))$ then $\mathrm{CLvl}(\mathrm{pop}_1{}^{k'}(s)) = 1$ for all $0 \leq k' < k$.

If λ is a loop from (q, s) to (q', s) such that $\lambda(1) = \mathrm{pop}_1(s) = \lambda(\mathrm{length}(\lambda) - 1)$, then we call λ a *low loop*. If λ is a loop from (q, s) to (q', s) that never passes $\mathrm{pop}_1(s)$, then we call λ a *high loop*.

For s some stack, we sometimes write λ is a loop *of* s. By this we express that λ is a loop and its initial (and final) stack is s. We now characterise runs connecting milestones of some stack in terms of loops.

Lemma 4.9. *Let ρ be a run ending in stack $s = w_1 : w_2 : \cdots : w_k$. Furthermore, let $m \in \mathrm{GMS}(s) \setminus \{s\}$ be such that ρ visits m. Let $i \in \mathrm{dom}(\rho)$ be maximal such that ρ visits m at i, i.e., the stack at $\rho(i)$ is m. Then ρ also visits the $\ll$-minimal generalised milestone $m' \in \mathrm{GMS}(s) \setminus \mathrm{GMS}(m)$ and for $i' \in \mathrm{dom}(\rho)$ maximal such that $\rho(i') = m'$, $\rho\!\restriction_{[i+1,i']}$ is a loop of m'.*

Proof. We distinguish the following cases.
- Assume that $m' = \mathrm{clone}_2(m)$. In this case $m = w_1 : w_2 : \cdots : w_{|m|}$. Thus, at the last position $j \in \mathrm{dom}(\rho)$ where $|\rho(j)| = |m|$, the stack at $\rho(j)$ is m (because ρ never changes the first $|m|$ many words after passing $\rho(j)$). Hence, $i = j$ by definition. Since $|s| > |m|$, it follows directly that the operation at i is a clone_2 leading to m'. Note that ρ never passes a stack of width $|m|$ again. Thus, it follows from Lemma 2.11 that for i' maximal with $\rho(i') = m'$ the run $\rho\!\restriction_{[i+1,i']}$ never visits $\mathrm{pop}_1{}^k(m')$ if $\mathrm{CLvl}(\mathrm{pop}_1{}^{k-1}(m')) = 2$. Thus, we conclude that $\rho\!\restriction_{[i+1,i']}$ is a loop.
- Assume that $m' = \mathrm{pop}_1(m)$. In this case, $m = w_1 : w_2 : \cdots : w_{|m|-1} : w$ for some w such that $w_{|m|-1} \sqcap w_{|m|} < w \leq w_{|m|-1}$. Thus, $w \not\leq w_{|m|}$ and creating $w_{|m|}$ as the $|m|$-th word on the stack requires passing $w_1 : w_2 : \cdots : w_{|m|-1} : w_{|m|-1} \sqcap w_{|m|}$. This is only possible via applying pop_1 or collapse of level 1 to m. Since we assumed i to be maximal, the operation at i must be pop_1 or collapse of level 1 and leads to m'.

We still have to show that $\rho\restriction_{[i+1,i']}$ is a loop. By definition of i, $\rho\restriction_{[i+1,i']}$ starts and ends in m'. Due to the maximality of i, $\rho\restriction_{[i+1,i']}$ does not visit the stack $\mathrm{pop}_2(m) = \mathrm{pop}_2(m')$. Furthermore, $\mathrm{top}_1(m')$ is a cloned element. Due to Lemma 2.11, if $\rho\restriction_{[i+1,i']}$ visits $\mathrm{pop}_1{}^k(m')$ then $\mathrm{CLvl}(\mathrm{pop}_1{}^{k-1}(m')) = 1$. Thus, $\rho\restriction_{[i+1,i']}$ is a loop.

- The last case is $m' = \mathrm{push}_{\sigma,l}(m)$ for $(\sigma, l) \in \Sigma \times \{1, 2\}$. In this case,

$$m = w_1 : w_2 : \cdots : w_{|m|-1} : w$$

for some w such that $w_{|m|-1} \sqcap w_{|m|} \leq w < w_{|m|}$. Creating $w_{|m|}$ on the stack requires pushing the missing symbols onto the stack as they cannot be obtained via clone operation from the previous word. Since i is maximal, the operation at i is some $\mathrm{push}_{\sigma,l}$ leading to m'. $\rho\restriction_{[i,i']}$ is a high loop due to the maximality of i (this part of ρ never visits $m = \mathrm{pop}_1(m')$ or any other proper substack of m'). $\qquad\square$

As a corollary of the lemma, we obtain that $\ll$ coincides with the order in which the generalised milestones appear for the last time in a given run starting in the initial configuration.

Corollary 4.10. *Let s be some stack and $m_1 \in \mathrm{MS}(s)$. Some run $\rho \in \mathrm{Runs}(m_1, s)$ that does not visit substacks of m_1 (after the initial configuration) decomposes as*

$$\rho = \rho_1 \circ \lambda_1 \circ \cdots \circ \rho_n \circ \lambda_n$$

where the λ_i are loops and the ρ_i are runs of length 1 that connect one generalised milestone of s with its $\ll$-successor (in $\mathrm{GMS}(s)$).

In particular, for $t = \mathrm{pop}_2{}^k(s)$ and configurations $c = (q, t)$ and $c' = (q', s)$ a run $\rho \in \mathrm{Runs}(c, c')$ witnesses $(c, c') \in R^{\Rightarrow}$ if and only if it decomposes as given above.

For the direction from right to left of the last claim note that, if $t = \mathrm{pop}_2{}^k(s)$ and $\rho \in \mathrm{Runs}(t, s)$ decomposes as above, then ρ_1 performs a clone_2. This implies that the run $\lambda_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \lambda_n$ cannot visit t again.

This corollary shows that it is sufficient to understand loops of generalised milestones of a given stack s in order to understand runs in $R^{\Rightarrow}$ ending in s. In Section 4.6 we show that the loops of a given stack s can be computed by a finite automaton on input $\mathrm{top}_2(s)$.

4.3. **Returns, 1-Loops and Decreasing Runs.** Having analysed the form of runs in $R^{\Rightarrow}$, we now analyse runs in the converse direction: how can we decompose a run in $R^{\Leftarrow}$? We need the notion of *returns* and of *level-1-loops* in order to answer this question.

Definition 4.11. Let $t = s : w$ be some stack with topmost word w. A *return from t to s* is a run $\rho \in \mathrm{Runs}(t, s)$ such that ρ never visits a substack of s before $\mathrm{length}(\rho)$ and such that one of the following holds:

(1) the last operation in ρ is pop_2, or
(2) the last operation in ρ is a collapse and $w < \mathrm{top}_2(\rho(\mathrm{length}(\rho) - 1))$, i.e., ρ pushes at first some new letters onto t and then performs a collapse of one of these new letters, or
(3) there is some $i \in \mathrm{dom}(\rho)$ such that $\rho\restriction_{[i,\mathrm{length}(\rho)]}$ is a return from $\mathrm{pop}_1(t)$ to s.

Remark 4.12. The technical restrictions in the second condition have the following intention. A return from t to $\mathrm{pop}_2(t)$ is a run ρ from t to $\mathrm{pop}_2(t)$ that does not use the level 2 links stored in $\mathrm{top}_2(t)$ (cf. [11] for a detailed discussion).

It is useful to note that any return from some stack s that visits $\mathrm{pop}_1(s)$ in fact satisfies the last condition in the definition of return.

Lemma 4.13. *Let ρ be some return from a stack s to $\mathrm{pop}_2(s)$. If $0 < i < \mathrm{length}(\rho)$ is the first position such that ρ visits $\mathrm{pop}_1(s)$ at i, then $\rho\!\restriction_{[i,\mathrm{length}(\rho)]}$ is a return from $\mathrm{pop}_1(s)$ to $\mathrm{pop}_2(s)$, i.e., i witnesses that ρ satisfies the third condition in the definition of a return.*

In the case that the topmost word of the initial stack only contains cloned elements, then there is an easy condition to verify that a run starting at this stack is a return.

Lemma 4.14. *Let s and t be stacks. Assume that $|t| < |s|$ and that $\mathrm{top}_2(s) \leq \mathrm{top}_2(t)$. For ρ some run of length l starting at stack s, ρ is a return if*
(1) *for all $0 \leq i < l, |\rho(i)| \geq |s|$ and*
(2) *$|t| \leq |\rho(l)| < |s|$.*

Proof. The proof is by induction on the length of $\mathrm{top}_2(s)$. Note that the operation at $\rho(l-1)$ is either pop_2 or a collapse of level 2. If it is a pop_2, then it is immediate that ρ is a return. If it is a collapse of level 2, then there is a prefix $w \leq \mathrm{top}_2(s)$ and some word v such that $\mathrm{top}_2(\rho(l-1)) = wv$. Since all level 2 links in $\mathrm{top}_2(s)$ point to stacks of width smaller than t, v must be nonempty by assumption (2). Thus, if $w = \mathrm{top}_2(s)$ we conclude immediately that ρ is a return. if $w < \mathrm{top}_2(s)$ then the only way to create a word wv on a stack of width at least $|s|$ with a link to a stack of smaller width requires to visit $\mathrm{pop}_2(s) : w$. But this implies that ρ visits $\mathrm{pop}_1(s)$ and we conclude by application of the induction hypothesis to the final part of ρ starting at the first occurrence of $\mathrm{pop}_1(s)$. $\qquad\square$

Definition 4.15. Let s be some stack and w some word. A run λ of length n is called a *level-1-loop* (or 1-loop) of $s : w$ if the following conditions are satisfied.
(1) $\lambda \in \mathrm{Runs}(s : w, s : s' : w)$ for some 2-word s',
(2) for every $i \in \mathrm{dom}(\lambda)$, $|\lambda(i)| > |s|$, and
(3) for every $i \in \mathrm{dom}(\lambda)$ such that $\mathrm{top}_2(\lambda(i)) = \mathrm{pop}_1(w)$, there is some $j > i$ such that $\lambda\!\restriction_{[i,j]}$ is a return.

Before we analyse the form of runs in $R^{\Leftarrow}$, we prove an auxiliary lemma.

Lemma 4.16. *Let ρ be a run of length l starting in some stack s with topmost word w such that*
(1) *ρ does not visit a substack of $\mathrm{pop}_2(s)$ before its final configuration,*
(2) *$\mathrm{top}_2(\rho(l-1))$ is a proper prefix of w, and*
(3) *the last stack operation in ρ is a collapse of level 2.*
If ρ is not a return, then there is some $0 < i < l$ such that
(1) *$\mathrm{top}_2(\rho(i)) = \mathrm{pop}_1(w)$,*
(2) *for all $i \leq j \leq l$ the subrun $\rho\!\restriction_{[i,j]}$ is not a return, and*
(3) *for all $0 < j < i$ such that $\mathrm{top}_2(\rho(j)) = \mathrm{pop}_1(w)$ there is a $j < k < i$ such that $\rho\!\restriction_{[j,k]}$ is a return.*

Proof. Assume that ρ is such a run and that it is not a return. We first prove that there is some $0 < i < l$ such that
- $\mathrm{top}_2(\rho(i)) = \mathrm{pop}_1(w)$ and
- for all $i \leq j \leq l$ the subrun $\rho\!\restriction_{[i,j]}$ is not a return.

Afterwards we show that the minimal such i also satisfies claim (3).

(1) Assume that the stack at $\rho(l-1)$ decomposes as $w_1 : w_2 : \cdots : w_k$ such that $w \not\leq w_{|s|}$. In this case let $i \leq l-1$ be minimal such that $|\rho(i)| = |s|$ and $w \not\leq \mathrm{top}_2(\rho(i))$. Since ρ does not visit a substack of $\mathrm{pop}_2(s)$ before i, we conclude immediately that the stack at $i-1$ is s and the stack at i is $\mathrm{pop}_1(s)$. We show that i witnesses the claim. Note that $\mathrm{top}_2(\rho(i)) = \mathrm{pop}_1(w)$. Heading for a contradiction, assume that there is some $i \leq j \leq l$ such that $\rho' := \rho\restriction_{[i,j]}$ is a return. This assumption implies directly that $|\rho(j)| < |\rho(i)| = |s|$ whence ρ' visits a substack of $\mathrm{pop}_2(s)$. Since ρ does not do so before l, we conclude that $l = j$. But this implies that ρ is a run that starts in s, passes $\mathrm{pop}_1(s)$ and continues with a return from $\mathrm{pop}_1(s)$. By definition, this implies that ρ is a return contradicting our assumptions. Thus, we conclude that there is no $j \geq i$ such that $\rho\restriction_{[i,j]}$ is a return.

(2) Otherwise, the stack at $\rho(l-1)$ decomposes as $w_1 : w_2 : \cdots : w_k$ for some $k > |s|$ and there is some $n > |s|$ such that $w \leq w_i$ for all $|s| \leq i < n$ and $w \not\leq w_n$. Let $i_0 \leq l-1$ be maximal such that $|\rho(i_0 - 1)| < n$. Then the stack at $\rho(i_0 - 1)$ is $w_1 : w_2 : \cdots : w_{n-1}$ and the operation at $i_0 - 1$ is clone_2. Thus, $w \leq \mathrm{top}_2(\rho(i_0))$. Let $i_0 < i_1 < l-1$ be minimal such that $|\rho(i_1)| = n$ and $\mathrm{top}_2(\rho(i_1)) < w$. By minimality of i_1, we conclude that $|\rho(i_1 - 1)| = n$ whence $\mathrm{top}_2(\rho(i_1 - 1)) = w$ and $\mathrm{top}_2(\rho(i_1)) = \mathrm{pop}_1(w)$. In order to prove that i_1 witnesses the claim of the lemma, we have to prove that for all $i_1 \leq j \leq l$, $\rho\restriction_{[i_1,j]}$ is not a return. But note that $|\rho(i_1)| = n \leq |\rho(j)|$ for all $i_1 \leq j < l$ whence $\rho\restriction_{[i_1,j]}$ is not a return for $i_1 \leq j < l$. Moreover, since ρ ends in a collapse on a prefix of $w = \mathrm{top}_2(\rho(0))$, we conclude that $|\rho(l)| < |\rho(0)| = |s| < n$. Thus, $|\rho(i_1)| - |(\rho(l)| \geq 2$ and $\rho\restriction_{[i_1,l]}$ is not a return because it does not end in $\mathrm{pop}_2(\rho(i_1))$.

This completes the first part of our proof. We still have to deal with claim (3). For this purpose let $0 < i < l$ be minimal such that

- $\mathrm{top}_2(\rho(i)) = \mathrm{pop}_1(w)$ and
- for all $i \leq j \leq l$ the subrun $\rho\restriction_{[i,j]}$ is not a return.

Heading for a contradiction, assume that there is some $0 < j < i$ with $\mathrm{top}_2(\rho(j)) = \mathrm{pop}_1(w)$ such that there is no $j < k \leq i$ such that $\rho\restriction_{[j,k]}$ is a return.

By minimality of i there is some $k > i$ such that $\rho_j := \rho\restriction_{[j,k]}$ is a return. Since ρ is no return, we directly conclude that $\rho(j) \neq \mathrm{pop}_1(s)$ whence $|\rho(j)| > |s|$. We distinguish two cases.

(1) If $|\rho(j)| > |\rho(i)|$ then the minimal $k_0 \geq j$ such that $|\rho(k_0)| < |\rho(j)|$ satisfies $k_0 \leq i$. But by definition, k_0 is the only candidate for $\rho\restriction_{[j,k_0]}$ being a return. Thus, $k = k_0$ which contradicts the assumption that $k > i$.

(2) If $|\rho(j)| \leq |\rho(i)|$, we conclude that $|s| \leq |\rho(k)| < |\rho(i)|$. Thus, there is also a minimal $k_0 \geq i$ such that $|s| \leq |\rho(k_0)| < |\rho(i)|$. Since $\mathrm{top}_2(\rho(i)) \leq \mathrm{top}_2(s) = w$, we conclude with Lemma 4.14 that $\rho\restriction_{[i,k_0]}$ is a return contradicting our choice of i.

$\square$

With this lemma we are prepared to prove our decomposition result.

Lemma 4.17. *Let $s, s' \in \mathrm{Stck}(\Sigma)$ such that $s' = \mathrm{pop}_2{}^k(s)$ for some $k \in \mathbb{N}$ and let ρ be some run. ρ is a run in $\mathrm{Runs}(s, s')$ that does not visit a substack of s' before its final configuration if and only if $\rho \in \mathrm{Runs}(s, s')$ and ρ decomposes as $\rho = \rho_1 \circ \rho_2 \circ \cdots \circ \rho_n$ where each ρ_i is of one of the following forms.*

F1. *ρ_i is a return,*

F2. ρ_i *is a* 1-*loop followed by a* collapse *of collapse level* 2,

F3. ρ_i *is a* 1-*loop followed by a* pop$_1$ *(or a* collapse *of collapse level* 1*), there is a* $j > i$ *such that* ρ_j *is of the form F2 and for all* $i < k < j$ ρ_k *is of the form F3.*

Proof. First of all, note that the case length$(\rho) = 0$ is solved by setting $n := 0$.

We proceed by induction on the length of ρ. Assume that $\rho \in \mathrm{Runs}(s, s')$ and that it does not visit s' before the final configuration. We write (q_i, s_i) for the configuration $\rho(i)$. Firstly, consider the case that there is some $m \in \mathrm{dom}(\rho)$ such that $\rho_1 := \rho\!\restriction_{[0,m]}$ is a return. Then ρ_1 is of the form F1. By induction hypothesis, $\rho\!\restriction_{[m,\mathrm{length}(\rho)]}$ decomposes as desired.

Otherwise, assume that there is no $m \in \mathrm{dom}(\rho)$ such that $\rho\!\restriction_{[0,m]}$ is a return.

Nevertheless, there is a minimal $m \in \mathrm{dom}(\rho)$ such that $|s_m| < |s|$. The last operation of $\hat\rho := \rho\!\restriction_{[0,m]}$ is a collapse such that $\mathrm{top}_2(s_{m-1}) \leq \mathrm{top}_2(s)$ (otherwise $\hat\rho$ would be a return). Writing $w := \mathrm{top}_2(s_{m-1})$, we distinguish two cases.

(1) First consider the case that $w = \mathrm{top}_2(s)$. Note that this implies $\mathrm{CLvl}(s) = 2$ because the last operation of $\hat\rho$ is a collapse of level 2.

Furthermore, we claim that $\hat\rho$ does not visit $\mathrm{pop}_1(s)$. Heading for a contradiction, assume that $\hat\rho(i) = \mathrm{pop}_1(s)$ for some $i \in \mathrm{dom}(\hat\rho)$. Since $\hat\rho$ does not visit $\mathrm{pop}_2(s)$ between i and $m-1$, $\mathrm{top}_2(\hat\rho(m-1)) = w$ is only possible if $\mathrm{CLnk}(w) = |s| - 1$ (cf. Lemma 2.11). But then $\hat\rho\!\restriction_{[i,m]}$ is a return of $\mathrm{pop}_1(s)$ whence by definition $\hat\rho$ is a return of s. This contradicts our assumption.

We claim that $\hat\rho$ is a 1-loop plus a collapse operation: we have already seen that $\hat\rho$ does not visit any proper substack of s. Thus, it suffices to show that $\hat\rho$ reaches a stack with topmost word $\mathrm{pop}_1(w)$ only at positions where a return starts.

Let i be a position such that $\mathrm{top}_2(\hat\rho(i)) = \mathrm{pop}_1(w)$. Recall that $\mathrm{top}_2(s_{m-1}) = w$, $\mathrm{CLvl}(w) = 2$ and $\mathrm{CLnk}(w) \leq |s| - 1$. Since $\hat\rho$ does not visit $\mathrm{pop}_1(s)$, $|\hat\rho(i)| > |s|$ and we cannot restore $\mathrm{top}_1(w)$ by a push operation. Thus, there is some minimal position $m > j > i$ such that $|\hat\rho(j)| < |\hat\rho(i)|$. Lemma 4.14 implies that $\hat\rho\!\restriction_{[i,j]}$ is a return.

Thus, $\rho_1 := \hat\rho$ is of the form F2.

(2) For the other case, assume that $w < \mathrm{top}_2(s)$. Since $\hat\rho$ is not a return, we may apply the previous lemma. We conclude that there is some $i \in \mathrm{dom}(\hat\rho)$ such that $\rho_1 := \hat\rho\!\restriction_{[0,i]}$ is a 1-loop followed by a pop_1 or a collapse of level 1 and such that there is no $j > i$ such that $\hat\rho\!\restriction_{[i,j]}$ is a return. In order to show that $\hat\rho$ is of the form F3 we have to check the side conditions on the segments following in the decomposition of ρ. For this purpose set $\rho' := \rho\!\restriction_{[i,\mathrm{length}(\rho)]}$. By induction hypothesis ρ' decomposes as $\rho' = \rho_2 \circ \rho_3 \circ \cdots \circ \rho_n$ where the ρ_i satisfy the claim of the lemma.

Now, by definition of i, ρ' does not start with a return. Thus, ρ_2 is of one of the forms F2 or F3. But these forms require that there is some $j \geq 2$ such that ρ_j is of form F2 and for all $2 \leq k < j$, ρ_k is of the form F3. From this condition it follows directly that $\rho = \rho_1 \circ \rho' = \rho_1 \circ \rho_2 \circ \rho_3 \circ \cdots \circ \rho_n$ and ρ_1 is of the form F3. $\qquad\square$

For the other direction, assume that $\rho \in \mathrm{Runs}(s, s')$ decomposes as $\rho = \rho_1 \circ \rho_2 \circ \cdots \circ \rho_n$ where each ρ_i is of one of the forms F1-F3. Let $i_1 < i_2 < \cdots < i_k = n$ be the subsequence of subruns of the forms F1 or F2. Let s_{i_j} be the stack of the final configuration of ρ_{i_j}. A straightforward induction shows that $|s_{i_1}| > |s_{i_2}| > \cdots > |s_{i_k}|$ and that all stacks that occur after the final configuration of ρ_{i_j} and before the final configuration of $\rho_{i_{j+1}}$ have width at least $|s_{i_j}|$. Analogously, all stacks occurring before the final configuration of ρ_{i_1} have width

at least $|s|$. Thus, we conclude immediately that substacks of s' cannot be visited before the final configuration of ρ.

4.4. **Decompositions for Runs in $R^{\Downarrow}$ or $R^{\Uparrow}$.** The decomposition of runs witnessing that $(c_1, c_2) \in R^{\Leftarrow}$ or $(c_1, c_2) \in R^{\Rightarrow}$, respectively, turns out to be very useful for proving tree-automaticity of $R^{\Leftarrow}$ and $R^{\Rightarrow}$. We use similar characterisations for runs witnessing that certain pairs of configurations are contained in $R^{\Downarrow}$ or $R^{\Uparrow}$. The proofs of the following characterisations are straightforward inductions.

