First-Order Logic on CPDA Graphs

Paweł Parys[⋆]

University of Warsaw, parys@mimuw.edu.pl

Abstract. We contribute to the question about decidability of first-order logic on configuration graphs of collapsible pushdown automata. Our first result is decidability of existential FO sentences on configuration graphs (and their ε-closures) of collapsible pushdown automata of order 3, restricted to reachable configurations. Our second result is undecidability of whole first-order logic on configuration graphs which are not restricted to reachable configurations, but are restricted to constructible stacks. Our third result is decidability of first-order logic on configuration graphs (for arbitrary order of automata) which are not restricted to reachable configurations nor to constructible stacks, under an alternative definition of stacks, called annotated stacks.

1 Introduction

Already in the 70's, Maslov ([1, 2]) generalized the concept of pushdown automata to higher-order pushdown automata (n-PDA) by allowing the stack to contain other stacks rather than just atomic elements. In the last decade, renewed interest in these automata has arisen. They are now studied not only as acceptors of string languages, but also as generators of graphs and trees. Knapik et al. [3] showed that the class of trees generated by deterministic n-PDA coincides with the class of trees generated by *safe* order-n recursion schemes (safety is a syntactic restriction on the recursion scheme). Driven by the question whether safety implies a semantical restriction to recursion schemes (which was recently proven [4, 5]), Hague et al. [6] extended the model of n-PDA to order-n collapsible pushdown automata (n-CPDA) by introducing a new stack operation called collapse (earlier, panic