Lemma 4.18. *Let $c_1, c_2 \in \mathrm{Cnf}$ and ρ some run.*
(1) *ρ witnesses $(c_1, c_2) \in R^{\Downarrow}$ if and only if $\rho \in \mathrm{Runs}(c_1, c_2)$ and ρ decomposes as*
$$\rho = \lambda_1 \circ \rho_1 \circ \lambda_2 \circ \rho_2 \circ \cdots \circ \lambda_n \circ \rho_n$$
where each λ_i is a high loop and each ρ_i is a run performing exactly one transition which is pop_1 or a collapse of level 1.
(2) *Analogously, ρ witnesses $(c_1, c_2) \in R^{\Uparrow}$ if and only if $\rho \in \mathrm{Runs}(c_1, c_2)$ and ρ decomposes as $\rho = \lambda_0 \circ \rho_1 \circ \lambda_1 \circ \rho_2 \circ \cdots \circ \lambda_{n-1} \circ \rho_n \circ \lambda_n$ where the λ_i are high loops and each ρ_i performs exactly one push operation.*

4.5. **Computing Returns.** In this section, we prove that the existence of returns starting at a given stack s inductively depend on the returns starting at $\mathrm{pop}_1(s)$. Later we use this result in order to show that there is a similar dependence of loops starting in s from loops and returns starting in $\mathrm{pop}_1(s)$. Let us fix a CPS $\mathcal{S}$. For some word w occurring in a level 2 stack, let $w\downarrow_0$ denote the word where each level 2 link is replaced by 0, i.e., each $(\sigma, 2, i)$ is replaced by $(\sigma, 2, 0)$.

Definition 4.19. Set $\mathrm{Rt}(w) := \{(q, q') : \text{there is a return from } (q, w\downarrow_0 : w\downarrow_0) \text{ to } (q', w\downarrow_0)\}$. We also set $\mathrm{Rt}(s) := \mathrm{Rt}(\mathrm{top}_2(s))$.

The main goal of this section is the proof of the following proposition.

Proposition 4.20. *There is a finite automaton with $2^{|Q \times Q|}$ many states that computes $\mathrm{Rt}(w)$ on input $w\downarrow_0$.*

Remark 4.21. In fact, the automaton can be effectively constructed from a given CPS (cf. [11]). The same holds analogously for Proposition 4.29 and Corollary 4.38.

This proposition relies on the observation that returns of a stack s with topmost word w are composed by runs that are prefixed by s and by runs that are returns of stacks with topmost word $\mathrm{pop}_1(w)$. Furthermore, it relies on the observation that stacks with equal topmost word share the same returns. The reader who is not interested in the proof details may safely skip these and continue reading Section 4.6.

Lemma 4.22. *Let s be a stack with $|s| \geq 2$. There is a return from (q, s) to $(q', \mathrm{pop}_2(s))$ if and only if $(q, q') \in \mathrm{Rt}(\mathrm{top}_2(s))$.*

Proof. Let $w := \mathrm{top}_2(s)\downarrow_0$. A tedious but straightforward induction on the length of the return provides a transition-by-transition copy of a return starting at (q, s) to a return starting at $(q, w : w)$ and vice versa. $\square$

The following auxiliary lemmas prepare the decomposition of returns into subparts that are returns starting at stacks with smaller topmost words and subparts that are prefixed by the first stack of the return.

Lemma 4.23. *Let ρ be a return starting at some stack s with $\mathrm{top}_2(s) = w$. If ρ visits $\mathrm{pop}_1(s)$, let k be the first occurrence of $\mathrm{pop}_1(s)$ in ρ, otherwise let $k := \mathrm{length}(\rho)$. If $0 < i < k$ is a position such that $s \trianglelefteq \rho(i-1)$ and $\mathrm{top}_2(\rho(i)) = \mathrm{pop}_1(w)$ then there is some $i < j < k$ such that $\rho\!\restriction_{[i,j]}$ is a return.*

Proof. Since $i < k$, the stack at $\rho(i)$ is not $\mathrm{pop}_1(s)$. Thus, the stack at $\rho(i)$ is of the form $s' : \mathrm{pop}_1(w)$ with $s \trianglelefteq s'$, in particular $|s| < |\rho(i)|$. There is a minimal $i < j \leq k$ such that $|\rho(j)| < |\rho(i)|$.

If $j < \mathrm{length}(\rho)$, we conclude by application of Lemma 4.14. Otherwise, $j = \mathrm{length}(\rho)$ and ρ does not visit $\mathrm{pop}_1(s)$. Since ρ is a return, the operation at $j-1$ is pop_2 (whence $\rho\!\restriction_{[i,j]}$ is a return) or there is a nonempty word v such that $\mathrm{top}_2(\rho(j-1)) = wv$ and the operation at $j-1$ is a collapse of level 2. Since v was created between i and $j-1$ its topmost link points to a stack of width at least $|\rho(i)| - 1$ and we conclude again with Lemma 4.14 that $\rho\!\restriction_{[i,j]}$ is a return. $\square$

Lemma 4.24. *Let ρ be some run, s some stack with topmost word $w := \mathrm{top}_2(s)$ such that the following holds.*

(1) *$s \trianglelefteq \rho(0)$,*
(2) *$\rho(i) \not\prec s$ for all $0 \leq i \leq \mathrm{length}(\rho)$, and*
(3) *for all $0 < i \leq \mathrm{length}(\rho)$ such that $s \trianglelefteq \rho(i-1)$ and $w \not\sqsubseteq \mathrm{top}_2(\rho(i))$, there is some $i < j \leq \mathrm{length}(\rho)$ such that $\rho\!\restriction_{[i,j]}$ is a return.*

There is a well-defined sequence

$$0 := j_0 \leq i_1 < j_1 \leq i_2 < j_2 \leq \cdots \leq i_n < j_n \leq i_{n+1} := \mathrm{length}(\rho)$$

with the following properties.

(1) *For $1 \leq k \leq n+1$, $s \trianglelefteq \rho\!\restriction_{[j_{k-1}, i_k]}$.*
(2) *For each $1 \leq k \leq n$, there is a stack s_k with $\mathrm{top}_2(s_k) = \mathrm{pop}_1(w)$ such that $\rho\!\restriction_{[i_k+1, j_k]}$ is a return from s_k to $\mathrm{pop}_2(s_k)$.*
(3) *For all $1 \leq k \leq n$, $\mathrm{top}_2(\rho(i_k)) = w$ and the operation at i_k in ρ is a pop_1 or a collapse of level 1.*

Proof. The proof is by induction on $\mathrm{length}(\rho)$. If $s \trianglelefteq \rho$ (in particular, if $\mathrm{length}(\rho) = 0$), set $n := 0$ and we are done. Otherwise, let $j_0 := 0$ and let $i_1 \in \mathrm{dom}(\rho)$ be the minimal position such that $s \trianglelefteq \rho(i_1)$ but $s \not\trianglelefteq \rho(i_1 + 1)$. Since $\rho(i_1 + 1) \not\prec s$, the stack at $\rho(i_1 + 1)$ must be of the form $s' : \mathrm{pop}_1(w)$ for some s' such that $s \trianglelefteq s'$. This requires that the stack at $\rho(i_1)$ is $s' : w$ and the operation at i_1 is pop_1 or collapse of level 1. By assumption on ρ, there is some $i_1 + 1 < j_1 \leq \mathrm{length}(\rho)$ such that $\rho\!\restriction_{[i_1+1, j_1]}$ is a return. Thus, the stack at $\rho(j_1)$ is s' whence $s \trianglelefteq \rho(j_1)$. Thus, we can apply the induction hypothesis to $\rho\!\restriction_{[j_1, \mathrm{length}(\rho)]}$ which settles the claim. $\square$

The previous lemma allows to classify returns as follows.

Corollary 4.25. *Let ρ be a run starting in some stack s with topmost word $w = \mathrm{top}_2(s)$. ρ is a return from s to $\mathrm{pop}_2(s)$ that does not pass $\mathrm{pop}_1(s)$ if and only if $\rho \in \mathrm{Runs}(s, \mathrm{pop}_2(s))$*

and there is a uniquely defined sequence

$$0 := j_0 \leq i_1 < j_1 \leq i_2 < j_2 \leq \cdots \leq i_n < j_n \leq i_{n+1} := \mathrm{length}(\rho) - 1$$

with the following properties.

(1) *For* $1 \leq k \leq n+1$, $s \trianglelefteq \rho\!\restriction_{[j_{k-1}, i_k]}$.
(2) *For each* $1 \leq k \leq n$, *there is a stack* s_k *with* $\mathrm{top}_2(s_k) = \mathrm{pop}_1(w)$ *such that* $\rho\!\restriction_{[i_k+1, j_k]}$ *is a return from* s_k *to* $\mathrm{pop}_2(s_k)$.
(3) *For all* $1 \leq k \leq n$, $\mathrm{top}_2(\rho(i_k)) = w$ *and the operation at* i_k *in* ρ *is a* pop_1 *or a collapse of level 1.*
(4) *Either* w *is a proper prefix of* $\mathrm{top}_2(\rho(i_{n+1}))$ *and the operation at* i_{n+1} *is a collapse of level 2 or* w *is a prefix of* $\mathrm{top}_2(\rho(i_{n+1}))$ *and the operation at* i_{n+1} *is a* pop_2.

Proof. First assume that ρ is such a return. Due to Lemma 4.23, we can apply Lemma 4.24 to $\rho\!\restriction_{[0,\mathrm{length}(\rho)-1]}$. This gives immediately the first three items. The last item is a direct consequence of the definition of a return.

Now assume that ρ is a run from s to $\mathrm{pop}_2(s)$ that satisfies conditions (1)-(4). Heading for a contradiction assume that ρ visits $\mathrm{pop}_1(s)$. Due to (1) this happens at some position $i_k + 1 \leq j \leq j_k - 1$. Due to (2) we conclude that the width of the stack at $\rho(j_k)$ is smaller than the width at j. But this contradicts condition 1. because $s \trianglelefteq \rho(j_k)$.

For similar reasons ρ does not visit a substack of $\mathrm{pop}_2(s)$ before the final configuration. Thus, condition (4) implies that ρ is a return. $\qquad\square$

Corollary 4.26. *Let* ρ *be a run starting in some stack* s *with topmost word* $w = \mathrm{top}_2(s)$. ρ *is a return from* s *to* $\mathrm{pop}_2(s)$ *that passes* $\mathrm{pop}_1(s)$ *if and only if* $\rho \in \mathrm{Runs}(s, \mathrm{pop}_2(s))$ *and there is a uniquely defined sequence*

$$0 := j_0 \leq i_1 < j_1 \leq i_2 < j_2 \leq \cdots \leq i_n < j_n = \mathrm{length}(\rho)$$

with the following properties.

(1) *For* $1 \leq k \leq n$, $s \trianglelefteq \rho\!\restriction_{[j_{k-1}, i_k]}$.
(2) *For each* $1 \leq k \leq n$, *there is a stack* s_k *with* $\mathrm{top}_2(s_k) = \mathrm{pop}_1(w)$ *such that* $\rho\!\restriction_{[i_k+1, j_k]}$ *is a return from* s_k *to* $\mathrm{pop}_2(s_k)$.
(3) *For all* $1 \leq k \leq n$, $\mathrm{top}_2(\rho(i_k)) = w$ *and the operation at* i_k *in* ρ *is a* pop_1 *or a collapse of level 1.*

Proof. First assume that ρ is such a return. Let $0 < k < \mathrm{length}(\rho)$ be the first occurrence of $\mathrm{pop}_1(s)$ in ρ. Application of Lemma 4.24 to $\rho\!\restriction_{[0,k-1]}$ yields a decomposition into s-prefixed parts and returns (ending with an s-prefixed part). Finally, due to Lemma 4.13 $\rho\!\restriction_{[k,\mathrm{length}(\rho)]}$ is also a return.

Now assume that ρ is a run from s to $\mathrm{pop}_2(s)$ that satisfies conditions (1)-(3). As in the previous corollary, we conclude that ρ does not visit substacks of $\mathrm{pop}_2(s)$ before the final configuration. Due to condition (2), $\rho(i_n + 1) = \mathrm{pop}_1(s)$ and $\rho\!\restriction_{[i_n+1,\mathrm{length}(\rho)]}$ is a return whence ρ is also a return. $\qquad\square$

In the following corollary, we assume that $\mathrm{Rt}(\varepsilon) = \emptyset$.

Corollary 4.27. *For each stack* s, $\mathrm{Rt}(s)$ *is determined by* $\mathrm{Rt}(\mathrm{pop}_1(s)), \mathrm{Sym}(s)$ *and* $\mathrm{CLvl}(s)$.

Proof. Let w and w' be words such that $\mathrm{Sym}(w) = \mathrm{Sym}(w')$, $\mathrm{CLvl}(w) = \mathrm{CLvl}(w')$ and $\mathrm{Rt}(\mathrm{pop}_1(w)) = \mathrm{Rt}(\mathrm{pop}_1(w'))$. Fix a return ρ starting in (q_1, s) for $s := w\!\downarrow_0 \; : \; w\!\downarrow_0$ and

ending in $(q_2, w\!\downarrow_0)$. We have to prove that there is a return ρ' from (q_1, s') to $(q_2, w'\!\downarrow_0)$ for $s' := w'\!\downarrow_0 : w'\!\downarrow_0$.

The proof is by induction on $|w|$. Assume that ρ does not visit $\mathrm{pop}_1(s)$ and let

$$0 := j_0 \leq i_1 < j_1 \leq i_2 < j_2 \leq \cdots \leq i_n < j_n \leq i_{n+1} := \mathrm{length}(\rho) - 1$$

be the sequence according to Corollary 4.25. If ρ visits $\mathrm{pop}_1(s)$, use the sequence according to Corollary 4.26 and proceed analogously. For all $k \leq n+1$, $s \trianglelefteq \rho\!\restriction_{[j_{k-1}, i_k]}$. Set $\rho_k := \rho\!\restriction_{[j_{k-1}, i_k]}[s/s']$ (cf. Lemma 2.15). This settles the claim if $n = 0$.

For the case $n > 0$, note that $\rho\!\restriction_{[i_k+1, j_k]}$ is a return starting at some stack with topmost word $\mathrm{pop}_1(w)$. Thus, $\mathrm{Rt}(\mathrm{pop}_1(w)) = \mathrm{Rt}(\mathrm{pop}_1(w')) \neq \emptyset$ whence w and w' are words of length at least 2. Let δ_k be the transition connecting $\rho(i_k)$ with $\rho(i_k + 1)$. Note that δ_k is either a pop_1 transition or a collapse transition and $\mathrm{CLvl}(\rho(i_k)) = 1$. Note that $\mathrm{top}_2(\rho(i_k)) = w\!\downarrow_0$ whence $\mathrm{top}_2(\rho_k(\mathrm{length}(\rho_k))[s/s'] = w'\!\downarrow_0$. Hence, δ_k is applicable to the last configuration of ρ_k and leads to a configuration c_k with topmost word $\mathrm{pop}_1(w'\!\downarrow_0)$.

Recall that $\rho\!\restriction_{[i_k+1, j_k]}$ is a return starting at some stack with topmost word $\mathrm{pop}_1(w\!\downarrow_0)$. Since $\mathrm{Rt}(\mathrm{pop}_1(w)) = \mathrm{Rt}(\mathrm{pop}_1(w'))$, Lemma 4.22 provides a return ρ'_k from c_k to $\rho_{k+1}(0)$.

Finally, let γ be the transition connecting $\rho(i_{n+1})$ with $\rho(i_{n+1} + 1) = \rho(\mathrm{length}(\rho))$. γ connects the last configuration of ρ_{n+1} with $(q_2, w'\!\downarrow_0) = (q_2, \mathrm{pop}_2(s'))$: either γ is a pop_2 transition and $|\rho(i_{n+1})| = 2 = |\rho_{n+1}(\mathrm{length}(\rho_{n+1}))|$ or γ is a collapse transition and $\mathrm{top}_1(\rho(i_{n+1})) = (\sigma, 2, 1) = \mathrm{top}_1(\rho(i_{n+1}))[s/s'] = \mathrm{top}_1(\rho_{n+1}(\mathrm{length}(\rho_{n+1})))$. Thus,

$$\rho' := \rho_1 \circ \delta_1 \circ \rho'_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \delta_n \circ \rho'_n \circ \rho_{n+1} \circ \gamma$$

is a return from (q_1, s') to $(q_2, \mathrm{pop}_2(s'))$. □

Proposition 4.20, which states that $\mathrm{Rt}(w)$ can be computed by a finite automaton, is a direct corollary of the previous lemma: $\mathrm{Rt}(w)$ is a subset of $Q \times Q$. An automaton in state $\mathrm{Rt}(\mathrm{pop}_1(w))$ can change to state $\mathrm{Rt}(w)$ on input $\mathrm{Sym}(w)$ and $\mathrm{CLvl}(w)$.

4.6. **Computing (1-) Loops.** In analogy to the results of the previous section, we now investigate the existence of loops. We follow exactly the same ideas except for the fact that a loop of some stack s depends on the loops *and* returns of $\mathrm{pop}_1(s)$. At the end of this section, we provide a similar result for 1-loops.

Definition 4.28. Set $\mathrm{Lp}(w) := \{(q, q') : \text{there is a loop from } (q, w\!\downarrow_0) \text{ to } (q', w\!\downarrow_0)\}$. Similarly, let $\mathrm{hLp}(w)$ and $\ell\mathrm{Lp}(w)$ be the analogous sets for high loops and low loops, respectively. Set $1\mathrm{Lp}(w) := \{(q, q') : \text{there is a stack } s \text{ and a 1-loop from } (q, w\!\downarrow_0) \text{ to } (q', s : w\!\downarrow_0)\}$. We also set $\mathrm{Lp}(s) := \mathrm{Lp}(\mathrm{top}_2(s))$ and analogous for hLp, $\ell\mathrm{Lp}$ and $1\mathrm{Lp}$.

Extending the result of the previous section, our main goal is the following automaticity result for Lp and $1\mathrm{Lp}$.

Proposition 4.29. *There is a finite automaton $\mathcal{A}$ with $2^{|Q \times Q|} \cdot 2^{|Q \times Q|} \cdot |\Sigma|^2 \cdot 2$ many states that computes $\mathrm{Rt}(w)$, $\mathrm{Lp}(w)$, $\mathrm{hLp}(w)$, $\ell\mathrm{Lp}(w)$, $1\mathrm{Lp}(w)$, $\mathrm{Sym}(w)$ and $\mathrm{CLvl}(w)$ on input $w\!\downarrow_0$ (where w is a word occurring as topmost word of some stack s, i.e., for $w = \mathrm{top}_2(s)$).*

The reader who is not interested in the proof details, can safely skip the rest of this section and continue with Section 5. In the following, we mainly use the same arguments as in the return case, but we have to consider loops of the stack $\mathrm{pop}_1(s)$ because those occur as subruns of low loops of s. We omit proofs whenever they are analogous to the return case.

Lemma 4.30. *Let s be some stack. There is a loop from (q, s) to (q', s) if and only if $(q_1, q_2) \in \mathrm{Lp}(\mathrm{top}_2(s))$. The analogous statement holds for* hLp*,* $\ell\mathrm{Lp}$*, and* $1\mathrm{Lp}$*.*

The next step towards the proof of our main proposition is a characterisation of $\mathrm{Lp}(w)$ in terms of $\mathrm{Lp}(\mathrm{pop}_1(w))$ and $\mathrm{Rt}(\mathrm{pop}_1(w))$ analogously to the result of Corollary 4.25 for returns. We do this in the following three lemmas. First, we present a unique decomposition of loops into high and low loops. Afterwards, we characterise low loops and high loops.

Lemma 4.31. *Let λ be a loop from (q, s) to (q', s). λ is either a high loop or it has a unique decomposition as $\lambda = \lambda_0 \circ \lambda_1 \circ \lambda_2$ where λ_0 and λ_2 are high loops and λ_1 is a low loop.*

Proof. If λ is not a high loop, let $i \in \mathrm{dom}(\lambda)$ be the minimal position just before the first occurrence of $\mathrm{pop}_1(s)$ and $j \in \mathrm{dom}(\lambda)$ be the position directly after the last occurrence of $\mathrm{pop}_1(s)$. By definition, $\lambda\!\restriction_{[0,i]}$ and $\lambda\!\restriction_{[j,\mathrm{length}(\lambda)]}$ are high loops and $\lambda\!\restriction_{[i,j]}$ is a low loop. $\square$

Corollary 4.32. *The set $\mathrm{Lp}(s)$ can be computed from the sets $\mathrm{hLp}(s)$ and $\ell\mathrm{Lp}(s)$ via $\mathrm{Lp}(s) := \mathrm{hLp}(s) \cup \{(q, q') : \exists q_1, q_2(q, q_1) \in \mathrm{hLp}(s), (q_1, q_2) \in \ell\mathrm{Lp}(s),$ and $(q_2, q') \in \mathrm{hLp}(s)\}$.*

In the following, we first explain how low loops depend on the loops of smaller stacks, afterwards we explain how high loops depend on returns of smaller stacks.

Lemma 4.33. *Let λ be a low loop starting and ending in stack s. Then $\lambda\!\restriction_{[1,\mathrm{length}(\lambda)-1]}$ is a loop starting and ending in $\mathrm{pop}_1(s)$. The operation at 0 is a pop_1 or a collapse of level 1. The operation at $\mathrm{length}(\lambda) - 1$ is a push_σ where $\mathrm{top}_1(s) = \sigma \in \Sigma$.*

Proof. Note that each low loop λ satisfies $\lambda(1) = \mathrm{pop}_1(s) = \lambda(\mathrm{length}(\lambda) - 1)$. Since $\mathrm{pop}_2(\mathrm{pop}_1(s)) = \mathrm{pop}_2(s)$, it follows directly that the run in between satisfies the definition of a loop. $\square$

Corollary 4.34. *$\ell\mathrm{Lp}(s)$ depends on $\mathrm{Sym}(s), \mathrm{CLvl}(s), \mathrm{Sym}(\mathrm{pop}_1(s))$ and $\mathrm{Lp}(\mathrm{pop}_1(s))$.*

Note that $\mathrm{Sym}(s)$ and $\mathrm{CLvl}(s)$ determine whether the first transition of a low loop can be applied to the stack and that $\mathrm{Sym}(\mathrm{pop}_1(s))$ determines whether the last transition of a low loop can be applied.

In analogy to the return case, we provide a decomposition of high loops which shows that $\mathrm{hLp}(s)$ is determined by the returns of $\mathrm{pop}_1(s)$ and by the topmost symbol and link level of s.

Lemma 4.35 (cf. Corollary 4.25). *Let λ be some run starting in some stack s with topmost word $w = \mathrm{top}_2(s)$. λ is a high loop from s to s if and only if $\lambda \in \mathrm{Runs}(s, s)$ and there is a sequence $0 =: j_0 \leq i_1 < j_1 \leq i_2 < j_2 \leq \cdots \leq i_n < j_n \leq i_{n+1} := \mathrm{length}(\lambda)$ such that*

(1) *for $1 \leq k \leq n + 1$, $s \trianglelefteq \lambda\!\restriction_{[j_{k-1}, i_k]}$ and*

(2) *for each $1 \leq k \leq n$, there is a stack s_k with $\mathrm{top}_2(s_k) = \mathrm{pop}_1(w)$ such that $\lambda\!\restriction_{[i_k+1, j_k]}$ is a return of s_k.*

Corollary 4.36 (cf. Corollary 4.27). *$\mathrm{Rt}(\mathrm{pop}_1(s)), \mathrm{Sym}(s),$ and $\mathrm{CLvl}(s)$ determine $\mathrm{hLp}(s)$.*

We conclude the proof of Proposition 4.29 by showing that $1\mathrm{Lp}(s)$ is determined by $\mathrm{Rt}(s), \mathrm{Sym}(s)$ and $\mathrm{CLvl}(s)$.

Lemma 4.37. *Let ρ be some run. ρ is a 1-loop from some stack s to some stack s' with $\mathrm{top}_2(s) = \mathrm{top}_2(s')$ if and only if ρ is a run from some stack s to some stack s' with $\mathrm{top}_2(s) = \mathrm{top}_2(s')$ and ρ decomposes as*

$$\rho = \lambda_0 \circ \rho_1 \circ \lambda_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \lambda_n$$

where $s \trianglelefteq \lambda_i$ and each ρ_i is a return of a stack s_i with $\mathrm{top}_2(s_i) = \mathrm{top}_2(s)$.

Proof. First assume that ρ is a 1-loop. If $s \trianglelefteq \rho$, set $n = 0$. Otherwise, let j be minimal such that $s \ntrianglelefteq \rho(j+1)$. Set $\lambda_0 := \rho\!\restriction_{[0,j]}$. Note that λ_0 is s prefixed. Since ρ does not visit substacks of $\mathrm{pop}_2(s)$, $\mathrm{top}_2(\rho(j+1)) = \mathrm{pop}_1(\mathrm{top}_2(s))$. This implies that $\mathrm{top}_2(\rho(j)) = \mathrm{top}_2(s)$. By definition of a 1-loop, there is some $k > j + 1$ such that $\rho\!\restriction_{[j+1,k]}$ is a return. It follows directly that $\rho_1 := \rho\!\restriction_{[j,k]}$ is a return of the stack $s_1 := \rho(j)$ and $\mathrm{top}_2(s_1) = \mathrm{top}_2(s)$.

The first direction of the lemma follows by iterating this construction.

Now assume that ρ is a run from s to s' with $\mathrm{top}_2(s') = \mathrm{top}_2(s)$ that decomposes as specified above. We show that ρ is a 1-loop. ρ cannot visit a stack t with $|t| < |s|$ because then it especially visits such a stack at $\lambda_j(0)$ for some $1 \leq j \leq n$ which contradicts $s \trianglelefteq \lambda_j(0)$. Moreover, if it visits some stack t with $\mathrm{top}_2(t) = \mathrm{top}_2(\mathrm{pop}_1(s))$ then t occurs within some return ρ_j before the final configuration of ρ_j. Assume that this position is k, i.e., the stack at $\rho_j(k)$ is t. Since ρ_j is a return, there is a minimal k' such that the stack at $\rho_j(k')$ is narrower than t. Since $|s| \leq |\rho_j(k')| < |t|$, we conclude by Lemma 4.14 that $\rho\!\restriction_{[k,k']}$ is a return. $\qquad\square$

Corollary 4.38. $\mathrm{1Lp}(s)$ *is determined by* $\mathrm{Rt}(s)$, $\mathrm{Sym}(s)$ *and* $\mathrm{CLvl}(s)$.

Proof. By definition, it suffices to consider stacks of width 1. Thus, let w and w' be words with $\mathrm{Sym}(w) = \mathrm{Sym}(w')$, $\mathrm{CLvl}(w) = \mathrm{CLvl}(w')$ and $\mathrm{Rt}(w) = \mathrm{Rt}(w')$. Set $s = [w]$ and $s' = [w']$. If $s \trianglelefteq \rho$, then $s' \trianglelefteq \rho[s/s']$. Moreover if ρ' is a return from $(q, \hat{s} : w)$ to $(q', \hat{s})$ for some $\hat{s} \in \mathrm{Stck}(\Sigma)$ then there is a return from $(q, \hat{s}' : w')$ to $(q', \hat{s}')$ for all $\hat{s}' \in \mathrm{Stck}(\Sigma)$. Using the decomposition from the previous lemma, we can apply stack replacement and the existence of similar returns in order to show that $\mathrm{1Lp}(w) = \mathrm{1Lp}(w')$. $\qquad\square$

Proposition 4.29 now follows from Corollaries 4.36, 4.34, 4.32, 4.38 and from Proposition 4.20: we can store $\mathrm{Rt}(\mathrm{pop}_1(w))$, $\mathrm{Lp}(\mathrm{pop}_1(w))$, $\mathrm{Sym}(\mathrm{pop}_1(w))$, $\mathrm{Sym}(w)$ and $\mathrm{CLvl}(w)$ in $2 \cdot (2^{|Q \times Q|})^2 \cdot |\Sigma|^2$ many states and update the information during a transition reading the next letter of some word. Of course, $\mathrm{Sym}(\mathrm{pop}_1(w))$ and $\mathrm{Sym}(w)$ are only defined for words of length at least 2. Note that we are only interested in words occurring as topmost words of stacks. In such words, the combination $\mathrm{Sym}(w) = \bot$ and $\mathrm{Sym}(\mathrm{pop}_1(w)) \in \Sigma$ does never occur because $\bot$ is the bottom of stack symbol. Thus, some states from $2^{Q \times Q} \times 2^{Q \times Q} \times \Sigma \times \{\bot\} \times \{1, 2\}$ can be used to deal with the cases of words of length at most 1 separately.

5. REGULARITY OF THE REACHABILITY PREDICATE VIA Enc

Using the decomposition and automaticity results from the previous Section, we show that for each CPS $\mathcal{S}$ the encoding Enc translates the relation Reach (on all possible configurations, not only those occurring in the graph) into an automatic relation. Using the closure of CPS under products with finite automata, we then extend this result to all reachability relations Reach_L where L is some regular language over the transition labels.

5.1. **Connection between Milestones and Enc.**

We want to show the regularity of the regular reachability relations on collapsible pushdown graphs. As a preparation, we develop two correspondences between the nodes of the encoding of some stack s and the (generalised) milestones $\mathrm{MS}(s)$ ($\mathrm{GMS}(s)$, respectively). Taking a node $d \in \mathrm{Enc}(s)$ to the stack encoded by $\mathrm{Enc}(s)\!\restriction_{\{e : e \leq_{\mathrm{lex}} d\}}$ is an order isomorphism between $(\mathrm{Enc}(s), \leq_{\mathrm{lex}})$ and $(\mathrm{MS}(s), \ll)$. We

denote the image of a node d under this isomorphism by $\mathrm{LStck}(d, s)$. After the discussion of this isomorphism we develop another correspondence between nodes of $\mathrm{Enc}(s)$ and the generalised milestones of s. For each $d \in \mathrm{Enc}(s)$, we define the *induced general milestone* $\mathrm{IgM}(d, s) \in \mathrm{GMS}(s)$. This is the $\ll$-maximal $m \in \mathrm{GMS}(s)$ such that $\mathrm{LStck}(d, s) \trianglelefteq m$. Apparently, if $\mathrm{LStck}(d, s) \trianglelefteq s$, then the induced general milestone is s. This occurs if and only if d is in the rightmost path of $\mathrm{Enc}(s)$. In all other cases $\mathrm{IgM}(d, s)$ is the $\ll$-maximal generalised milestone of s whose topmost word is a copy of the topmost word of $\mathrm{LStck}(d, s)$. In this case $\mathrm{top}_2(\mathrm{LStck}(d, s))$ is not a prefix of $\mathrm{top}_2(s)$ and $\mathrm{IgM}(d, s)$ is not a milestone.

LStck and IgM are useful concepts for the analysis of runs from some milestone m of s to s due to the fact that any generalised milestone of s occurs in the image of LStck or in the image of IgM. Furthermore, the generalised milestones associated to some node d are closely connected to those associated to its successors. Assume that $d, d0, d1 \in \mathrm{Enc}(s)$. Then $\mathrm{LStck}(d0, s)$ is the $\ll$-successor of $\mathrm{LStck}(d, s)$, $\mathrm{LStck}(d1, s)$ is the $\ll$-successor of $\mathrm{IgM}(d0, s)$ and $\mathrm{IgM}(d, s) = \mathrm{IgM}(d1, s)$. If $m \ll \mathrm{LStck}(d, s)$, Corollary 4.10 implies that a run from m to s which does not visit substacks of m visits $\mathrm{LStck}(d, s), \mathrm{LStck}(d0, s), \mathrm{IgM}(d0, s)$, etc. It also implies that $\ll$-successors are connected by one operation followed by some loop.

We will later show that the combination of these observations is the key to the regularity of the reachability relations. A finite automaton may guess at each node d the last states in which $\mathrm{LStck}(d, s)$ and $\mathrm{IgM}(d, s)$ are visited by some run from some milestone m to s. Since the direct $\ll$-successors of these stacks are encoded in the successor or predecessor of d, the automaton can check that these guesses are locally consistent, i.e., that there is a single transitions followed by a loop connecting (q_1, s_1) to (q_2, s_2) where s_1 and s_2 are the generalised milestones represented by d and its successor (or predecessor). If the guess of the automaton is locally consistent at all nodes, it witnesses the existence of a run from m to s.

Definition 5.1. Let $T \in \mathbb{T}^{\mathrm{Enc}}$ be a tree and $d \in T \setminus \{\varepsilon\}$. Then the *left and downward closed tree of* d is $LT(d, T) := T{\restriction}_D$ where $D := \{d' \in T : d' \leq_{\mathrm{lex}} d\}$. We denote by $\mathrm{LStck}(d, T) := \pi_2(\mathrm{Dec}(LT(d, T)))$ the *left stack induced by* d. π_2 denotes the projection to the stack of $\mathrm{Dec}(LT(d, T))$. If T is clear from the context, we omit it.

Remark 5.2. We exclude the case $d = \varepsilon$ from the definition because the root encodes the state of the configuration and not a part of the stack. In order to simplify notation, we use the following conventions. Let $c = (q, s)$ be a configuration. For arbitrary $d \in \{0, 1\}^*$, we set $\mathrm{LStck}(d, s) := \mathrm{LStck}(0d, c) := \mathrm{LStck}(0d, \mathrm{Enc}(c))$.

Recall that $w := \mathrm{top}_2(\mathrm{LStck}(d, s)){\downarrow}_0$ is $\mathrm{top}_2(\mathrm{LStck}(d, s))$ where all level 2 links are set to 0. Due to the definition of the encoding, for every $d \in \mathrm{Enc}(s)$, w is determined by the path from the root to d: interpreting ε as empty word, the word along this path contains the pairs of stack symbols and collapse levels of the letters of $\mathrm{top}_2(\mathrm{LStck}(d, s))$. Since all level 2 links in w are 0, w is determined by this path. Thus, Proposition 4.29 implies that there is an automaton that calculates at each position $d \in \mathrm{Enc}(q, s)$ the existence of loops of $\mathrm{LStck}(d, \mathrm{Enc}(q, s))$ with given initial and final state.

$\mathrm{LStck}(d, \mathrm{Enc}(q, s))$ is a substack of s for all $d \in \mathrm{Enc}(q, s)$. This observation follows from Remark 3.4 combined with the fact that the left stack is induced by a lexicographically downward closed subset.

Lemma 5.3. *Let* $s \in \mathrm{Stck}(\Sigma)$. *For each* $d \in \mathrm{Enc}(s)$ *we have* $\mathrm{LStck}(d, s) \in \mathrm{MS}(s)$. *Furthermore, for each* $s' \in \mathrm{MS}(s)$ *there is some* $d \in \mathrm{Enc}(s)$ *such that* $s' = \mathrm{LStck}(d, s)$.

Proof. For the first claim, let $d \in \mathrm{Enc}(s)$. We know that $s_d := \mathrm{LStck}(d, s)$ is a substack of s. Recall that the path from the root to s_d encodes $\mathrm{top}_2(s_d)$. Furthermore, by definition of Enc, d corresponds to some maximal block b occurring in s in the following sense: there are 2-words s_1, s_2 and a word w such that $s = s_1 : (w \setminus b) : s_2$ and such that the subtree rooted at d encodes b. Moreover, d encodes the first letter of b, i.e., if b is a τ-block, then the path from the root to d encodes $w\tau$.

Note that by maximality of b, the greatest common prefix of the last word of s_1 and the first word of $w \setminus b$ is a prefix of $w\tau$. Since the elements that are lexicographically smaller than d encode the blocks to the left of b, one sees that $s_d = s_1 : w\tau$. Setting $k := |s_d|$, we conclude that s_d is a substack of s such that the greatest common prefix of the $(k-1)$-st and the k-th word of s is a prefix of $\mathrm{top}_2(s_d)$. Recall that this matches exactly the definition of a milestone of s. Thus, s_d is a milestone of s and we completed the proof of the first claim.

Now we turn to the second claim. The fact that every milestone $s' \in \mathrm{MS}(s)$ is indeed represented by some node of $\mathrm{Enc}(s)$ can be seen by induction on the block structure of s'. Assume that $s' \in \mathrm{MS}(s)$ and that s' decomposes as $s' = b_0 : b_1 : \cdots : b_{m-1} : b'_m$ into maximal blocks. We claim that s then decomposes as $s = b_0 : b_1 : \cdots : b_{m-1} : b_m : \cdots : b_n$ into maximal blocks. In order to verify this claim, we have to prove that b_{m-1} cannot be the initial segment of a larger block $b_{m-1} : b_m$ in s. Note that if b'_m only contains one letter, then by definition of a milestone the last word of b_{m-1} and the first word occurring in s after b_{m-1}, which is the first word of b_m, can only have a common prefix of length at most 1. Hence, their composition does not form a block. Otherwise, the first word of b'_m contains two letters which do not coincide with the first two letters of the words in b_{m-1}. Since this word is by definition a prefix of the first word in b_m, we can conclude again that $b_{m-1} : b_m$ does not form a block.

Note that all words in the blocks b_i for $1 \leq i \leq n$ and in the block b'_m share the same first letter which is encoded at the position ε in $\mathrm{Enc}(s)$ and in $\mathrm{Enc}(s')$. By the definition of $\mathrm{Enc}(s)$ the blockline induced by b_i is encoded in the subtree rooted at $1^i 0$ in $\mathrm{Enc}(s)$. For $i < m$ the same holds in $\mathrm{Enc}(s')$. We set $d := 1^m$. Note that $\mathrm{Enc}(s')$ and $\mathrm{Enc}(s)$ coincide on all elements that are lexicographically smaller than d (because these elements encode the blocks $b_1 : b_2 : \ldots b_{m-1}$.

Now, we distinguish the following cases.

(1) Assume that $b'_m = [\tau]$ for $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Then the block b'_m consists of only one letter. In this case d is the lexicographically largest element of $\mathrm{Enc}(s')$ whence $s' = \mathrm{LStck}(d, \mathrm{Enc}(s')) = \mathrm{LStck}(d, \mathrm{Enc}(s))$.

(2) Otherwise, there is a $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ such that
$$b_m = \tau \setminus (c_0 : c_1 : \cdots : c_{m'-1} : c_{m'} : \cdots : c_{n'}) \text{ and}$$
$$b'_m = \tau \setminus (c_0 : c_1 : \cdots : c_{m'-1} : c'_{m'})$$
for some $m' \leq n'$ such that $c_0 : c_1 : \cdots : c_{n'}$ are the maximal blocks of the blockline induced by b_m and $c_0 : c_1 : \ldots c_{m'-1} : c'_{m'}$ are the maximal blocks of the blockline induced by b'_m. Now, $c_1 : c_2 : \cdots : c_{m'-1}$ are encoded in the subtrees rooted at $d01^i 0$ for $0 \leq i \leq m'-1$ in $\mathrm{Enc}(s)$ as well as in $\mathrm{Enc}(s')$. $c_{m'+1} : c_{m'+2} : \cdots : c_{n'}$ is encoded in the subtree rooted at $d01^{m'+1}$ in $\mathrm{Enc}(s)$ and these elements are all lexicographically larger than $d01^{m'} 0$. Hence, we can set $d' := d01^{m'}$ and repeat this case distinction on $d', c'_{m'}$ and $c_{m'}$ instead of d, b'_m and b_m.

Since s' is finite, by repeated application of the case distinction, we will eventually end up in the first case where we find a $d \in \mathrm{Enc}(s)$ such that $s' = \mathrm{LStck}(d, \mathrm{Enc}(s))$. $\square$

The next lemma states the tight connection between milestones of a stack (with substack relation) and elements in the encoding of this stack (with lexicographic order).

Lemma 5.4. $\mathrm{LStck}(\cdot, s)$ *is an isomorphism between* $(dom(\mathrm{Enc}(s)), \leq_{\mathrm{lex}})$ *and* $(\mathrm{MS}(s), \ll)$.

Proof. If the successor of d in lexicographic order is $d0$, then the left stack of the latter extends the former by just one letter. Otherwise, the left and downward closed tree of the successor of d contains more elements ending in 1, whence it encodes a stack of larger width. Since each left and downward closed tree induces a milestone, it follows that g is an order isomorphism. $\square$

Recall that by Corollary 4.10, each run to a configuration (q, s) visits the milestones of s in the order given by the substack relation. With the previous lemma, this translates into the fact that the left stacks induced by the elements of $\mathrm{Enc}(q, s)$ are visited by the run in lexicographical order of the elements of $\mathrm{Enc}(q, s)$.

Beside the tight correspondence of milestones and nodes of the encoding, there is another correspondence between generalised milestones and nodes. Using both correspondences, each generalised milestone is represented by a node of the encoding. The following definition describes the second correspondence. Recall that $\leq$ denotes the prefix relation on trees.

Definition 5.5. Let $T \in \mathbb{T}^{\mathrm{Enc}}$ be the encoding of a configuration. Let $d \in T \setminus \{\varepsilon\}$. Set $D := LT(d, T) \cup \{d' \in T : d \leq d'\}$. Let $\mathrm{exLStck}(d, T) := \pi_2(\mathrm{Dec}(T{\restriction}_D))$ where $\pi_2(c)$ is the projection to the stack of the configuration c.

By case distinction on the rightmost branch of T, we define the *generalised milestone induced by d* as follows.

(1) If d is in the rightmost branch of T, then $\mathrm{IgM}(d, T) := \mathrm{exLStck}(d, T) = \pi_2(\mathrm{Dec}(T))$,
(2) otherwise, set $\mathrm{IgM}(d, T) := \mathrm{exLStck}(d, T) : \mathrm{top}_2(\mathrm{LStck}(d, T))$.

Remark 5.6. As the name indicates, $\mathrm{IgM}(d, T)$ is a always a generalised milestone of $\mathrm{Dec}(T)$. For some d in the rightmost branch of T this holds trivially. For all $d \in T$ that are not in the rightmost branch, note the following:

- If $d1 \in T$, then $\mathrm{IgM}(d, T) = \mathrm{IgM}(d1, T)$.
- If d is a leaf, then $\mathrm{IgM}(d, T) = \mathrm{clone}_2(\mathrm{LStck}(d, T))$. Since $\mathrm{LStck}(d, T)$ is the maximal milestone of $\mathrm{Dec}(T)$ of width $|\mathrm{LStck}(d, T)|$, $\mathrm{IgM}(d, T)$ is a generalised milestone of $\mathrm{Dec}(T)$ (since d is not in the rightmost branch, $|\mathrm{Dec}(T)| > |\mathrm{LStck}(d, T)|$).
- If $d, e \in T$ such that $e = d0$ and $d1 \notin T$, then $\mathrm{IgM}(d, T) = \mathrm{pop}_1(\mathrm{IgM}(e, T))$.
- If $d, e \in T$ such that $e = d0$ and $d1 \in T$, then $\mathrm{LStck}(d1, T) = \mathrm{pop}_1(\mathrm{IgM}(e, T))$.

By induction from the leaves to the inner nodes one concludes that $\mathrm{IgM}(d, T)$ is a generalised milestone of $\mathrm{Dec}(T)$ for each $d \in T$. One also shows that, for each generalised milestone m of $\mathrm{Dec}(T)$ that is not a milestone, there is some d such that $m = \mathrm{IgM}(d, T)$ as follows. For $s = w_1 : w_2 : \cdots : w_n$, let w_{i_k} be the k-th word occurring in s such that w_{i_k} is not a prefix of w_{i_k+1}. Then $w_1 : w_2 : \cdots : w_{i_k-1} : w_{i_k} = \mathrm{LStck}(d, T)$ for the k-th leaf d and $\mathrm{IgM}(d, T) = w_1 : w_2 : \cdots : w_{i_k-1} : w_{i_k} : w_{i_k}$ is the k-th generalised milestone of the form $w_1 : w_2 : \cdots : w_{j-1} : w_j : w_j$ for some $j \leq n$ which is not a milestone. Now one can show the following. If $\mathrm{IgM}(d, T)$ is not a milestone (but a generalised one) and $\mathrm{pop}_1(\mathrm{IgM}(d, T))$ is not a milestone, then $\mathrm{pop}_1(\mathrm{IgM}(d, T))$ is a generalised milestone, there is a node e such that

$d = e01^k$ for some $k \in \mathbb{N}$, and $\text{pop}_1(\text{IgM}(d, T)) = \text{IgM}(e, T)$. Apparently, every generalised milestone that is not a milestone is of the form $\text{pop}_1^k(w_1 : w_2 : \cdots : w_{i-1} : w_i : w_i)$ for some $i \leq n$. Thus, this proves the claim.

5.2. **Tree-Automaticity of Reachability.** In this section we show that the reachability relation Reach is automatic via Enc. In the next section we extend this result to the regular reachability predicates. Recall that due to Remark 4.3, a proof of the regularity of the relations $R^{\Leftarrow}, R^{\Downarrow}, R^{\Uparrow}$ and $R^{\Rightarrow}$ implies the regularity of Reach.

5.2.1. *Regularity of the Relation $R^{\Leftarrow}$.* Recall that a pair of configurations (c_1, c_2) is in $R^{\Leftarrow}$ if and only if they are connected by a run ρ that decomposes as explained in Lemma 4.17. In the Appendix B we construct an automaton $\mathcal{A}_{R^{\Leftarrow}}$ recognising $R^{\Leftarrow}$ based on the following idea. $\mathcal{A}_{R^{\Leftarrow}}$ guesses the decomposition according to Lemma 4.17 and identifies a node representing the stack reached after each part of the decomposition. $\mathcal{A}_{R^{\Leftarrow}}$ labels this node by the initial and final state of the segment of the decomposition starting at the corresponding stack and checks whether the labelling of all the representatives fit together. The fact that $\mathcal{A}_{R^{\Leftarrow}}$ can check the correctness of its guess relies heavily on the computability of the returns and 1-loops of the stack $\text{LStck}(d, \text{Enc}(q, s))$ along the path from the root of $\text{Enc}(q, s)$ to d. We next explain how $\mathcal{A}_{R^{\Leftarrow}}$ processes two configurations $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$. Let us assume that $s_2 = \text{pop}_2^k(s_1)$ and let ρ be a run from c_1 to c_2 witnessing $(c_1, c_2) \in R^{\Leftarrow}$. Let us first assume that ρ is a sequence of returns $\rho = \rho_1 \circ \cdots \circ \rho_k$. Due to the special form of s_2, $\text{Enc}(s_1)$ and $\text{Enc}(s_2)$ agree on the domain of $\text{Enc}(s_2)$. Moreover $\text{Enc}(s_1)$ extends $\text{Enc}(s_2)$ by k paths to nodes $d_1, \ldots, d_k$ such that each d_i does not have a 0-successor. Moreover, all d_i are lexicographically larger than all nodes in $\text{Enc}(s_2)$. There is a close correspondence between the ρ_i and the d_i: $\text{LStck}(d_i, \text{Enc}(s_1)) = \text{pop}_2^{k-i}(s_1)$ and ρ_i starts in $\text{LStck}(d_{k+1-i}, \text{Enc}(s_1))$ and ends in $\text{LStck}(d_{k-i}, \text{Enc}(s_1))$ (where we set d_0 to be the rightmost leaf of $\text{Enc}(s_2)$). $\mathcal{A}_{R^{\Leftarrow}}$ guesses the existence of the run ρ as follows: at first, it checks that $\text{Enc}(s_1)$ and $\text{Enc}(s_2)$ agree on the domain of $\text{Enc}(s_2)$. Secondly, along the common prefix it propagates the initial and final state of ρ, i.e., the states q_1 an q_2 and it computes at each node d the possible returns at the corresponding milestone. Now assume that at some node e there starts a left and a right branch such that the left branch is a prefix of $d_1, \ldots, d_i$ and the right branch is a prefix of $d_{i+1}, \ldots, d_k$. At this position the automaton guesses that $\text{LStck}(d_{i+1}, \text{Enc}(c_1))$ and $\text{LStck}(d_i, \text{Enc}(c_1))$ are connected via a return and guesses the initial and final state q_i, q_e. Now the automaton propagates along the left branch starting at e the state information q_e and q_2 (trying to find a run from $(q_e, \text{LStck}(d_i, \text{Enc}(c_1))$ to c_2) and along the right branch the information $q_1, (R, q_i, q_e)$ (trying to find a run from c_1 to $c' = (q_i, \text{LStck}(d_{i+1}, \text{Enc}(c_1)))$ such that there is a return starting in c' and ending in state q_e). Doing the same at each splitting points of the prefixes of the $d_1, \ldots, d_k$, the automaton finally reaches each node d_i with a tuple (R, q_i, q_i') of states such that it has to check whether there is a return starting in $(q_i, \text{LStck}(d_i, \text{Enc}(c_1)))$ and ending in state q_i'. But since the returns of $\text{LStck}(d_i, \text{Enc}(c_1))$ are computable with an automaton reading the path to d_i, this can be easily checked with a tree-automaton.

The case that ρ decomposes as a sequence of returns is the easiest one because all parts of the decomposition then start and end in stacks that are milestones of c_1. Now assume that in the decomposition of ρ according to Lemma 4.17 1-loops followed by pop_1 or collapse operations occur. For simplicity of the explanation assume that $\rho = \lambda_1 \circ \lambda_2 \circ \lambda_3$ such that

λ_1 and λ_3 decompose as sequences of returns and $\lambda_2 = \rho_1 \circ \cdots \circ \rho_n$ decomposes such that ρ_i is a sequence of 1-loops followed by pop_1 and collapse where only ρ_n ends in a collapse of level 2. Using the notation from the previous case, we find d_i and d_j ($i < j$) among $d_1, \ldots, d_k$ such that λ_2 starts in stack $\mathrm{LStck}(d_j, \mathrm{Enc}(c_1))$ and ends in $\mathrm{LStck}(d_i, \mathrm{Enc}(c_1))$. We would like to treat this case similar to the return case, but note that the final stacks of $\rho_1, \ldots, \rho_{n-1}$ do not necessarily appear as milestones of s_1: in general, these stack can be wider than s_1! The key observation that allows to represent these stack by certain milestones is the following: let $t_0 := \mathrm{LStck}(d_j, \mathrm{Enc}(c_1))$ be the stack in which ρ_1 starts. By definition of a 1-loop followed by a pop_1 operation, the final stack of ρ_1 is some stack t_1 such that $\mathrm{top}_2(t_1) = \mathrm{top}_2(\mathrm{pop}_1(t_0))$. In particular, each level 2 collapse link in $\mathrm{top}_2(t_1)$ points to the same substack as the corresponding element of $\mathrm{top}_2(t_0)$. If e_1 is the unique ancestor of d_j such that $d_j = e_1 0 1^x$ for some $x \in \mathbb{N}$, one sees that t_1 and $\mathrm{LStck}(e_1, \mathrm{Enc}(c_1))$ agree on their topmost words including the targets of their level 2 collapse links. Note that λ_2 (modulo widening the stack during 1-loops) basically performs pop_1/collapse of level 1 on $\mathrm{top}_2(t_0)$ and finally a collapse of level 2 on a prefix of $\mathrm{top}_2(t_0)$. Thus, with respect to the stack operations induced by the run $\rho_2 \circ \cdots \circ \rho_n$, $m_1 := \mathrm{LStck}(e_1, \mathrm{Enc}(c_1))$ and t_1 agree and e_1 can serve as representative of t_1. Iterating this argument, we find nodes $e_2, \ldots, e_{n-1}$ such that $m_x := \mathrm{LStck}(e_x, \mathrm{Enc}(c_1))$ agrees with the final stack t_x of ρ_x on the topmost word including the targets of all level 2 collapse links for all $x < n$. Moreover, $\mathrm{collapse}(t_{n-1}) = \mathrm{collapse}(m_{n-1}) = \mathrm{LStck}(d_i, \mathrm{Enc}(c_1))$. The automaton $\mathcal{A}_{R^{\Leftarrow}}$ uses this observation as follows. Processing the encodings of c_1 and c_2 from the root to the leaves it arrives at the greatest common prefix f of d_i and d_j with a guess (q_i, q_e) of the initial and final state of the subrun of ρ which decomposes as $\lambda_1' \circ \lambda_2 \circ \lambda_3'$ where λ_2 is defined as before and λ_1' is a suffix of λ_1 and λ_3' a prefix of λ_3. Now the automaton guesses the initial and final state (q_1, q_2) of the run λ_2 and propagates along the left branch the guess that λ_3' is a run from q_2 to q_e and along the right branch a guess of the form (C, q_1, q_2) meaning that the last part of $\lambda_1' \circ \lambda_2$ is a 1-loop followed by collapse (hence the "C") from state q_1 to state q_2. It then nondeterministically guesses the path to d_j and updates a guess (X, q_1, q_2) with $X \in \{C, P\}$ at node e_x as follows. From e_x ($x \in \mathbb{N}$) on it propagates a guess (P, q_1, q_2') towards d_j meaning that the part of λ_2 connecting the corresponding nodes of e_{x-1} and e_x is a 1-loop followed by a pop_1/collapse of level 1 ("P" for pop) from state q_1 to state q_2'. It verifies the compatibility of the guess (X, q_1, q_2) at e_x and the guess (P, q_1, q_2') by checking that for any stack with the same topmost word as $m_x = \mathrm{LStck}(e_x, \mathrm{Enc}(c_1))$ there is a 1-loop followed by an operation induced by X from state q_2' to state q_1 (induced operation means collapse of level 2 if $X = C$ and pop_1 or collapse of level 1 if $X = P$). This way the automaton reaches d_j with a guess (X, q_1, q_2) and needs to verify that there is a 1-loop followed by an operation induced by X starting in (q_1, t_0) and ending in q_2. Since an automaton can keep track of the possible 1-loops at each node of the tree this is possible. The automaton can guess states and successfully verify its assumptions if and only if there is a run from $\mathrm{LStck}(d_j, \mathrm{Enc}(c_1))$ to $\mathrm{LStck}(d_i, \mathrm{Enc}(c_1))$ with initial and final state as guessed at the node f that decomposes into 1-loops followed by one pop_1 or collapse operation each. Combining this idea with the verification of guesses on parts of the run ρ that are returns, the tree-automaton accepts the encodings of two configurations if and only if this pair of configurations is in $R^{\Leftarrow}$.

In Appendix B we show that the described automaton works correctly and can be implemented with exponentially many states in the number of states of the collapsible pushdown system. Thus, we obtain the following result.

Lemma 5.7. *There are two polynomials p_1, p_2 such that the following holds. Let $\mathcal{S}$ be a level 2 collapsible pushdown system with stack alphabet Σ and state space Q. There is an automaton with $p_1(|\Sigma|) \cdot \exp(p_2(|Q|))$ many states that accepts the convolution of two trees if and only if this convolution is of the form $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ for c_1, c_2 configurations with $(c_1, c_2) \in R^{\Leftarrow}$.*

5.2.2. *Regularity of the Relation $R^{\Downarrow}$.* Recall that the relation $R^{\Downarrow}$ from Definition 4.2 contains pairs (c_1, c_2) if $c_2 = \mathrm{pop}_1{}^m(c_1)$ and there is some run from c_1 to c_2 not visiting any substack of c_2 before its final configuration. A simple induction on the blocks in c_1 and c_2 yields the following characterisation of $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$. For $d \in \{0,1\}^*$, $|d|_0$ denotes the number of 0's in d.

Lemma 5.8. *Let s_1, s_2 be stacks and d the rightmost leaf of $\mathrm{Enc}(s_2)$. $s_2 = \mathrm{pop}_1{}^m(s_1)$ for some $m \in \mathbb{N}$ if and only if $\mathrm{Enc}(s_1) \otimes \mathrm{Enc}(s_2)$ is of one of the following forms.*
1. *If $d \in \mathrm{Enc}(s_1)$, then $dom(\mathrm{Enc}(s_1)) = dom(\mathrm{Enc}(s_2)) \cup \{d0^k : k \leq m\}$ and $\mathrm{Enc}(s_1)$ and $\mathrm{Enc}(s_2)$ agree on $dom(\mathrm{Enc}(s_2))$.*
2. *If $d \notin \mathrm{Enc}(s_1)$, then $d \in \{0,1\}^*1$. Let $c \in \{0,1\}^*$ be the predecessor of d. There is some $e \in \{0,1\}^*$ such that ce is the rightmost leaf of $\mathrm{Enc}(s_1)$. Then $|e|_0 = m$,*
$$dom(\mathrm{Enc}(s_1)) = \big(dom(\mathrm{Enc}(s_2)) \cup \{x : c \leq x \leq e\}\big) \setminus \{d\}$$
   and $\mathrm{Enc}(s_1)$ and $\mathrm{Enc}(s_2)$ agree on $dom(\mathrm{Enc}(s_2)) \setminus \{d\}$. Moreover, there exists some $f \in \{0,1\}^*1$ such that
   - *$ce = f0^k$ for some $k \leq m$ and*
   - *$dom(\mathrm{Enc}(s_1)) \setminus dom(\mathrm{Enc}(s_2)) = \{f \leq x \leq ce\}$.*

In Remark 5.2 we pointed out that the path from the root to the rightmost leaf of $\mathrm{Enc}(s_1)$ encodes $\mathrm{top}_2(s_1)$. If the first case of the characterisation applies, then the k-th predecessor x_k of the rightmost leaf of $\mathrm{Enc}(s_1)$ satisfies $\mathrm{LStck}(x_k, s_1) = \mathrm{pop}_1{}^k(s_1)$ for all $k < m$. If the second case applies, for all $c \leq x \leq e$ with $|x|_0 - |c|_0 = k \leq m$, the path to x encodes $w_k := \mathrm{top}_2(\mathrm{pop}_1{}^{m-k}(s_1))$ and $\mathrm{top}_2(\mathrm{LStck}(x, s_1)) = w_k$. Thus, the elements on the path from d (or c, respectively) to the rightmost leaf of $\mathrm{Enc}(s_1)$ may serve as representatives of the stacks that a run from s_1 to s_2 passes.

Recall the decomposition into high loops of witnessing runs for $(c_1, c_2) \in R^{\Downarrow}$ from Lemma 4.18. We describe informally the automaton $\mathcal{A}_{R^{\Downarrow}}$ that recognises the relation $R^{\Downarrow}$. $\mathcal{A}_{R^{\Downarrow}}$ guesses the path to the rightmost leaf of $\mathrm{Enc}(c_2)$ and keeps track of $\mathrm{hLp}(d)$ at each node d on this path. Each node d on the path from the rightmost leaf of $\mathrm{Enc}(c_2)$ to the rightmost leaf of $\mathrm{Enc}(c_1)$ is labelled by the state of c_1, by $\mathrm{Rt}(\mathrm{pop}_1(\mathrm{LStck}(d, c_1)))$, by $\mathrm{Sym}(\mathrm{LStck}(d, c_1))$, by $\mathrm{CLvl}(\mathrm{LStck}(d, c_1))$ and by a guess $q_d \in Q$ of a final state of some run from $\mathrm{Enc}(c_1)$ to $\mathrm{pop}_1{}^{m-k}(s_1)$ for k appropriate such that $\mathrm{top}_2(\mathrm{LStck}(d, c_1)) = \mathrm{top}_2(\mathrm{pop}_1{}^{m-k}(c_1))$. Recall that such a state determines the set $\mathrm{hLp}(\mathrm{LStck}(d, c_1))$. The automaton can verify that the guesses of the q_d are consistent in the following sense. If it has labelled some node d with a state q_d such that there is a run from c_1 to $(q_d, \mathrm{pop}_1{}^{m-k}(s_1))$, then there is also a run from c_1 to $(q_c, \mathrm{pop}_1{}^{m-k-(1-i)}(s_1))$ for c the node such that $ci = d$. If $i = 1$, $q_c = q_d$ and if $i = 0$ then the run to $s' := \mathrm{pop}_1{}^{m-k-(1-i)}(s_1)$ is extended by a high loop of s' followed by a pop_1 or collapse of level 1. Since the automaton "knows" the possible high loops, the topmost symbol and the link level of the stack, this check is trivial. Since it also stores the state of c_1, $\mathcal{A}_{R^{\Downarrow}}$ can verify that its guess at the rightmost leaf of $\mathrm{Enc}(c_1)$ is a state q such that

there is a high loop starting in c_1 and ending in state q. We postpone the formal definition of $\mathcal{A}_{R^{\Downarrow}}$ and the proof of the following lemma to Appendix C.

Lemma 5.9. *There are polynomials p_1, p_2 such that the following holds. Let $\mathcal{S}$ be a level 2 collapsible pushdown system with stack alphabet Σ and state space Q. There is an automaton with $p_1(|\Sigma|) \cdot \exp(p_2(|Q|))$ many states that accepts the convolution of two trees if and only if this convolution is of the form $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ for c_1, c_2 configurations with $(c_1, c_2) \in R^{\Downarrow}$.*

5.2.3. *Regularity of the Relation $R^{\Uparrow}$.* Recall that the relation $R^{\Uparrow}$ is in some sense the backward version of the relation $R^{\Downarrow}$. $(c_1, c_2) \in R^{\Downarrow}$ holds if there is a sequence of high loops, pop_1 and collapse of level 1 connecting c_1 with c_2. Analogously, $(c_1, c_2) \in R^{\Uparrow}$ holds if there is a sequence of high loops and $\mathrm{push}_{\sigma,l}$ operations that generates c_2 from c_1. Since $(c_1, c_2) \in R^{\Uparrow}$ implies that $c_1 = \mathrm{pop}_1{}^k(c_2)$ for some $k \in \mathbb{N}$, Lemma 5.8 applies analogously. There is just one further condition: if $(c_1, c_2) \in R^{\Uparrow}$ and $\mathrm{top}_2(c_1) < \mathrm{top}_2(c_2) \sqcap \mathrm{top}_2(\mathrm{pop}_2(c_2))$, then there is some word w such that $\mathrm{top}_2(c_2) \sqcap \mathrm{top}_2(\mathrm{pop}_2(c_2)) = \mathrm{top}_2(c_1)w$ and w only contains links of level 1 (otherwise, the stack of c_2 cannot be generated from the stack of c_1 without passing $\mathrm{pop}_2(c_1)$).

The automaton $\mathcal{A}_{R^{\Uparrow}}$ recognising the relation $R^{\Uparrow}$ via Enc does the following. Due to Lemma 5.8, the path from the rightmost leaf of $\mathrm{Enc}(c_1)$ to the rightmost leaf of $\mathrm{Enc}(c_2)$ has the following form: For each $|\mathrm{top}_2(c_1)| \le m \le |\mathrm{top}_2(c_2)|$ it contains nodes $d, d1, \ldots, d1^{k_m}$ with $|d|_0 = m$ such that $\mathrm{top}_2(\mathrm{LStck}(d, \mathrm{Enc}(c_2))$ is the prefix of $\mathrm{top}_2(c_2)$ of length m. Recall that $\mathcal{A}_{R^{\Downarrow}}$ tries to label d with a state q_e and $e = d1^{k_m}0$ with a state q'_e such that q'_e and q_e are connected by a high loop of $\mathrm{LStck}(e, \mathrm{Enc}(c_1))$ plus a pop_1 or collapse of level 1. Since $\mathcal{A}_{R^{\Uparrow}}$ works in the other direction, it labels d with a state q_i and $d1^{k_m}0$ with a state q'_i such that q_i and q'_i are connected by a high loop of $\mathrm{LStck}(d, \mathrm{Enc}(c_2))$ followed by a $\mathrm{push}_{\sigma,l}$. Furthermore, it checks that $l = 1$ as long as the path to d encodes a proper prefix of $\mathrm{top}_2(c_2) \sqcap \mathrm{top}_2(\mathrm{pop}_2(c_2))$ which is not a prefix of $\mathrm{top}_2(c_1)$. With these remarks, the formal construction of $\mathcal{A}_{R^{\Uparrow}}$ from $\mathcal{A}_{R^{\Downarrow}}$ (cf. Appendix C) is left to the reader.

Lemma 5.10. *There are two polynomials p_1, p_2 such that the following holds. Let $\mathcal{S}$ be a level 2 collapsible pushdown system with stack alphabet Σ and state space Q. There is an automaton with $p_1(|\Sigma|) \cdot \exp(p_2(|Q|))$ many states that accepts the convolution of two trees if and only if this convolution is of the form $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ for c_1, c_2 configurations with $(c_1, c_2) \in R^{\Uparrow}$.*

5.2.4. *Regularity of the Relation $R^{\Rightarrow}$.* Given a CPS $\mathcal{S} = (Q, \Sigma, \Gamma, \Delta_{\mathcal{S}}, q_0)$, we define an automaton $\mathcal{A}_{R^{\Rightarrow}}$ that recognises the relation $R^{\Rightarrow}$ in the following sense. Given configurations $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$, $\mathcal{A}_{R^{\Rightarrow}}$ accepts $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ if and only if $s_1 = \mathrm{pop}_2{}^k(s_2)$ for some $k \in \mathbb{N}$ and there is a run ρ of $\mathcal{S}$ from c_1 to c_2 witnessing $(c_1, c_2) \in R^{\Rightarrow}$.

We informally explain how $\mathcal{A}_{R^{\Rightarrow}}$ processes the encoding $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ of two configurations c_1, c_2 in order to verify $(c_1, c_2) \in R^{\Rightarrow}$. First of all the automaton guarantees that $s_1 = \mathrm{pop}_2{}^k(s_2)$ for some $k \in \mathbb{N}$ (this is the case if and only if $\mathrm{Enc}(s_1)$ is a subtree of $\mathrm{Enc}(s_2)$, the rightmost leaf l of $\mathrm{Enc}(s_1)$ does not have a 0-successor in $\mathrm{Enc}(s_2)$ and for each $d \le_{\mathrm{lex}} l$, $d \in \mathrm{Enc}(s_2) \Leftrightarrow d \in \mathrm{Enc}(s_1)$).

Assume that $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$ with $s_1 = \mathrm{pop}_2{}^k(s_2)$. If there is a run from c_1 to c_2 its form is described in Corollary 4.10: it is a sequence of loops followed by one operation each that starts in some generalised milestone of s_2 and leads to the next generalised

milestone (with respect to $\ll$). Recall the following: each node in $\mathrm{Enc}(s_2)$ which is not contained in $\mathrm{Enc}(s_1)$ corresponds to a milestone in $\mathrm{MS}(s_2) \setminus \mathrm{MS}(s_1)$ via $\mathrm{LStck}(d, \mathrm{Enc}(s_2))$. Moreover, each node in the rightmost branch of $\mathrm{Enc}(s_1)$ or in $\mathrm{Enc}(s_2) \setminus \mathrm{Enc}(s_1)$ corresponds to a generalised milestone in $\mathrm{GMS}(s_2) \setminus \mathrm{GMS}(s_1)$ via IgM. The essence of Remark 5.6 is that for each generalised milestones represented by a node d the node representing the $\ll$-successor of this generalised milestone can be found locally around d. Since we can compute the possible loops of the stack $\mathrm{LStck}(d, \mathrm{Enc}(c_2))$ and $\mathrm{IgM}(d, \mathrm{Enc}(c_2))$ along the path from the root to d, a tree-automaton may guess the initial and final states of each part of the decomposition of a run according to Corollary 4.10 and check the local compatibility of each of the guesses.

The detailed definition of $\mathcal{A}_{R\Rightarrow}$ as well as a proof of the following lemma can be found in appendix D.

Lemma 5.11. *There are two polynomials p_1, p_2 such that the following holds. Let $\mathcal{S}$ be a level 2 collapsible pushdown system with stack alphabet Σ and state space Q. There is an automaton with $p_1(|\Sigma|) \cdot \exp(p_2(|Q|))$ many states that accepts the convolution of two trees if and only if this convolution is of the form $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ for c_1, c_2 configurations with $(c_1, c_2) \in R^{\Rightarrow}$.*

5.3. **Regularity of** Reach_L**.** In this part, we use the closure of collapsible pushdown systems under products with finite automata in order to provide a proof of the automaticity of all regular reachability predicates (Proposition 3.9): we reduce regular reachability to reachability in a product of the collapsible pushdown system with the automaton for the regular language.

Recall that for $L \subseteq \Gamma^*$ some (string-) language, Reach_L is the binary relation that contains configurations $(c, \hat{c})$ if and only if there is a run ρ from c to $\hat{c}$ such that the labels of the transitions used in ρ form a word $w \in L$.

Let L be some regular language and $\mathcal{A}_L$ an automaton recognising L. We construct the product $\mathcal{S} \times \mathcal{A}_L$. Reach_L on $\mathrm{CPG}(\mathcal{S})$ is expressible via the relation Reach on $\mathrm{CPG}(\mathcal{S} \times \mathcal{A}_L)$. As a corollary of this result, we obtain the tree-automaticity of Reach_L.

Definition 5.12. Let $\mathcal{S} = (Q, \Sigma, \Gamma, q_i, \Delta)$ be a 2-CPS and let $\mathcal{A}_L = (Q_L, \Gamma, i_0, F, \Delta_L)$ be a finite word-automaton. We define the product of $\mathcal{S}$ and $\mathcal{A}_L$ to be the collapsible pushdown system

$$\mathcal{S} \times \mathcal{A}_L := (Q \times Q_L, \Sigma, \Gamma, (q_i, i_0), \bar{\Delta}) \text{ where}$$
$$\bar{\Delta} := \{((q, q_l), \sigma, \gamma, (q', q_l'), \mathrm{op}) : (q, \sigma, \gamma, q', \mathrm{op}) \in \Delta \text{ and } (q_l, \gamma, q_l') \in \Delta_L\}.$$

A straightforward induction shows that there is a run of $\mathcal{S} \times \mathcal{A}_L$ from $((q, i_0), s)$ to $((q', q_f), s')$ for $q, q' \in Q$, and $q_f \in F$ if and only if there is a run of $\mathcal{S}$ from (q, s) to (q', s') such that the labels of the run form a word in L. Since Reach is a tree-automatic relation, we obtain the proof of Proposition 3.9.

Proof of Proposition 3.9. Recall Remark 4.3. It says that there is a positive existential first-order formula defining Reach in terms of $R^{\Leftarrow}, R^{\Downarrow}, R^{\Uparrow}$ and $R^{\Rightarrow}$. Due to Lemmas 3.7, 5.7, 5.9, 5.10 and 5.11, there are polynomials p and p' such that there is a (nondeterministic) tree-automaton $\mathcal{A}$ corresponding to Reach on $\mathcal{S} \times \mathcal{A}_L$ with $p(|\Sigma|) \cdot \exp(p'(|Q| \cdot |P|))$ many states. We obtain that for states q, q' and stacks s, s' there is a final state $q_f \in F$ of $\mathcal{A}_L$

such that $\mathcal{A}$ accepts $(\mathrm{Enc}((q, i_0), s), \mathrm{Enc}((q', q_f), s'))$ if and only if $((q, s), (q', s')) \in \mathrm{Reach}_L$ holds in $\mathcal{S}$.

This almost completes the proof. We only have to modify $\mathcal{A}$ in such a way that it guesses q_f and treats the configuration (q, s) as if it was $((q, i_0), s)$. This can easily be done without increasing the number of states of the automaton because the states of the configurations are encoded in the roots of the trees. We explain in Appendix E the detailed modification. $\square$

6. Conclusion

We have shown that level 2 collapsible pushdown graphs are uniformly tree-automatic. Thus, their first-order theories are decidable with nonelementary complexity. Moreover, even first-order extended by regular reachability is decidable because of the automaticity of the regular reachability relations. Our result is sharp in several directions. First, we have also shown a nonelementary lower bound for the complexity of the first-order model-checking problem on collapsible pushdown graphs. Furthermore, Broadbent [5] showed that the first-order theories of collapsible pushdown graphs are undecidable from level 3 on (which implies that they are not tree-automatic).

Acknowledgement

References

[1] R. Alur, S. Chaudhuri, and P. Madhusudan. Languages of nested trees. In *Proc. 18th International Conference on Computer-Aided Verification*, volume 4144 of *LNCS*, pages 329–342. Springer, 2006.
[2] A. Blumensath. Automatic structures. Diploma thesis, RWTH Aachen, 1999.
[3] A. Blumensath. On the structure of graphs in the Caucal hierarchy. *Theoretical Computer Science*, 400:19–45, 2008.
[4] C. H. Broadbent, A. Carayol, C.-H. Luke Ong, and O. Serre. Recursion schemes and logical reflection. In *LICS*, Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, pages 120–129, 2010.
[5] Christopher H. Broadbent. The limits of decidability for first-order logic on cpda graphs. In Christoph Dürr and Thomas Wilke, editors, *STACS*, volume 14 of *LIPIcs*, pages 589–600. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
[6] 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.
[7] D. Caucal. On infinite terms having a decidable monadic theory. In *MFCS 02*, pages 165–176, 2002.

[8] 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.

[9] Kevin J. Compton and C. Ward Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Ann. Pure Appl. Logic*, 48(1):1–79, 1990.

[10] 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.

[11] A. Kartzow. *First-Order Model Checking On Generalisations of Pushdown Graphs*. PhD thesis, Technische Universität Darmstadt, Fachbereich Mathematik, 2011.

[12] Alexander Kartzow. Collapsible pushdown graphs of level 2 are tree-automatic. In Jean-Yves Marion and Thomas Schwentick, editors, *STACS*, volume 5 of *LIPIcs*, pages 501–512. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

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

[14] A. N. Maslov. The hierarchy of indexed languages of an arbitrary level. *Sov. Math., Dokl.*, 15:1170–1174, 1974.

[15] A. N. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 12:38–43, 1976.

[16] S. Rubin. Automata presenting structures: A survey of the finite string case. *Bulletin of Symbolic Logic*, 14(2):169–209, 2008.

[17] Hugo Volger. Turing machines with linear alternation, theories of bounded concatenation and the decision problem of first order theories. *Theor. Comput. Sci.*, 23:333–337, 1983.

Appendix A. Proof of Bijectivity of Enc

We start by explicitly constructing the inverse of Enc. This inverse is called Dec. Since Enc removes the collapse links of the elements in a stack, we have to restore these now. In order to restore the collapse links we use the following auxiliary functions for each $g \in \mathbb{N}$

$$f_g : \{\varepsilon\} \cup (\Sigma \times \{1,2\}) \to \{\varepsilon\} \cup \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$$

which map labels of trees to 1-words of length up to 1. We set

$$f_g(\tau) := \begin{cases} \sigma & \text{if } \tau = (\sigma, 1), \\ (\sigma, 2, g) & \text{if } \tau = (\sigma, 2), \\ \varepsilon & \text{if } \tau = \varepsilon. \end{cases}$$

In the next definition g is the width of the stack decoded so far.

Definition A.1. Let $\Gamma := (\Sigma \times \{1,2\}) \cup \{\varepsilon\}$. Recall that encodings of stacks are trees in $\mathbb{T}_\Gamma$. We define the function $\text{Dec} : \mathbb{T}_\Gamma \times \mathbb{N} \to (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{*2}$ as follows. Let

$$\text{Dec}(T, g) = \begin{cases} f_g(T(\varepsilon)) & \text{if } \text{dom}(T) = \{\varepsilon\}, \\ f_g(T(\varepsilon)) \setminus \text{Dec}(T_0, g) & \text{if } 1 \notin \text{dom}(T), 0 \in \text{dom}(T), \\ f_g(T(\varepsilon)) \setminus (\varepsilon : \text{Dec}(T_1, g+1)) & \text{if } 0 \notin \text{dom}(T), 1 \in \text{dom}(T), \\ f_g(T(\varepsilon)) \setminus (\text{Dec}(T_0, g) : \text{Dec}(T_1, g + G(T_0))) & \text{otherwise,} \end{cases}$$

where $G(T_0) := |\text{Dec}(T_0, 0)|$ is the width of the stack encoded in T_0. For a tree $T \in \mathbb{T}^{\text{Enc}}$, the decoding of T is

$$\text{Dec}(T) := (T(\varepsilon), \text{Dec}(T_0, 0)) \in Q \times (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}.$$

Remark A.2. Obviously, for each $T \in \mathbb{T}^{\text{Enc}}$, $\text{Dec}(T) \in Q \times (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$. In fact, the image of Dec is contained in Cnf, i.e., $\text{Dec}(T) = (q, s)$ such that s is a level 2 stack. The verification of this claim relies on two important observations.

Firstly, $T(0) = (\bot, 1)$ due to condition 2 of Definition 3.5. Thus, all words in s start with letter $\bot$. s is a stack if and only if the link structure of s can be created using the push, clone and pop_1 operations. The proof of this claim can be done by a tedious but straightforward induction. We only sketch the most important observations for this fact.

Every letter a of the form $(\sigma, 2, g)$ occurring in s is either a clone or can be created by the $\text{push}_{\sigma, 2}$ operation. We call a a clone if a occurs in s in some word waw' such that the word to the left of this word has wa as prefix. Note that cloned elements are those that can be created by use of the clone_2 and pop_1 operations from a certain substack of s.

If a is not a clone in this sense, then Dec creates the letter a because there is some $(\sigma, 2)$-labelled node in T corresponding to a. Now, the important observation is that Dec defines $a = f_g((\sigma, 2))$ where $g+1$ is the width of the stack decoded from the lexicographically smaller nodes. Hence, the letter a occurs in the $(g+1)$-st word of s and points to the g-th word. Such a letter a can clearly be created by a $\text{push}_{\sigma, 2}$ operation. Thus, all 2-words in the image of Dec can be generated by stack operations from the initial stack. A reformulation of this observation is that the image of Dec only contains configurations.

Now, we prove that Dec is injective on $\mathbb{T}^{\text{Enc}}$. Afterwards, we show that $\text{Dec} \circ \text{Enc}$ is the identity on the set of all configurations. This implies that Dec is a surjective map from $\mathbb{T}^{\text{Enc}}$ to Cnf. Putting both facts together, we obtain that Dec is the inverse of Enc whence, of course, Enc is bijective.

Lemma A.3. Dec *is injective on* $\mathbb{T}^{\mathrm{Enc}}$.

Proof. Assume that there are trees $T', U' \in \mathbb{T}^{\mathrm{Enc}}$ with $\mathrm{Dec}(T') = \mathrm{Dec}(U') = (q, s)$. Then by definition $T'(\varepsilon) = U'(\varepsilon) = q$. Thus, we only have to compare the subtrees rooted at 0, i.e., $T := T'_0$ and $U := U'_0$. From our assumption it follows that $\mathrm{Dec}(T, 0) = \mathrm{Dec}(U, 0)$.

Note that the roots of T and of U are both labelled by $(\bot, 1)$. The lemma follows from the following claim.

Claim. Let T and U be trees such that there are $T', U' \in \mathbb{T}^{\mathrm{Enc}}$ and $d \in \mathrm{dom}(T') \setminus \{\varepsilon\}$, $e \in \mathrm{dom}(U') \setminus \{\varepsilon\}$ such that $T = T'_d$ and $U = U'_e$. If $\mathrm{Dec}(T, m) = \mathrm{Dec}(U, m)$ and either $T(\varepsilon) = U(\varepsilon) = \varepsilon$ or $T(\varepsilon) \in \Sigma \times \{1, 2\}$ and $U(\varepsilon) \in \Sigma \times \{1, 2\}$, then $U = T$.[4]

The proof is by induction on the depth of the trees U and T. If $\mathrm{dpt}(U) = \mathrm{dpt}(T) = 0$, $\mathrm{Dec}(U, m)$ and $\mathrm{Dec}(T, m)$ are uniquely determined by the label of their roots. A straightforward consequence of the definition of Dec is that $U(\varepsilon) = T(\varepsilon)$ whence $U = T$.

Now, assume that the claim is true for all trees of depth at most k for some fixed $k \in \mathbb{N}$. Let U and T be trees of depth at most $k + 1$.

We proceed by a case distinction on whether the left or right subtree of T and U are defined. In fact, we will later prove that $\mathrm{Dec}(T, m) = \mathrm{Dec}(U, m)$ implies that

(1) $T_0 \neq \emptyset$ if and only if $U_0 \neq \emptyset$ and
(2) $T_1 \neq \emptyset$ if and only if $U_1 \neq \emptyset$.

We first prove that $\mathrm{Dec}(T, m) = \mathrm{Dec}(U, m)$ and conditions (1) and (2) imply that $U = T$. Afterwards we show that all possible combinations that do not satisfy these conditions imply $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(1) Assume that $U_0 = U_1 = T_0 = T_1 = \emptyset$. Then $\mathrm{dpt}(T) = \mathrm{dpt}(U) = 0$. For trees of depth 0 we have already shown that $\mathrm{Dec}(U, 0) = \mathrm{Dec}(T, 0)$ implies $U = T$.

(2) Assume that $U_0 = \emptyset$, $U_1 \neq \emptyset$, $T_0 = \emptyset$ and $T_1 \neq \emptyset$. In this case

$$\mathrm{Dec}(U, m) = f_m(U(\varepsilon)) \setminus (\varepsilon : \mathrm{Dec}(U_1, m + 1)) \text{ and}$$
$$\mathrm{Dec}(T, m) = f_m(T(\varepsilon)) \setminus (\varepsilon : \mathrm{Dec}(T_1, m + 1)).$$

Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$, we can directly conclude that $U(\varepsilon) = T(\varepsilon)$. But then $\mathrm{Dec}(T, m) = \mathrm{Dec}(U, m)$ implies that $\mathrm{Dec}(T_1, m + 1) = \mathrm{Dec}(U_1, m + 1)$. Since $\mathrm{dpt}(T_1) \leq k$ and $\mathrm{dpt}(U_1) \leq k$, the induction hypothesis implies that $T_1 = U_1$. We conclude that $T = U$.

(3) Assume that $U_0 \neq \emptyset$, $U_1 = \emptyset$, $T_0 \neq \emptyset$, and $T_1 = \emptyset$. In this case,

$$\mathrm{Dec}(U, m) = f_m(U(\varepsilon)) \setminus \mathrm{Dec}(U_0, m) \text{ and}$$
$$\mathrm{Dec}(T, m) = f_m(T(\varepsilon)) \setminus \mathrm{Dec}(T_0, m).$$

Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$, we conclude that $U(\varepsilon) = T(\varepsilon)$ and $\mathrm{Dec}(U_0, m) = \mathrm{Dec}(T_0, m)$. Since the depths of U_0 and of T_0 are at most k, the induction hypothesis implies $U_0 = T_0$ whence $U = T$.

(4) Assume that $U_0 \neq \emptyset$, $U_1 \neq \emptyset$, $T_0 \neq \emptyset$, and $T_1 \neq \emptyset$. Then we have

$$\mathrm{Dec}(U, m) = f_m(U(\varepsilon)) \setminus \big(\mathrm{Dec}(U_0, m) : \mathrm{Dec}(U_1, m + m')\big) \text{ and}$$
$$\mathrm{Dec}(T, m) = f_m(T(\varepsilon)) \setminus \big(\mathrm{Dec}(T_0, m) : \mathrm{Dec}(T_1, m + m'')\big)$$

for some natural numbers $m', m'' > 0$.

---

[4]Since a node d of a tree in $\mathbb{T}^{\mathrm{Enc}}$ is labelled by ε iff $d \in \{0, 1\}^* 1$, the pair of subtrees T_i and U_i inherit this condition for all $i \in \{0, 1\}$.

Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$ this implies that the roots of U and T coincide. Hence,

$$\mathrm{Dec}(U_0, m) : \mathrm{Dec}(U_1, m + m') = \mathrm{Dec}(T_0, m) : \mathrm{Dec}(T_1, m + m'')$$

If $\mathrm{Dec}(U_0, m) = \mathrm{Dec}(T_0, m)$, then the induction hypothesis yields $U_0 = T_0$. Furthermore, this implies $\mathrm{Dec}(U_1, m+m') = \mathrm{Dec}(T_1, m+m'')$ and $m' = m''$ whence by induction hypothesis $U_1 = T_1$. In this case we conclude immediately that $T = U$.

The other case is that $\mathrm{Dec}(U_0, m) \neq \mathrm{Dec}(T_0, m)$. We conclude immediately that the width of $\mathrm{Dec}(U_0, m)$ and the width of $\mathrm{Dec}(T_0, m)$ do not coincide. We prove that this case contradicts the assumption that $\mathrm{Dec}(U, m) = \mathrm{Dec}(T, m)$.

Let us assume that $\mathrm{Dec}(U_0, m) = \mathrm{pop}_2{}^z (\mathrm{Dec}(T_0, m))$ for some $z \in \mathbb{N} \setminus \{0\}$. Note that this implies that the first word of $\mathrm{Dec}(U_1, m + m')$ is a word in $\mathrm{Dec}(T_0, m)$.

Since $U(0)$ is a left successor in some tree belonging to $\mathbb{T}^{\mathrm{Enc}}$, it is labelled by some $(\sigma, l) \in \Sigma \times \{1, 2\}$. We make a case distinction on l.

(a) Assume that $U(0) = (\sigma, 2)$ for some $\sigma \in \Sigma$. Then all words in $\mathrm{Dec}(T_0, m)$ start with the letter $(\sigma, 2, m)$. Thus, the first word of $\mathrm{Dec}(U_1, m + m')$ must also start with $(\sigma, 2, m)$. But all collapse links of level 2 in $\mathrm{Dec}(U_1, m+m')$ are at least $m+m' > m$. This is a contradiction.

(b) Otherwise, $U(1) = (\sigma, 1)$ for some $\sigma \in \Sigma$. Thus, all words in $\mathrm{Dec}(T_0, m)$ start with the letter σ. Thus, the first word of $\mathrm{Dec}(U_0, m)$ and the first word of $\mathrm{Dec}(U_1, m+m')$ have to start with σ. But this implies $U(0) = U(10) = (\sigma, 1)$. This contradicts the assumption that U is a proper subtree of a tree from $\mathbb{T}^{\mathrm{Enc}}$ (cf. condition 6 of Definition 3.5).

Both cases result in contradictions. Thus, it is not the fact that there is some $z \in \mathbb{N} \setminus \{0\}$ such that

$$\mathrm{Dec}(U_0, m) = \mathrm{pop}_2{}^z (\mathrm{Dec}(T_0, m))$$

By symmetry, we obtain that there is no $z \in \mathbb{N} \setminus \{0\}$ such that

$$\mathrm{Dec}(T_0, m) = \mathrm{pop}_2{}^z (\mathrm{Dec}(U_0, m)) .$$

Thus, we conclude that $\mathrm{Dec}(T_0, m) = \mathrm{Dec}(U_0, m)$ whence $U = T$ as shown above.

If $\mathrm{Dec}(T, m) = \mathrm{Dec}(U, m)$, one of the previous cases applies because the following case distinction shows that all other cases for the defined or undefined subtrees of T and U imply $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(1) Assume that $U_0 = U_1 = T_0 = \emptyset$ and $T_1 \neq \emptyset$. In this case, $\mathrm{Dec}(U, m)$ is $[\varepsilon]$ or $[\tau]$ for some $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Furthermore,

$$\mathrm{Dec}(T, m) = f_m(T(\varepsilon)) \setminus (\varepsilon : \mathrm{Dec}(T_1, m + 1)).$$

It follows that $|\mathrm{Dec}(T, m)| \geq 2 > |\mathrm{Dec}(U, m)| = 1$ whence $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(2) Assume that $U_0 = U_1 = \emptyset$, $T_0 \neq \emptyset$, and $T_1 = \emptyset$. In this case, $\mathrm{Dec}(U, m)$ is again $[\varepsilon]$ or $[\tau]$ for some $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$, we conclude that $|f_m(T(\varepsilon)| = |\mathrm{Dec}(U, m)|$. Moreover,

$$\mathrm{Dec}(T, m) = f_m(T(\varepsilon)) \setminus f_m(T(0)) \setminus s$$

for some 2-word s. Since T is a subtree of a tree in $\mathbb{T}^{\mathrm{Enc}}$, $T(0) \in \Sigma \times \{1, 2\}$. Thus, $f_m(T(0)) \in \Sigma \cup (\Sigma \times \{1, 2\} \times \mathbb{N})$. We conclude that the length of the first word of $\mathrm{Dec}(T, m)$ is greater than the length of the first word of $\mathrm{Dec}(U, m)$. Thus, $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(3) Assume that $U_0 = U_1 = \emptyset$, $T_0 \neq \emptyset$, and $T_1 \neq \emptyset$. Completely analogous to case 1, we conclude that $|\mathrm{Dec}(T, m)| \geq 2 > |\mathrm{Dec}(U, m)| = 1$ whence $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(4) Assume that $U_0 = \emptyset$, $U_1 \neq \emptyset$, and $T_0 = T_1 = \emptyset$. Exchanging the roles of U and T, this is exactly the same as case 1.

(5) Assume that $U_0 = \emptyset$, $U_1 \neq \emptyset$, $T_0 \neq \emptyset$, and $T_1 = \emptyset$. Analogously to case 2, we derive that the length of the first word of $\mathrm{Dec}(T, m)$ is greater than the length of the first word of $\mathrm{Dec}(U, m)$. Thus, $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(6) Assume that $U_0 = \emptyset$, $U_1 \neq \emptyset$, $T_0 \neq \emptyset$, and $T_1 \neq \emptyset$. Analogously to case 2, we derive that the length of the first word of $\mathrm{Dec}(T, m)$ is greater than the length of the first word of $\mathrm{Dec}(U, m)$. Thus, $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(7) Assume that $U_0 \neq \emptyset$, and $U_1 = T_0 = T_1 = \emptyset$. Exchanging the roles of U and T, this is exactly the case 2.

(8) Assume that $U_0 \neq \emptyset$, $U_1 = T_0 = \emptyset$, and $T_1 \neq \emptyset$. Exchanging the roles of U and T, this is exactly the case 5.

(9) Assume that $U_0 \neq \emptyset$, $U_1 = \emptyset$, $T_0 \neq \emptyset$, and $T_1 \neq \emptyset$. In this case,

$$\mathrm{Dec}(U, m) = f_m(U(\varepsilon)) \setminus \mathrm{Dec}(U_0, m)$$

$$\text{and } \mathrm{Dec}(T, m) = f_m(T(\varepsilon)) \setminus \big(\mathrm{Dec}(T_0, m) : \mathrm{Dec}(T_1, m + m')\big)$$

for some $m' \in \mathbb{N} \setminus \{0\}$. Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$, we conclude that $U(\varepsilon) = T(\varepsilon)$. Now,

$$\mathrm{Dec}(U_0, m) = \tau \setminus u'$$

for $\tau = f_m(U(0)) \in \Sigma \cup (\Sigma \times \{2\} \times \{m\})$ and u' some level 2-word. We distinguish the following cases.

First assume that $\tau = (\sigma, 2, m)$. For all letters in $T' := \mathrm{Dec}(T_1, m + m')$ of collapse level 2, the collapse link is greater or equal to $m + m'$. Hence, T' does not contain a symbol $(\sigma, 2, m)$ whence $\mathrm{Dec}(U, m) \neq \mathrm{Dec}(T, m)$.

Otherwise, $\tau \in \Sigma$. But then $\mathrm{Dec}(U, m) = \mathrm{Dec}(T, m)$ would imply that

$$\mathrm{Dec}(T_0, m) = \tau \setminus T'$$

$$\text{and } \mathrm{Dec}(T_1, m + m') = \tau \setminus T''$$

for certain nonempty level 2-words T' and T''. Since $T(1) = \varepsilon$, it follows that $T(0) = T(10) = (\tau, 1)$ which contradicts the fact that T is a subtree of some tree from $\mathbb{T}^{\mathrm{Enc}}$.

Thus, we conclude that $\mathrm{Dec}(T, m) \neq \mathrm{Dec}(U, m)$.

(10) Assume that $U_0 \neq \emptyset$, $U_1 \neq \emptyset$, and $T_0 = T_1 = \emptyset$. Exchanging the roles of U and T, this is the same as case 3.

(11) Assume that $U_0 \neq \emptyset$, $U_1 \neq \emptyset$, $T_0 = \emptyset$, and $T_1 \neq \emptyset$. Exchanging the roles of U and T, this is the same as case 6.

(12) Assume that $U_0 \neq \emptyset$, $U_1 \neq \emptyset$, $T_0 \neq \emptyset$, and $T_1 = \emptyset$. Exchanging the roles of U and T, this is the same as case 9.

Hence, we have seen that $\mathrm{Dec}(T, m) = \mathrm{Dec}(U, m)$ implies that each of the subtrees of T is defined if and only if the corresponding subtree of U is defined. Under this condition, we concluded that $U = T$. Thus, the claim holds and the lemma follows as indicated above. $\square$

Next, we prove that Dec is a surjective map from $\mathbb{T}^{\mathrm{Enc}}$ to Cnf. This is done by induction on the size of blocklines used to encode a stack. In this proof we use the notion of *left-maximal* blocks and *good* blocklines. Let

$$s : \big(w \setminus (w' : b)\big) : s'$$

be a stack where s and s' are 2-words, w, and w' are words, and b is a τ-block.[5] We call b *left maximal* in this stack if either $b = [\tau]$ or $b = \tau\tau' \setminus b'$ such that w' does not start with $\tau\tau'$ for some $\tau' \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. We call a blockline in some stack *good*, if its first block is left maximal. Furthermore, we call the blockline starting with the block b *left maximal* if w' does not start with τ. Recall that the encoding of stacks works on left maximal blocks and good blocklines.

Lemma A.4. Dec $\circ$ Enc *is the identity, i.e.,* $\mathrm{Dec}(\mathrm{Enc}(c)) = c$, *for all* $c \in \mathrm{Cnf}$.

Corollary A.5. Dec $: \mathbb{T}^{\mathrm{Enc}} \to \mathrm{Cnf}$ *is surjective.*

Proof of Lemma A.4. Let $c = (q, s)$ be a configuration. Since Dec and Enc encode and decode the state of c in the root of $\mathrm{Enc}(c)$, it suffices to show that

$$\mathrm{Dec}(\mathrm{Enc}(s, (\bot, 1)), 0) = s$$

for all stacks $s \in \mathrm{Stck}(\Sigma)$. We proceed by induction on blocklines of the stack s. For this purpose we reformulate the lemma in the following claim.

Claim. Let s' be some stack which decomposes as $s' = s'' : (w \setminus b) : s'''$ such that $b \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$ is a good τ-blockline for some $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Then
(1) $\mathrm{Dec}(\mathrm{Enc}(b, \varepsilon), |s''|) = b'$ for the unique 2-word b' such that $b = \tau \setminus b'$ and
(2) if b is left maximal, then $\mathrm{Dec}(\mathrm{Enc}(b, (\sigma, l)), |s''|) = b$ where $\sigma = \mathrm{Sym}(\tau)$ and $l = \mathrm{CLvl}(\tau)$.
Note that the conditions in the second part require that either $\tau \in \Sigma$ or $\tau = (\sigma, 2, |s''|)$ for some $\sigma \in \Sigma$.

The lemma follows from the second part of the claim because every stack is a left maximal $\bot$-blockline.

We prove both claims by parallel induction on the size of b. As an abbreviation we set $g := |s''|$. We write $\overset{(1)}{=} (\overset{(2)}{=}$, respectively) when some equality is due to the induction hypothesis of the first claim (the second claim , respectively). The arguments for the first claim are as follows.

- If $b = [\tau]$ for $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$, the claim is true because

$$\mathrm{Dec}(\mathrm{Enc}(b, \varepsilon), g) = \mathrm{Dec}(\varepsilon, g) = \varepsilon.$$

- If there are $b_1, b_1' \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{*2}$ such that

$$b = [\tau] : b_1 = [\tau] : \big(\tau \setminus b_1'\big) \text{ then}$$

$$\mathrm{Dec}(\mathrm{Enc}(b, \varepsilon), g) = \mathrm{Dec}(\varepsilon \langle \emptyset; \mathrm{Enc}(b_1, \varepsilon)\rangle, g)$$
$$= f_g(\varepsilon) \setminus (\varepsilon : \mathrm{Dec}(\mathrm{Enc}(b_1, \varepsilon), g + 1))$$
$$\overset{(1)}{=} \varepsilon \setminus (\varepsilon : b_1') = \varepsilon : b_1' = b'.$$

---

[5]In this definition, we explicitly allow the case $s = w' = \emptyset$, i.e., a stack of the form $(w \setminus b) : s'$.

- Assume that there is some $\tau' \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ and some $b_1 \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{*2}$ such that

$$b = \tau\tau' \setminus b_1.$$

  The assumption that b is good implies that the blockline $\tau' \setminus b_1$ is left maximal whence

$$\mathrm{Dec}(\mathrm{Enc}(b, \varepsilon), g) = \mathrm{Dec}(\varepsilon \langle \mathrm{Enc}(\tau' \setminus b_1, (\mathrm{Sym}(\tau'), \mathrm{CLvl}(\tau'))); \emptyset \rangle, g)$$
$$= f_g(\varepsilon) \setminus \mathrm{Dec}(\mathrm{Enc}(\tau' \setminus b_1, (\mathrm{Sym}(\tau'), \mathrm{CLvl}(\tau')), g))$$
$$\overset{(2)}{=} \tau' \setminus b_1 = b'.$$

- The last case is that

$$b = \tau \setminus ((\tau' \setminus b_1) : b_2)$$

  for b_2 a blockline of s not starting with τ'. By this we mean that $b_2 \neq \tau' w' : b_2'$ for any word w' and any 2-word b_2'. Since b is good, $\tau' \setminus b_1$ is a left maximal blockline. Furthermore, $\tau \setminus b_2$ is a good blockline. Thus,

$$\mathrm{Dec}(\mathrm{Enc}(b, \varepsilon), g)$$
$$= \mathrm{Dec}\left(\varepsilon \langle \mathrm{Enc}\left(\tau' \setminus b_1, (\mathrm{Sym}(\tau'), \mathrm{CLvl}(\tau'))\right); \mathrm{Enc}(\tau \setminus b_2, \varepsilon) \rangle, g\right)$$
$$= f_g(\varepsilon) \setminus \left(\mathrm{Dec}(\mathrm{Enc}(\tau' \setminus b_1, (\mathrm{Sym}(\tau'), \mathrm{CLvl}(\tau'))), g) : \mathrm{Dec}(\mathrm{Enc}(\tau \setminus b_2, \varepsilon), g + f)\right),$$

  where

$$f = |\mathrm{Dec}(\mathrm{Enc}(\tau' \setminus b_1, (\mathrm{Sym}(\tau'), \mathrm{CLvl}(\tau'))), g)| \overset{(2)}{=} |b_1|.$$

  From this, we obtain that

$$\mathrm{Dec}(\mathrm{Enc}(b, \varepsilon), g)$$
$$\overset{(2)}{=} \varepsilon \setminus \left((\tau' \setminus b_1) : \mathrm{Dec}(\mathrm{Enc}(\tau \setminus b_2, \varepsilon), g + f)\right)$$
$$\overset{(1)}{=} (\tau' \setminus b_1) : b_2 = b'.$$

For the proof of the second claim, note that the calculations are basically the same, but $f_g(\varepsilon)$ is replaced by $f_g(\sigma, l)$. Thus, if $l = 1$ then $f_g(\sigma, l) = \sigma = \tau$. For the case $l = 2$, recall that $g = |s''|$ whence $f_g(\sigma, l) = (\sigma, 2, |s''|)$. Note that $\mathrm{CLnk}(\tau) = |s''|$ due to the left maximality of b.

Thus, one proves the second case using the same calculations, but replacing ε by τ. $\square$

From the previous lemmas, we directly obtain Lemma 3.8, i.e., we obtain that Enc is bijective.

APPENDIX B. AUTOMATON FOR RELATION $R^{\Leftarrow}$

Given a CPS $\mathcal{S}$, there is an automaton $\mathcal{A}_{R^{\Leftarrow}}$ that accepts the tree $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ for arbitrary configurations c_1 and c_2 if and only if $c_2 = \mathrm{pop}_2^k(c_1)$ and there is a run ρ from c_1 to c_2 such that $\rho(j) \not\leq c_2$ for all $j < \mathrm{length}(\rho)$, i.e., if and only if $(c_1, c_2) \in R^{\Leftarrow}$. The states of $\mathcal{A}_{R^{\Leftarrow}}$ come from the set

$$\{\bot, q_I, q_\emptyset, q_=\} \cup M \text{ with } M := \{S, R, P, C\} \times Q \times Q \times \Sigma \times \{1, 2\} \times 2^{Q \times Q}.$$

Before giving a definition of $\mathcal{A}_{R^{\Leftarrow}}$, we informally describe how $\mathcal{A}_{R^{\Leftarrow}}$ processes some tree $T := \mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$. An accepting run on T labels $d \in \{0, 1\}^*$

C1. by $\perp$ if $d \in T_+$;

C2. by q_I if $d = \varepsilon$; it is the initial state in which the automaton reads the states of c_1 and c_2, before it processes the encodings of the stacks,

C3. by $q_=$ if $d \in \text{Enc}(c_2)$ but not in the rightmost branch of $\text{Enc}(c_2)$; this state is used to check equality of the parts of $\text{Enc}(c_1)$ and $\text{Enc}(c_2)$ that are left of the rightmost branch of $\text{Enc}(c_2)$,

C4. by some element from $\{S\} \times Q \times Q \times \Sigma \times \{1,2\} \times 2^{Q \times Q}$ if d is in the rightmost branch of $\text{Enc}(c_2)$; S stands for searching the node encoding the final configuration of the run.

C5. by some element from the set $\{q_\emptyset\} \cup (\{R, P, C\} \times Q \times Q \times \Sigma \times \{1,2\} \times 2^{Q \times Q})$ if $d \in \text{Enc}(c_1) \setminus \text{Enc}(c_2)$.

Those labels that come from M are used to check the existence of some run from c_1 to c_2 as follows. For all $d \in \text{Enc}(c_1)$, let $d^\nearrow$ be the rightmost leaf of the subtree induced by d in $\text{Enc}(c_1)$. Set $s_d^\nearrow := \text{LStck}(d^\nearrow, \text{Enc}(c_1))$. Let $\bar{q} \in M$ be the label of some node d. By $\pi_i(\bar{q})$ we denote the projection of $\bar{q}$ to the i-th component. Depending on $\pi_1(\bar{q})$ we define a stack s_d as follows.

(1) If $\pi_1(\bar{q}) = S$, set s_d to be the stack of c_2.
(2) If $\pi_1(\bar{q}) = R$, set $s_d := \text{pop}_2(\text{LStck}(d, \text{Enc}(c_1)))$.
(3) If $\pi_1(\bar{q}) = C$, set $s_d := \text{collapse}(\text{LStck}(d, \text{Enc}(c_1)))$.
(4) If $\pi_1(\bar{q}) = P$, set $s_d := \text{pop}_1(\text{LStck}(d, \text{Enc}(c_1)))$.

An accepting run ρ of $\mathcal{A}_{R \Leftarrow}$ will label some node d by $\bar{q} \in M$ such that

C6. $\pi_4(\bar{q}) = \text{Sym}(\text{LStck}(d, \text{Enc}(c_1)))$,

C7. $\pi_5(\bar{q}) = \text{CLvl}(\text{LStck}(d, \text{Enc}(c_1)))$, and

C8. $\pi_6(\bar{q}) = \text{Rt}(\text{LStck}(d, \text{Enc}(c_1)))$.

C9. Moreover, if $\pi_1(\bar{q}) \neq P$ then there is a run ρ from $(\pi_2(\bar{q}), s_d^\nearrow)$ to $(\pi_3(\bar{q}), s_d)$ (which is an infix of some run witnessing $(c_1, c_2) \in R^\Leftarrow$). The meaning of the labels R and C is as follows. If $\pi_1(\bar{q}) = R$ then ρ ends in $\text{pop}_2(\text{LStck}(d, \text{Enc}(c_1)))$. If $\pi_1(\bar{q}) = C$ then ρ ends in $\text{collapse}(\text{LStck}(d, \text{Enc}(c_1)))$ and the collapse level is 2 (moreover, ρ actually performs as the last operation a collapse on a copy of the topmost element of $\text{LStck}(d, c_1)$). Thus, in both cases the run will end in the stack s_d. To be more precise, the run ends in s_d and does not visit any substack of s_d before its final configuration.

C10. If $\pi_1(\bar{q}) = P$ then there is some stack s' with $s_d \leq s'$ and $\text{top}_2(s_d) = \text{top}_2(s')$ such that there is a run from $(\pi_2(\bar{q}), s_d^\nearrow)$ to $(\pi_3(\bar{q}), s')$ (which is again an infix of some run witnessing $(c_1, c_2) \in R^\Leftarrow$).

C11. ρ will label d by $q_\emptyset$ if there is a run from c_1 to c_2 not passing $s_e^\nearrow$ for all $d \leq e$.

Let us fix some notation. In this section, γ ranges over Γ, y ranges over $(\Sigma \times \{1,2\}) \cup \{\varepsilon\}$ and w ranges over all words of the form $w = \text{top}_2(s)\downarrow_0$. Whenever w is fixed, we write $\sigma := \text{Sym}(w)$ and $l = \text{CLvl}(w)$. Furthermore, x ranges over $\{(\sigma, l), \varepsilon\}$. The variables $q_1, q_2, q_1', q_2', q, q'$ range over Q. τ ranges over $\Sigma \setminus \{\perp\}$ and k over $\{1,2\}$. We use the abbreviation "$(q, \sigma, q', \text{ColPop}_k) \in \Delta$" for "$\exists \gamma$ such that

(1) $(q, \sigma, \gamma, q', \text{pop}_1) \in \Delta$ or
(2) $(q, \sigma, \gamma, q', \text{collapse}) \in \Delta$ and $k = 1$".

If w, τ and k are fixed, we write $w\tau_k$ for the word $w\theta$ where $\theta = \begin{cases} (\tau, 2, 0) & \text{if } k = 2, \\ \tau & \text{if } k = 1. \end{cases}$

Definition B.1. Fix some CPS $\mathcal{S} = (Q, \Sigma, \Gamma, \Delta, q_0)$. Define $\mathcal{A}_{R\Leftarrow} := (Q_A, \Sigma_A, \perp, F, \Delta_A)$ with $Q_A := \{\perp, q_I, q_\emptyset, q_=\} \cup M$, $\Sigma_A = (\{\varepsilon, \square\} \cup \{\Sigma \times \{1, 2\}\} \cup Q)^2$, $F = \{q_I\}$. Δ_A contains the following transitions.

T1. $(q_I, (q_1, q_2), (S, q_1, q_2, \perp, 1, \mathrm{Rt}(\perp_2)), \perp)$,

T2. $(q_\emptyset, (y, \square), Y, Z)$ for $Y, Z \in \{\perp, q_\emptyset\}$, and

T3. $(q_=, (y, y), Y, Z)$ for $Y, Z \in \{\perp, q_=\}$,

Fix some $\bar{q} := (S, q_1, q_2, \sigma, l, \mathrm{Rt}(w))$. We add the following transitions to Δ_A:

T4. $(\bar{q}, (x, x), \perp, \perp)$ if $q_1 = q_2$;

T5. $(\bar{q}, (x, x), \perp, \bar{q}_1)$ for $\bar{q}_1 = (R, q_1, q_2, \sigma, l, \mathrm{Rt}(w))$;

T6. $(\bar{q}, (x, x), X, \bar{q})$ for $X \in \{\perp, q_=\}$;

T7. $(\bar{q}, (x, x), \bar{q}_0, \perp)$ for $\bar{q}_0 = (S, q_1, q_2, \tau, k, \mathrm{Rt}(w\tau_k))$;

T8. $(\bar{q}, (x, x), \bar{q}_0, \bar{q}_1)$ for $\bar{q}_1 = (R, q_1, q_2', \sigma, l, \mathrm{Rt}(w))$ and $\bar{q}_0 = (S, q_2', q_2, \tau, k, \mathrm{Rt}(w\tau_k))$.

Fix some $\bar{q} := (R, q_1, q_2, \sigma, l, \mathrm{Rt}(w))$. We add the following transitions to Δ_A:

T9. $(\bar{q}, (x, \square), \perp, \perp)$ if $(q_1, q_2) \in \mathrm{Rt}(w)$;

T10. $(\bar{q}, (x, \square), \perp, \bar{q}_1)$ for $\bar{q}_1 = (R, q_1, q_2', \sigma, l, \mathrm{Rt}(w))$ such that $(q_2', q_2) \in \mathrm{Rt}(w)$;

T11. $(\bar{q}, (x, \square), \bar{q}_0, \perp)$ for $\bar{q}_0 = (R, q_1, q_2, \tau, i, \mathrm{Rt}(w\tau_i))$ for $i \in \{1, 2\}$;

T12. $(\bar{q}, (x, \square), \bar{q}_0, \bar{q}_1)$ for

$$\bar{q}_1 = (R, q_1, q_2', \sigma, l, \mathrm{Rt}(w)),$$
$$\bar{q}_0 = (R, q_2', q_2, \tau, i, \mathrm{Rt}(w\tau_i)) \text{ and}$$
$$i \in \{1, 2\};$$

T13. $(\bar{q}, (x, \square), \bar{q}_0, \perp)$ for $\bar{q}_0 = (C, q_1, q_2, \tau, 2, \mathrm{Rt}(w\tau_2))$;

T14. $(\bar{q}, (x, \square), \bar{q}_0, \bar{q}_1)$ for $\bar{q}_1 = (R, q_1, q_2', \sigma, l, \mathrm{Rt}(w))$ and $\bar{q}_0 = (C, q_2', q_2, \tau, 2, \mathrm{Rt}(w\tau_2))$.

Fix some $\bar{q} := (P, q_1, q_2, \sigma, l, \mathrm{Rt}(w))$. We add the following transitions to Δ_A:

T15. $(\bar{q}, (x, \square), \perp, \perp)$ if there is a q with $(q_1, q) \in \mathrm{1Lp}(w)$ and $(q, \sigma, q_2, \mathrm{ColPop}_l) \in \Delta$;

T16. $(\bar{q}, (x, \square), X, \bar{q})$ for $X \in \{\perp, q_\emptyset\}$;

T17. $(\bar{q}, (x, \square), \bar{q}_0, \perp)$ for

$$\bar{q}_0 = (P, q_1, q_2', \tau, k, \mathrm{Rt}(w\tau_k)),$$
$$(q_2', q) \in \mathrm{1Lp}(w) \text{ and}$$
$$(q, \sigma, q_2, \mathrm{ColPop}_l) \in \Delta;$$

T18. $(\bar{q}, (x, \square), \bar{q}_0, \bar{q}_1)$ for

$$\bar{q}_1 = (R, q_1, q_1', \sigma, l, \mathrm{Rt}(w)),$$
$$\bar{q}_0 = (P, q_1', q_2', \tau, k, \mathrm{Rt}(w\tau_k)),$$
$$(q_2', q) \in \mathrm{1Lp}(w) \text{ and}$$
$$(q, \sigma, q_2, \mathrm{ColPop}_l) \in \Delta.$$

Fix some $\bar{q} := (C, q_1, q_2, \sigma, l, \mathrm{Rt}(w))$ with $l = 2$ (whence x ranges here over $\{(\sigma, 2), \varepsilon\}$). We add the following transitions to Δ_A:

T19. $(\bar{q}, (x, \square), \perp, \perp)$ if there is a q with $(q_1, q) \in \mathrm{1Lp}(w)$ and $(q, \sigma, \gamma, q_2, \mathrm{collapse}) \in \Delta$;

T20. $(\bar{q}, (x, \square), X, \bar{q})$ for $X \in \{\perp, q_\emptyset\}$;

T21. $(\bar{q}, (x, \square), \bot, \bar{q}_1)$ for

$$\bar{q}_1 = (R, q_1, q_2', \sigma, 2, \mathrm{Rt}(w)),$$
$$(q_2', q) \in 1\mathrm{Lp}(w) \text{ and}$$
$$(q, \sigma, \gamma, q_2, \mathrm{collapse}) \in \Delta;$$

T22. $(\bar{q}, (x, \square), \bar{q}_0, \bot)$ for

$$\bar{q}_0 = (P, q_1, q_2', \tau, k, \mathrm{Rt}(w\tau_k)),$$
$$(q_2', q) \in 1\mathrm{Lp}(w) \text{ and}$$
$$(q, \sigma, \gamma, q_2, \mathrm{collapse}) \in \Delta;$$

T23. $(\bar{q}, (x, \square), \bar{q}_0, \bar{q}_1)$ for

$$\bar{q}_1 = (R, q_1, q_1', \sigma, 2, \mathrm{Rt}(w)),$$
$$\bar{q}_0 = (P, q_1', q_2', \tau, k, \mathrm{Rt}(w\tau_k)),$$
$$(q_2', q) \in 1\mathrm{Lp}(w) \text{ and}$$
$$(q, \sigma, \gamma, q_2, \mathrm{collapse}) \in \Delta.$$

Lemma B.2. *If $\mathcal{A}_{R\Leftarrow}$ accepts a tree $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ for configurations c_1, c_2, then $c_2 = \mathrm{pop}_2{}^k(c_1)$ and there is some run from c_1 to c_2 that does not reach a substack of c_2 before the final configuration, i.e., $(c_1, c_2) \in R^{\Leftarrow}$.*

Proof. Assume that $\rho_{R\Leftarrow}$ is an accepting run of $\mathcal{A}_{R\Leftarrow}$ on $T := \mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$. A straightforward induction from the root to the leaves shows that $c_1 = \mathrm{pop}_2{}^k(c_2)$, that Conditions C1 – C8 hold and that the rightmost leaf of T is labelled by some element of M. Furthermore, $\rho_{R\Leftarrow}(0) \in M$ and $(\pi_1(\rho_{R\Leftarrow}(0)), \pi_2(\rho_{R\Leftarrow}(0)), \pi_3(\rho_{R\Leftarrow}(0))) = (S, q_1, q_2)$ for $c_i = (q_i, s_i)$. Moreover, note that $\rho_{R\Leftarrow}(d) \in M$ and $\pi_1(\rho_{R\Leftarrow}(d)) = C$ implies $\mathrm{CLvl}(\mathrm{LStck}(d, c_1)) = 2$: if a transition at some node d labels the 0-successor $d0$ by C, then we always have $\pi_5(\rho_{R\Leftarrow}(d0)) = 2$. By construction, the transition of $\rho_{R\Leftarrow}$ applied at $d0$ enforces that $\pi_5(\rho_{R\Leftarrow}(d0))$ is the link level encoded in the tree at $d0$. Thus, the claim holds for all 0 successors. Moreover, a 1-successor is labelled by C only if its predecessor is also labelled by C. Thus, by induction on the distance to the first ancestor which is a 0-successor the claim holds also for 1-successors.

By induction from the leaves to the root, we show that Conditions C9 and C10 hold. This completes the proof, because $\rho_{R\Leftarrow}(0)$ then witnesses that there is a run from c_1 to c_2 not passing a substack of c_2 before its final configuration. For the base case, assume that $d \in T$ is a leaf labelled by some $\bar{q} \in M$. Depending on $\pi_1(\bar{q})$ we have the following cases.

- If $\pi_1(\bar{q}) = S$, then d is the rightmost leaf of $\mathrm{Enc}(c_2)$. Thus, $s_d^{\nearrow} = s_d = s_2$. Since ρ is an accepting run, it uses a transition of the form T4. Thus, $\pi_2(\bar{q}) = \pi_3(\bar{q})$ and Condition C9 is trivially satisfied.
- If $\pi_1(\bar{q}) = R$, ρ applies a transition of the form T9. Recall that d is a leaf whence $s_d = \mathrm{pop}_2(\mathrm{LStck}(d, c_1))$, $s_d^{\nearrow} = \mathrm{LStck}(d, c_1)$, and $\pi_6(\bar{q}) = \mathrm{Rt}(\mathrm{LStck}(d, c_1))$. Thus, the condition in T9 ensures that there is a run from $(\pi_2(\bar{q}), s_d^{\nearrow})$ to $(\pi_3(\bar{q}), s_d)$, i.e., Condition C9 holds.
- If $\pi_1(\bar{q}) = C$, then ρ applies a transition of the form T19. Since the collapse level of $s_d^{\nearrow}$ is 2, we conclude analogously to the previous case that Condition C9 holds.
- If $\pi_1(\bar{q}) = P$, the transition of $\rho_{R\Leftarrow}$ at d is of the form T15. Since $s_d^{\nearrow} = \mathrm{LStck}(d, c_1)$ the conditions of T15 ensure there exists some stack s' with $s_d^{\nearrow} \leq s'$ and $\mathrm{top}_2(s') = \mathrm{top}_2(s_d^{\nearrow})$

such that there is a 1-loop from $s_d^{\nearrow}$ to s' followed by a pop_1 operation or a collapse of level 1. Note that $s_d \leq \mathrm{pop}_1(s')$ and $\mathrm{top}_2(s_d) = \mathrm{top}_2(\mathrm{pop}_1(s'))$. Thus, Condition C10 is satisfied.

A tedious but easy case distinction shows that Conditions C9 and C10 carry over to all nodes of T. Instead of giving the full case distinction, we mention briefly the underlying ideas.

(1) If $d0 \in \mathrm{Enc}(c_1) \setminus \mathrm{Enc}(c_2)$, $d1 \in \mathrm{Enc}(c_1) \setminus \mathrm{Enc}(c_2)$ and $\rho_{R\Leftarrow}(d0) \in M$, then also $\rho_{R\Leftarrow}(d1) \in M$ and $\pi_1(\rho_{R\Leftarrow}(d1)) = R$ whence $s_{d1} = s_{d0}^{\nearrow}$. Thus, we can compose the run associated to $d1$ with the run associated to $d0$ and obtain a run associated to d.

(2) If $i \in \{0,1\}$ minimal such that $di \in \mathrm{Enc}(c_1)$, then either $s_{di} = s_d$ such that the final part of the run associated to s_{di} can serve as final part of the run associated to s_d or $s_{di} = \mathrm{LStck}(d, c_1)$ and the conditions on the transition at d ensure that this run can be extended to a run to s_d if $\pi_1(\rho_{R\Leftarrow}(d)) \neq P$. If $\pi_1(\rho_{R\Leftarrow}(d)) = P$, this run can be extended to some stack s' with $s_d \leq s'$ and $\mathrm{top}_2(s_d) = \mathrm{top}_2(s')$.

(3) If $i \in \{0,1\}$ is maximal such that $di \in \mathrm{Enc}(c_1)$ then $s_d^{\nearrow} = s_{di}^{\nearrow}$ and the run associated to di may serve as initial part of the run associated to d. $\qquad\square$

Lemma B.3. *Let $c_1 = (q, s_1), c_2 = (q', s_2)$ be configurations such that $s_2 = \mathrm{pop}_2^k(s_1)$ and there is a run from c_1 to c_2 that passes a substack of s_2 only in its final configuration. Then there is an accepting run of $\mathcal{A}_{R\Leftarrow}$ on $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$.*

Proof. Let ρ be some run from c_1 to c_2. Recall the decomposition $\rho = \rho_1 \circ \rho_2 \circ \cdots \circ \rho_n$ provided by Lemma 4.17. Let $\rho_0 := \rho\restriction_{[0,0]}$ and let q_i denote the final state of ρ_i for all $0 \leq i \leq n$. In the following, we will use the notation $\hat{d} := \mathrm{LStck}(d, c_1)$ for all $d \in \{0,1\}^*$. Let d be the rightmost leaf of $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$. Note that

$$\hat{d} = s_1 = \rho(0) = \rho_0(0) = \rho_0(\mathrm{length}(\rho_0))$$

whence ρ_0 ends in $(q_0, \hat{d})$. We define an accepting run $\rho_{R\Leftarrow}$ of $\mathcal{A}_{R\Leftarrow}$ on $T := \mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ by induction as follows. Let $d \in T$ be the lexicographically maximal node of $\mathrm{Enc}(c_1)$ such that $\rho_{R\Leftarrow}$ has not been defined yet at d. Assume that there is some maximal $i \geq 1$ such that $\rho_i(0) = (q_{i-1}, s)$ for some stack s satisfying $\mathrm{pop}_2(\hat{d}) \leq \mathrm{pop}_2(s)$ and $\mathrm{top}_2(\hat{d}) = \mathrm{top}_2(s)$. Furthermore, assume that $s = \hat{d}$ if ρ_{i-1} is not of the form F3. Depending on the form of ρ_i, we proceed as follows.

(1) If ρ_i is of the Form F1, let d' be the minimal element such that $d = d'0^m$ for some $m \in \mathbb{N}$. We set $\rho_{R\Leftarrow}(d) := (R, q_{i-1}, q_i, \mathrm{Sym}(\hat{d}), \mathrm{CLvl}(\hat{d}), \mathrm{Rt}(\hat{d}))$ and for $d' \leq e < d$ we set $\rho_{R\Leftarrow}(e) := (R, q_e, q_i, \mathrm{Sym}(\hat{e}), \mathrm{CLvl}(\hat{e}), \mathrm{Rt}(\hat{e}))$ where $q_e = \pi_2(\rho_{R\Leftarrow}(ej))$ for $j = \max\{i \in \{0,1\} : di \in \mathrm{Enc}(c_1)\}$.

(2) If ρ_i is of the Form F2, let e' be minimal such that $d = e'0^{m_0}1^{m_1}$ for some $m_0, m_1 \in \mathbb{N}$. For each e satisfying $e'0^{m_0} \leq e \leq d$ and for all $e0 \leq f \in \mathrm{Enc}(c_1)$ we set $\rho_{R\Leftarrow}(e) := (C, q_{i-1}, q_i, \mathrm{Sym}(\hat{e}), \mathrm{CLvl}(\hat{e}), \mathrm{Rt}(\hat{e}))$ and $\rho_{R\Leftarrow}(f) := q_\emptyset$.
    For all e with $e' \leq e < e'0^{m_0}$ we define $\rho_{R\Leftarrow}(e) := (R, q_e, q_i, \mathrm{Sym}(\hat{e}), \mathrm{CLvl}(\hat{e}), \mathrm{Rt}(\hat{e}))$ where q_e is defined as in the previous case.

(3) If ρ_i is of the Form F3, then we proceed as follows.
    - If d is a leaf of $\mathrm{Enc}(c_1)$, set $\rho_{R\Leftarrow}(d) := (P, q_{i-1}, q_i, \mathrm{Sym}(\hat{d}), \mathrm{CLvl}(\hat{d}), \mathrm{Rt}(\hat{d}))$;
    - otherwise, let $j \in \{0,1\}$ be maximal such that $dj \in \mathrm{Enc}(c_1)$. Then we set $\rho_{R\Leftarrow}(d) := \left(P, \pi_2(\rho_{R\Leftarrow}(ej)), q_i, \mathrm{Sym}(\hat{d}), \mathrm{CLvl}(\hat{d}), \mathrm{Rt}(\hat{d})\right)$.

In case that there is some $e \in \{0,1\}^*$ such that $d = e1$, then define $\rho_{R\Leftarrow}(e0f) := q_\emptyset$ for all $f \in \{0,1\}^*$ such that $e0f \in \mathrm{Enc}(c_1)$.

These rules define $\rho_{R\Leftarrow}$ on $\mathrm{Enc}(c_1) \setminus \mathrm{Enc}(c_2)$. Let d be the rightmost leaf of $\mathrm{Enc}(c_2)$, and set $\rho_{R\Leftarrow}(d) := \left(S, q_n, q_n, \mathrm{Sym}(\hat{d}), \mathrm{CLvl}(\hat{d}), \mathrm{Rt}(\hat{d})\right)$. Let $0 \leq d$ be the maximal element in the rightmost branch of $\mathrm{Enc}(c_2)$ such that $\rho_{R\Leftarrow}(d)$ is undefined. Let $j \in \{0,1\}$ be maximal such that $dj \in \mathrm{Enc}(c_1)$. We set $\rho_{R\Leftarrow}(d) := \left(S, \pi_2(\rho_{R\Leftarrow}(dj)), q_n, \mathrm{Sym}(\hat{d}), \mathrm{CLvl}(\hat{d}), \mathrm{Rt}(\hat{d})\right)$. We complete the definition by $\rho_{R\Leftarrow}(\varepsilon) = q_I$ and $\rho_{R\Leftarrow}(d) := q_=$ for all $d \in \mathrm{Enc}(c_2)$ that are not in the rightmost branch of $\mathrm{Enc}(c_2)$. A tedious, but straightforward induction shows that $\rho_{R\Leftarrow}$ is an accepting run of $\mathcal{A}_{R\Leftarrow}$ on $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$. $\qquad\square$

Appendix C. Automaton for Relation $R^\Downarrow$

In the following definition, w ranges over words, τ over letters from $\Sigma \setminus \{\bot\}$, k over $\{1,2\}$, q_i, q_e, q_e' over Q and z, z' over $\{q_=, \bot\}$. Whenever we have fixed a word w, then $\sigma := \mathrm{Sym}(w)$, $l := \mathrm{CLvl}(w)$ and x ranges over $\{(\sigma, l), \varepsilon\}$.

Definition C.1. $\mathcal{A} := (Q_\mathcal{A}, \Sigma_\mathcal{A}, \bot, \{q_I\}, \Delta_\mathcal{A})$ where

- $\Sigma_\mathcal{A} := (Q \cup (\Sigma \times \{1,2\}) \cup \{\varepsilon, \square\})^2$,
- $Q_\mathcal{A} := \{q_I, \bot, q_=, (\square, \varepsilon)\} \cup (Q \times Q \times 2^{Q \times Q} \times \Sigma \times \{1,2\} \times \{S, P_1, P_2\})$, and
- $\Delta_\mathcal{A}$ contains the following transitions:
  - (a) $(q_I, (q_1, q_2), (q_1, q_2, \emptyset, \bot, 1, S), \bot)$;
  - (b) $(q_=, (y, y), z, z')$ for all $y \in (\Sigma \times \{1,2\}) \cup \{\varepsilon\}$;
  - (c) $((\square, \varepsilon), (\square, \varepsilon), \bot, \bot)$;
  - now fix an arbitrary $\bar{q} = (q_i, q_e, \mathrm{Rt}(\mathrm{pop}_1(w)), \mathrm{Sym}(w), \mathrm{CLvl}(w), S)$. $\Delta_\mathcal{A}$ contains
  - (a) $(\bar{q}, (x, x), \bar{q}_0, \bot)$ for each $\bar{q}_0 = (q_i, q_e, \mathrm{Rt}(w), \tau, k, S)$;
  - (b) $(\bar{q}, (x, x), z, \bar{q})$;
  - (c) $(\bar{q}, (x, x), \bot, \bot)$ if $q_i = q_e$;
  - (d) $(\bar{q}, (x, x), \bar{q}_0^2, \bot)$ and $(\bar{q}, (x, x), \bar{q}_0^1, (\square, \varepsilon))$ for $\bar{q}_0^j = (q_i, q_e', \mathrm{Rt}(w), \tau, k, P_j)$ such that there is some $q \in Q$ with $(q_e', q) \in \mathrm{hLp}(w\tau_k)$ and $(q, \tau, q_e, \mathrm{ColPop}_k) \in \Delta$;
  - now fix an arbitrary $\bar{q} = (q_i, q_e, \mathrm{Rt}(\mathrm{pop}_1(w)), \mathrm{Sym}(w), \mathrm{CLvl}(w), P_2)$. $\Delta_\mathcal{A}$ contains
  - (a) $(\bar{q}, (x, \square), \bot, \bot)$ if $q_i = q_e$;
  - (b) $(\bar{q}, (x, \square), \bar{q}_0, \bot)$ for $\bar{q}_0 = (q_i, q_e', \mathrm{Rt}(w), \tau, k, P_2)$ such that there is a $q \in Q$ with $(q_e', q) \in \mathrm{hLp}(w\tau_k)$ and $(q, \tau, q_e, \mathrm{ColPop}_k) \in \Delta$;
  - now fix an arbitrary $\bar{q} = (q_i, q_e, \mathrm{Rt}(\mathrm{pop}_1(w)), \mathrm{Sym}(w), \mathrm{CLvl}(w), P_1)$. $\Delta_\mathcal{A}$ contains
  - (a) $(\bar{q}, (x, x), \bar{q}_0, \bot)$ for $\bar{q}_0 = (q_i, q_e', \mathrm{Rt}(w), \tau, k, P_1)$ such that there is a $q \in Q$ with $(q_e', q) \in \mathrm{hLp}(w\tau_k)$ and $(q, \tau, q_e, \mathrm{ColPop}_k) \in \Delta$;
  - (b) $(\bar{q}, (x, x), z, \bar{q})$;
  - (c) $(\bar{q}, (x, x), z, \bar{q}_1)$ for $\bar{q}_1 = (q_i, q_e, \mathrm{Rt}(\mathrm{pop}_1(w)), \sigma, l, P_2)$.

Let us explain the use of the flags S ('**S**earching the rightmost leaf of the second input'), P_1 and P_2 ('**P**op sequence'). Let $c = (q, s)$ and $c' = (p, t)$ be configurations such that $t = \mathrm{pop}_1{}^k(s)$. Then we can always define nodes d_1, d_2, d_3 (and an auxiliary node e_3) in the convolution of $\mathrm{Enc}(c) \otimes \mathrm{Enc}(c')$ as follows. Let d_3 be the rightmost leaf of $\mathrm{Enc}(c)$, let e_3 be the rightmost leaf of $\mathrm{Enc}(c')$, let d_2 be the minimal node of the rightmost path of $\mathrm{Enc}(c)$ which is not in $\mathrm{Enc}(c') \setminus \{e_3\}$ and let d_1 be the maximal node of the rightmost path of $\mathrm{Enc}(c)$ which is on the rightmost path of $\mathrm{Enc}(c')$. See Figure 4 for an example. By definition one
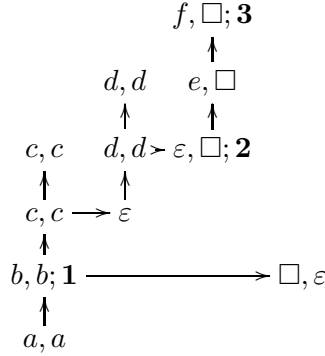
$$
\begin{array}{c}
f, \square; \mathbf{3} \\
\uparrow \\
d, d \qquad e, \square \\
\uparrow \qquad \uparrow \\
c, c \qquad d, d \succ \varepsilon, \square; \mathbf{2} \\
\uparrow \qquad \uparrow \\
c, c \longrightarrow \varepsilon \\
\uparrow \\
b, b; \mathbf{1} \longrightarrow \square, \varepsilon \\
\uparrow \\
a, a
\end{array}
$$

Figure 4: Nodes d_1, d_2, d_3 in case of $s = abcc : abcdd : abcdef$ and $t = abcc : abcdd : ab$ are marked by boldface numbers $\mathbf{1}, \mathbf{2}, \mathbf{3}$, respectively.

concludes that $d_1 \leq d_2 \leq d_3$. An accepting run of $\mathcal{A}$ labels all nodes up to d_1 with flag S, the nodes strictly between d_1 and d_2 with P_1 and the nodes between d_2 and d_3 by P_2. Using these flags the automaton guarantees that $t = \mathrm{pop}_1{}^k(s)$ for some s. Furthermore the transitions used at the nodes labelled by P_1 or P_2 guarantee that there is a sequence of loops and pop operations connecting the two configurations.

Lemma C.2. *Let c_1 and c_2 be configurations. $\mathcal{A}_{R^{\Downarrow}}$ accepts $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ if and only if $c_2 = \mathrm{pop}_1{}^m(c_1)$ such that $(c_1, c_2) \in R^{\Downarrow}$.*

Proof (sketch). Assume that ρ is an accepting run of $\mathcal{A}_{R^{\Downarrow}}$ on $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$.

Every accepting run labels the root by q_I. Now an easy induction shows that there are nodes $0 \leq d_1 \leq d_2 \leq d_3$ such that the following holds.

- d_3 is the rightmost leaf of $\mathrm{Enc}(c_1)$.
- All nodes $0 \leq e \leq d_3$ are labelled by elements in

$$
M := Q \times Q \times 2^{Q \times Q} \times \Sigma \times \{1, 2\} \times \{S, P_1, P_2\}
$$

such that $\pi_6(e) = \begin{cases} S & \text{if } 0 \leq e \leq d_1, \\ P_1 & \text{if } d_1 < e \leq d_2, \\ P_2 & \text{if } d_2 < e \leq d_3. \end{cases}$

Furthermore, all nodes in $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ to the left of this branch are labelled by $q_=$ which ensures that the two configurations agree on these nodes. We distinguish the following cases:

$d_1 = d_2 = d_3$ In this case, no transition of the form (?d) is used in the run. One easily concludes that $c_1 = c_2$ and $(c_1, c_2) \in R^{\Downarrow}$ is witnessed by the run of length 0 connecting c_1 with c_2.

$d_1 = d_2 < d_3$ In this case, we prove by induction from d_1 to d_3 that the automaton uses at d_1 a transition of the first form of (?d), between d_1 and d_3 it uses transitions from (?b), and at d_3 it uses a transition from (?a). Due to Lemma 5.8 (first case) this implies that $c_2 = \mathrm{pop}_1{}^m(c_1)$ for some $m \in \mathbb{N}$ and $\mathrm{LStck}(d_1, c_1)$ is the stack of c_2. Now, by induction from d_3 to d_1 one proves for each $d_1 \leq e \leq d_3$ that $\pi_1(e)$ is the state of c_1 and $\pi_2(e)$ is a state such that there is a run witnessing $(c_1, (\pi_2(e), \mathrm{LStck}(e, c_1))) \in R^{\Downarrow}$. We conclude by another induction showing that $\pi_2(e)$ is the state of c_2.

$d_1 < d_2 < d_3$ Analogously to the previous case, we use Lemma 5.8 (second case) to show that $c_2 = \mathrm{pop}_1{}^m(c_1)$ for some $m \in \mathbb{N}$. Induction from d_3 to d_1 shows that for each $d_1 \leq e \leq d_3$ there is some number $k(e) \leq m$ such that

$$\mathrm{top}_2(\mathrm{LStck}(e, c_1)) = \mathrm{top}_2(\mathrm{pop}_1{}^{k(e)}(c_1))$$

and the label of e is such that there is a run from c_1 to $(\pi_2(e), \mathrm{pop}_1{}^{k(e)}(c_1))$ witnessing that this pair is in $R^{\Downarrow}$. Moreover, $k(d_1) = m$ and $\pi_2(d_1)$ is the state of c_2 whence $(c_1, c_2) \in R^{\Downarrow}$.

For the other direction let $c_1 = (q_1, s_1), c_2 = (q_2, s_2)$ be configurations and ρ a run that witnesses $(c_1, c_2) \in R^{\Downarrow}$. We only consider the case that $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ is as described in (2) of Lemma 5.8. The other case is similar. Let $b \in \mathrm{Enc}(c_2)$ be such that $b1$ is the rightmost leaf of $\mathrm{Enc}(c_2)$. Let c be maximal in $\mathrm{Enc}(c_2)$ such that $c1 \in \mathrm{Enc}(c_1) \setminus \mathrm{Enc}(c_2)$. Let d be the rightmost leaf of $\mathrm{Enc}(c_1)$. Due to Lemma 5.8, $b < c < d$.

For all $x \leq d$, let $w_x := \mathrm{top}_2(\mathrm{LStck}(x, \mathrm{Enc}(c_1)))$. For $b \leq x \leq d$ let $i_x \in \mathrm{dom}(\rho)$ be minimal such that $\rho(i_x) = (q, \mathrm{pop}_2(s_1) : w_x)$. Let $q_x \in Q$ be the state at $\rho(i_x)$. We define an accepting run λ of $\mathcal{A}_{R^{\Downarrow}}$ on $\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$ as follows. For all $\varepsilon < x \leq d$ let

$$\lambda(x) := (q_1, f_x, \mathrm{Rt}(\mathrm{pop}_1(w_x)), \mathrm{Sym}(w_x), \mathrm{CLvl}(w_x), Y_x) \text{ where}$$

$$f_x = \begin{cases} q_2 & \text{if } x \leq b, \\ q_x & \text{if } b < x \leq d, \end{cases} \quad \text{and}$$

$$Y_x = \begin{cases} S & \text{if } x \leq b, \\ P_1 & \text{if } b < x \leq c, \\ P_2 & \text{if } c < x \leq d. \end{cases}$$

Furthermore, we set $\lambda(b1) := (\square, \varepsilon)$, $\lambda(\varepsilon) := q_I$ and $\lambda(x) := q_=$ for all other nodes $x \in \mathrm{Enc}(c_1) \otimes \mathrm{Enc}(c_2)$.

Since ρ decomposes as a sequence of loops, pop_1 operations and collapse operations of level 1, it is straightforward to show that λ is an accepting run of $\mathcal{A}_{R^{\Downarrow}}$. $\square$

APPENDIX D. AUTOMATON FOR RELATION $R^{\Rightarrow}$

In the following definition, we use the same convention regarding ranges of variables as in Appendix B.

Before we define the automaton recognising the relation $R^{\Rightarrow}$ formally, we explain how a successful run of it will process a tree $\mathrm{Enc}(q_1, s_1) \otimes \mathrm{Enc}(q_2, s_2)$. The states of $\mathcal{A}_{R^{\Rightarrow}}$ come from the set $\{q_I, q_=, \bot\} \cup M$ where

$$M := Q \times Q \times \Sigma \times \{1, 2\} \times (2^{Q \times Q}) \times (2^{Q \times Q}) \times \{R, L\} \times \{S, N\}.$$

q_I is the final state that is exclusively used to label the root. $q_=$ is the state for all nodes in $\mathrm{Enc}(q_1, s_1)$ that do not belong to the rightmost branch of this tree. This state is used to check that $\mathrm{Enc}(q_1, s_1)$ and $\mathrm{Enc}(q_2, s_2)$ agree on this part of the convolution. The state $\bot$ is the initial state only used for marking the end of the tree, i.e., $\bot$ is the label for the nodes in $(\mathrm{Enc}(q_1, s_1) \otimes \mathrm{Enc}(q_2, s_2))_+$. The rest of the nodes are labelled by elements from M.

For some $\bar{q} \in M$ we write $\pi_i(\bar{q})$ for the projection to the i-th component. In an accepting run, a node d is labelled by $\bar{q} \in M$ if the following is satisfied.

C1. $\pi_8(\bar{q}) = S$ iff d is in the rightmost branch of $\mathrm{Enc}(q_1, s_1)$. S stands for "searching the rightmost leaf of (q_1, s_1)" while N stands for "normal reachability". We ensure that $\pi_8(\bar{q}) = N$ iff d is in $\mathrm{Enc}(q_2, s_2) \setminus \mathrm{Enc}(q_1, s_1)$.

C2. $\pi_7(\bar{q}) = R$ iff d is in the rightmost branch of $\mathrm{Enc}(q_2, s_2)$ (by definition this is also the rightmost branch of $\mathrm{Enc}(q_1, s_1) \otimes \mathrm{Enc}(q_2, s_2)$. R stands for "rightmost branch" while L stands for "left".

C3. $\pi_6(\bar{q}) = \mathrm{Lp}(\mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2)))$. Since $s_1 \in \mathrm{MS}(s_2)$, $d \in \mathrm{Enc}(q_1, s_1)$ implies that $\pi_6(\bar{q}) = \mathrm{Lp}(\mathrm{LStck}(d, \mathrm{Enc}(q_1, s_1)))$.

C4. $\pi_5(\bar{q}) = \mathrm{Rt}(\mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2)))$. Since $s_1 \in \mathrm{MS}(s_2)$, $d \in \mathrm{Enc}(q_1, s_1)$ implies that $\pi_5(\bar{q}) = \mathrm{Rt}(\mathrm{LStck}(d, \mathrm{Enc}(q_1, s_1)))$.

C5. $\pi_4(\bar{q}) = \mathrm{CLvl}(\mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2)))$.

C6. $\pi_3(\bar{q}) = \mathrm{Sym}(\mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2)))$.

C7. Let $q_i := \pi_1(\bar{q})$ and $q_e := \pi_2(\bar{q})$. For $d \in \mathrm{Enc}(q_2, s_2) \setminus \mathrm{Enc}(q_1, s_1)$ there is a run from $(q_i, \mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2)))$ to $(q_e, \mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)))$. If d is in the rightmost path of $\mathrm{Enc}(q_1, s_1)$, $q_i = q_1$ and there is a run from (q_1, s_1) to $(q_e, \mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)))$.

Definition D.1. Let $\mathcal{S} = (Q, \Sigma, \Gamma, \Delta_{\mathcal{S}}, q_0)$ be some CPS. Define the automaton $\mathcal{A}_{R\Rightarrow}$ as follows. The set of states is contained in $\{q_I, q_=, \bot\} \cup M$ where

$$M := Q \times Q \times \Sigma \times \{1, 2\} \times (2^{Q \times Q}) \times (2^{Q \times Q}) \times \{R, L\} \times \{S, N\}.$$

$\bot$ is the initial state and q_I is the only final state. The transition relation $\Delta_{\mathcal{A}}$ of $\mathcal{A}$ contains the following transitions.

T1. $(q_I, (q_1, q_2), (q_1, q_2, \bot, 1, \mathrm{Rt}(\bot), \mathrm{Lp}(\bot), R, S), \bot) \in \Delta_{\mathcal{A}}$ for all pairs $(q_1, q_2) \in Q^2$, and

T2. $(q_=, (y, y), X, Y)$ for $X, Y \in \{q_=, \bot\}$ and $y \in (\Sigma \times \{1, 2\}) \cup \{\varepsilon\}$.

Fix some $\bar{q} := (q_1, q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), R, S)$. Then we add the following transitions.

T3. $(\bar{q}, (x, x), \bot, \bot)$ if $q_1 = q_2$;

T4. $(\bar{q}, (x, x), \bot, \bar{q}_1)$ for

$$(q_1, \sigma, \gamma, q, \mathrm{clone}_2) \in \Delta_{\mathcal{S}},$$
$$(q, q_1') \in \mathrm{Lp}(w), \text{ and}$$
$$\bar{q}_1 = (q_1', q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), R, N);$$

T5. $(\bar{q}, (x, x), X, \bar{q})$ for $X \in \{q_=, \bot\}$;

T6. $(\bar{q}, (x, x), \bar{q}_0, \bot)$ for $\bar{q}_0 := (q_1, q_2, \tau, k, \mathrm{Rt}(w\tau_k), \mathrm{Lp}(w\tau_k), R, S)$;

T7. $(\bar{q}, (x, x), \bar{q}_0, \bar{q}_1)$ for

$$\bar{q}_0 := (q_1, q_2', \tau, k, \mathrm{Rt}(w\tau_k), \mathrm{Lp}(w\tau_k), L, S),$$
$$(q_2', \tau, \gamma, q, \mathrm{ColPop}_k) \in \Delta_{\mathcal{S}},$$
$$(q, q_1') \in \mathrm{Lp}(w), \text{ and}$$
$$\bar{q}_1 := (q_1', q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), R, N);$$

Fix some $\bar{q} := (q_1, q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), L, S)$. Then we add the following transitions.

T8. $(\bar{q}, (x, x), \bot, \bot)$ for $(q_1, \sigma, \gamma, q, \mathrm{clone}_2) \in \Delta_{\mathcal{S}}$ and $(q, q_2) \in \mathrm{Lp}(w)$;

T9. $(\bar{q}, (x,x), \bar{q}_0, \bot)$ for

$$\bar{q}_0 := (q_1, q_2', \tau, k, \mathrm{Rt}(w\tau_k), \mathrm{Lp}(w\tau_k), L, S),$$
$$(q_2', \tau, q, \mathrm{ColPop}_k) \in \Delta_{\mathcal{S}} \text{ and}$$
$$(q, q_2) \in \mathrm{Lp}(w);$$

T10. $(\bar{q}, (x,x), X, \bar{q})$ for $X \in \{q_=, \bot\}$;

T11. $(\bar{q}, (x,x), \bot, \bar{q}_1)$ for

$$(q_1, \sigma, \gamma, q, \mathrm{clone}_2) \in \Delta_{\mathcal{S}},$$
$$(q, q_1') \in \mathrm{Lp}(w) \text{ and}$$
$$\bar{q}_1 := (q_1', q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), L, N);$$

T12. $(\bar{q}, (x,x), \bar{q}_0, \bar{q}_1)$ for

$$\bar{q}_0 := (q_1, q_2', \tau, k, \mathrm{Rt}(w\tau_k), \mathrm{Lp}(w\tau_k), L, S),$$
$$(q_2', \tau, q, \mathrm{ColPop}_k) \in \Delta_{\mathcal{S}},$$
$$(q, q_1') \in \mathrm{Lp}(w) \text{ and}$$
$$\bar{q}_1 := (q_1', q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), L, N).$$

Fix some $\bar{q} := (q_1, q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), R, N)$. Then we add the following transitions.

T13. $(\bar{q}, (\Box, x), \bot, \bot)$ if $q_1 = q_2$;

T14. $(\bar{q}, (\Box, x), \bar{q}_0, \bot)$ for

$$(q_1, \sigma, \gamma, q, \mathrm{push}_{\tau,k}) \in \Delta_{\mathcal{S}},$$
$$(q, q_1') \in \mathrm{Lp}(w\tau_k) \text{ and}$$
$$\bar{q}_0 := (q_1', q_2, \tau, k, \mathrm{Rt}(w\tau_k), \mathrm{Lp}(w\tau_k), R, N);$$

T15. $(\bar{q}, (\Box, x), \bot, \bar{q}_1)$ for

$$(q_1, \sigma, \gamma, q, \mathrm{clone}_2) \in \Delta_{\mathcal{S}},$$
$$(q, q_1') \in \mathrm{Lp}(w) \text{ and}$$
$$\bar{q}_1 := (q_1', q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), R, N);$$

T16. $(\bar{q}, (\Box, x), \bar{q}_0, \bar{q}_1)$ for

$$(q_1, \sigma, \gamma, q, \mathrm{push}_{\tau,k}) \in \Delta_{\mathcal{S}},$$
$$(q, q_1') \in \mathrm{Lp}(w\tau_k),$$
$$\bar{q}_0 := (q_1', q_2', \tau, k, \mathrm{Rt}(w\tau_k), \mathrm{Lp}(w\tau_k), L, N),$$
$$(q_2', \tau, q', \mathrm{ColPop}_k) \in \Delta_{\mathcal{S}},$$
$$(q', q_1'') \in \mathrm{Lp}(w) \text{ and}$$
$$\bar{q}_1 := (q_1'', q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), R, N).$$

Fix some $\bar{q} := (q_1, q_2, \sigma, l, \mathrm{Rt}(w), \mathrm{Lp}(w), L, N)$. Then we add the following transitions.

T17. $(\bar{q}, (\Box, x), \bot, \bot)$ for $(q_1, \sigma, \gamma, q, \mathrm{clone}_2) \in \Delta_{\mathcal{S}}$ and $(q, q_2) \in \mathrm{Lp}(w)$;

T18. $(\bar{q},(\square,x),\bar{q}_0,\bot)$ for

$$(q_1,\sigma,\gamma,q,\mathrm{push}_{\tau,k})\in\Delta_{\mathcal{S}},$$
$$(q,q_1')\in\mathrm{Lp}(w\tau_k),$$
$$\bar{q}_0:=(q_1',q_2',\tau,k,\mathrm{Rt}(w\tau_k),\mathrm{Lp}(w\tau_k),L,N),$$
$$(q_2',\tau,q',\mathrm{ColPop}_k)\in\Delta_{\mathcal{S}}\text{ and}$$
$$(q',q_2)\in\mathrm{Lp}(w);$$

T19. $(\bar{q},(\square,x),\bot,\bar{q}_1)$ for

$$(q_1,\sigma,\gamma,q,\mathrm{clone}_2)\in\Delta_{\mathcal{S}},$$
$$(q,q_1')\in\mathrm{Lp}(w),\text{ and}$$
$$\bar{q}_1:=(q_1',q_2,\sigma,l,\mathrm{Rt}(w),\mathrm{Lp}(w),L,N);$$

T20. $(\bar{q},(\square,x),\bar{q}_0,\bar{q}_1)$ for

$$(q_1,\sigma,\gamma,q,\mathrm{push}_{\tau,k})\in\Delta_{\mathcal{S}},$$
$$(q,q_1')\in\mathrm{Lp}(w\tau_k),$$
$$\bar{q}_0:=(q_1',q_2',\tau,k,\mathrm{Rt}(w\tau_k),\mathrm{Lp}(w\tau_k)),L,N),$$
$$(q_2',\tau,q',\mathrm{ColPop}_k)\in\Delta_{\mathcal{S}},$$
$$(q',q_1'')\in\mathrm{Lp}(w)\text{ and}$$
$$\bar{q}_1:=(q_1'',q_2,\sigma,l,\mathrm{Rt}(w),\mathrm{Lp}(w),L,N).$$

The next lemma is a first step towards the proof that any accepting run of $\mathcal{A}_{R^\Rightarrow}$ on a tree $\mathrm{Enc}(c_1)\otimes\mathrm{Enc}(c_2)$ witnesses the existence of some run from c_1 to c_2.

Lemma D.2. *Let $\mathcal{S}$ be some* CPS. *Let $(q_1,s_1),(q_2,s_2)$ be configurations and let ρ be an accepting run on $T:=\mathrm{Enc}(q_1,s_1)\otimes\mathrm{Enc}(q_2,s_2)$. Then Conditions C1–C6 of the beginning of this section hold and $s_1\in\mathrm{MS}(s_2)$.*

The proof consists of straightforward inductions.

Lemma D.3. *Let $\mathcal{S}$ be some* CPS. *Let $(q_1,s_1),(q_2,s_2)$ be configurations and let ρ be an accepting run on $T:=\mathrm{Enc}(q_1,s_1)\otimes\mathrm{Enc}(q_2,s_2)$. Let $T_1:=dom(T)\setminus dom(\mathrm{Enc}(q_1,s_1))$ and T_2 be the rightmost branch of $\mathrm{Enc}(q_1,s_1)$ without the root. Furthermore, for all $d\in T_1$, let $s_d:=\mathrm{LStck}(d,\mathrm{Enc}(q_2,s_2))$ and for each $d\in T_2$, let $s_d:=s_1$. For each $d\in T_1\cup T_2$ we have $\rho(d)\in M$ and there is a run $\rho_{\mathcal{S}}$ of $\mathcal{S}$ from $(\pi_1(\rho(d)),s_d)$ to $(\pi_2(\rho(d)),\mathrm{IgM}(d,\mathrm{Enc}(q_2,s_2)))$ such that for all $0<i\le\mathrm{length}(\rho_{\mathcal{S}})$, $\rho_{\mathcal{S}}(i)\ne s_1$.*

Proof. The proof is by induction starting at the leaves. The base cases are the following.

- Assume that $s_1=s_2$. Due to C1 and C2, the rightmost leaf d of T satisfies $(\pi_7(d),\pi_8(d))=(R,S)$. Thus, ρ applies at d some transition of the form T3. Due to the existence of this transition, we conclude that $\pi_1(\rho(d))=\pi_2(\rho(d)))$. Since $s_d=\mathrm{IgM}(d,\mathrm{Enc}(q_2,s_2))=s_1$, we conclude there is a loop of length 0 from $(\pi_1(\rho(d)),s_d)$ to $(\pi_2(\rho(d)),\mathrm{IgM}(d,\mathrm{Enc}(q_2,s_2)))$.
- Assume that $s_1\ne s_2$. Let d be the rightmost leaf of T, i.e., d is the rightmost leaf of $\mathrm{Enc}(q_2,s_2)$. Due to C1 and C2, $(\pi_7(d),\pi_8(d))=(R,N)$. Thus, ρ applies a transition of the form T13 whence $\pi_1(\rho(d))=\pi_2(\rho(d))$. As in the previous case we obtain $s_d=\mathrm{IgM}(d,\mathrm{Enc}(q_2,s_2))=s_2$ and a run of length 0 connects $(\pi_1(\rho(d)),s_d)$ with $(\pi_2(\rho(d)),\mathrm{IgM}(d,\mathrm{Enc}(q_2,s_2)))$ because the two configurations agree.

Now let $d \in T_1$ be a leaf of T that is not in the rightmost branch. Due to C1 and C2, $(\pi_7(d), \pi_8(d)) = (L, N)$. Thus, ρ applies a transition of the form T17. Note that $s_d = \mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2))$ and $\mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)) = \mathrm{clone}_2(s_d)$. Due to the conditions on the existence of a transition of form T17 one immediately concludes that there is a run from $(\pi_1(\rho(d)), s_d)$ to $(\pi_2(\rho(d)), \mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)))$.

Now let d be the rightmost leaf in T that is in $\mathrm{Enc}(q_1, s_1)$. Since d is not in the rightmost branch of T and due to C1 and C2, $(\pi_7(d), \pi_8(d)) = (L, S)$. Thus, ρ applies a transition of the form T8. Note that $s_d = \mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2))$ and $\mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)) = \mathrm{clone}_2(s_d)$. Due to the conditions on the existence of a transition of form T8 one immediately concludes that there is a run from $(\pi_1(\rho(d)), s_d)$ to $(\pi_2(\rho(d)), \mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)))$.

Note that all the runs obtained in the base cases do not visit the stack s_1 except for the first configuration in the run associated to the rightmost leaf of $\mathrm{Enc}(q_1, s_1)$.

Analogously to the base case, the inductive step consists of a lengthy but rather straightforward case distinction. Instead of stating all cases, we mention the crucial ideas underlying the proof.

- For $d \in \{0,1\}^*$ and $i \in \{0,1\}$ such that di is in the rightmost branch of $\mathrm{Enc}(q_1, s_1)$, then $s_d = s_{di} = s_1$. Thus, the run associated to di serves as initial part of the run associated to d.
- For $d \in \mathrm{Enc}(q_2, s_2) \setminus \mathrm{Enc}(q_1, s_1)$ or d the rightmost leaf of $\mathrm{Enc}(q_1, s_1)$, let $i \in \{0,1\}$ minimal such that $di \in \mathrm{Enc}(q_2, s_2)$. Then $\mathrm{LStck}(d, (q_2, s_2))$ and $\mathrm{LStck}(di, (q_2, s_2))$ differ in one stack operation op. The transition of $\mathcal{A}_{R \Rightarrow}$ used at d ensures that there is a run from $(\pi_1(\rho(d)), \mathrm{LStck}(d, (q_2, s_2)))$ to $(\pi_1(\rho(di)), \mathrm{LStck}(di, (q_2, s_2)))$ that performs this operation op followed by a loop. The composition of this run with the run associated to di serves as initial part of the run associated to d.
- If d is in the rightmost branch of $\mathrm{Enc}(q_2, s_2)$ and $i \in \{0,1\}$ is maximal such that $di \in \mathrm{Enc}(q_2, s_2)$, then $\mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)) = \mathrm{IgM}(di, \mathrm{Enc}(q_2, s_2))$ whence the run associated to di serves as final part of the run associated to d.
- If $d1 \in \mathrm{Enc}(q_2, s_2)$, then $\mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)) = \mathrm{IgM}(d1, \mathrm{Enc}(q_2, s_2))$ whence the run associated to $d1$ serves as final part of the run associated to d.
- If d is not in the rightmost branch of $\mathrm{Enc}(q_2, s_2)$ and $d1 \notin \mathrm{Enc}(q_2, s_2)$, then we have $\mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)) = \mathrm{pop}_1(\mathrm{IgM}(d0, \mathrm{Enc}(q_2, s_2)))$. Furthermore, the transition of $\mathcal{A}_{R \Rightarrow}$ used at d ensures that there exists a run from $(\pi_2(\rho(d0)), \mathrm{IgM}(d0, \mathrm{Enc}(q_2, s_2))$ to $(\pi_2(\rho(d)), \mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)))$. This run serves as final part of the run associated to d.
- If $d0, d1 \in \mathrm{Enc}(q_2, s_2)$, then $\mathrm{LStck}(d1, (q_2, s_2)) = \mathrm{pop}_1(\mathrm{IgM}(d0, \mathrm{Enc}(q_2, s_2)))$. Furthermore, the existence of the transition of $\mathcal{A}_{R \Rightarrow}$ used at d ensures that there is a run from $(\pi_2(\rho(d0)), \mathrm{IgM}(d0, \mathrm{Enc}(q_2, s_2)))$ to $(\pi_1(\rho(d1)), \mathrm{LStck}(d1, (q_2, s_2)))$. This run is used to connect the initial part induced by $d0$ with the final part induced by $d1$ in order to obtain the run associated to d. $\square$

Remark D.4. Due to the transitions of the form T1 any accepting run of $\mathcal{A}_{R \Rightarrow}$ on a tree $\mathrm{Enc}(q_1, s_1) \otimes \mathrm{Enc}(q_2, s_2)$ satisfies $(\pi_1(\rho(0)), \pi_2(\rho(0))) = (q_1, q_2)$. Moreover, recall that $\mathrm{IgM}(0, \mathrm{Enc}(q_2, s_2)) = s_2$. Thus, the lemma implies that $((q_1, s_1), (q_2, s_2)) \in R^{\Rightarrow}$ if there is an accepting run of $\mathcal{A}_{R \Rightarrow}$ on $\mathrm{Enc}(q_1, s_1) \otimes \mathrm{Enc}(q_2, s_2)$.

Lemma D.5. *Let $\mathcal{S}$ be some CPS. Let $c_1 := (q_1, s_1)$, $c_2 := (q_2, s_2)$ be configurations such that $s_1 = \mathrm{pop}_2{}^k(s_2)$ for some $k \in \mathbb{N}$. Let $\rho_\mathcal{S}$ be a run from (q_1, s_1) to (q_2, s_2) witnessing $(c_1, c_2) \in R^{\Rightarrow}$. Then there is an accepting run of $\mathcal{A}$ on $T := \mathrm{Enc}(q_1, s_1) \otimes \mathrm{Enc}(q_2, s_2)$.*

Proof. We define the accepting run ρ as follows. Set $\rho(\varepsilon) := q_I$, $\rho(d) := \perp$ for all $d \in T_+$, $\rho(d) := q_=$ for all $d \in \mathrm{Enc}(q_1, s_1) \setminus B$ where B is the rightmost branch of $\mathrm{Enc}(q_1, s_1)$, and for all other $d \in \mathrm{dom}(T)$, set

$$\rho(d) := (q_i^d, q_e^d, \mathrm{Sym}(s'), \mathrm{CLvl}(s'), \mathrm{Rt}(s'), \mathrm{Lp}(s'), X, Y)$$

where $s' := \mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2))$ if $d \notin B$ and $s' = s_1$ if $d \in B$ and q_i^d, q_e^d, X and Y are defined as follows.

- $q_i^d = q_1$ if $d \in \mathrm{Enc}(q_1, s_1)$. Otherwise let $j \in \mathrm{dom}(\rho_S)$ be maximal such that $\rho_S(j) = (q, \mathrm{LStck}(d, \mathrm{Enc}(q_2, s_2)))$ for some $q \in Q$. Set $q_i^d := q$.
- Let $j \in \mathrm{dom}(\rho_S)$ be maximal such that $\rho_S(j) = (q, \mathrm{IgM}(d, \mathrm{Enc}(q_2, s_2)))$ for some $q \in Q$. Set $q_e^d := q$.
- Set $X = R$ if d is in the rightmost branch of T and set $X = L$ otherwise.
- Set $Y = S$ if d is in the rightmost branch of $\mathrm{Enc}(q_1, s_1)$ and set $Y = N$ otherwise.

A straightforward, but tedious induction shows that ρ is accepting on T. It relies on the decomposition result for runs witnessing $(c_1, c_2) \in R^\Rightarrow$ from Corollary 4.10. □

Appendix E. Modifications for the Proof of Proposition 3.9 (cf. page 36)

We replace the automaton $\mathcal{A}_{R\Leftarrow}$ in the construction of Reach on the product of the pushdown system with the automaton for the regular language L with the following version $\mathcal{A}_{R\Leftarrow'}$. Let Q_L be the states of a finite automaton recognising L, $i_0 \in Q_L$ be its initial state and $F \subseteq Q_L$ be its final states. We replace transitions of the form $(q_I, (q_1, q_2), (S, q_1, q_2, \perp, 1, \mathrm{Rt}(\perp_2)), \perp)$ by transitions of the form $(q_I, (q_1, q_2), (S, (q_1, i_0), (q_2, \hat{q}), \perp, 1, \mathrm{Rt}(\perp_2)), \perp)$ for $q_1, q_2 \in Q$ and $\hat{q} \in Q_L$. Constructing φ with $\mathcal{A}_{R\Leftarrow'}$ instead of $\mathcal{A}_{R\Leftarrow}$ ensures that T_1 encodes some configuration (q, s) where the state q is in Q, but it checks for runs starting in $((q, i_0), s)$.

Furthermore, we replace the automaton $\mathcal{A}_{R\Rightarrow}$ with the version $\mathcal{A}_{R\Rightarrow'}$ where we replace the transitions of the form $(q_I, (q_1, q_2), (q_1, q_2, \perp, 1, \mathrm{Rt}(\perp), \mathrm{Lp}(\perp), R, S), \perp) \in \Delta_{\mathcal{A}}$ with the transitions $(q_I, (q_1, q_2), ((q_1, \hat{q}), (q_2, q_f), \perp, 1, \mathrm{Rt}(\perp), \mathrm{Lp}(\perp), R, S), \perp) \in \Delta_{\mathcal{A}}$ for $q_1, q_2 \in Q$, $\hat{q} \in Q_L$ and $q_f \in F$. Constructing φ with $\mathcal{A}_{R\Rightarrow'}$ instead of $\mathcal{A}_{R\Rightarrow}$ ensures that T_2 encodes some configuration (q, s) where the state q is in Q, but it checks for runs ending in $((q, q_f), s)$ for some final state q_f of $\mathcal{A}_L$.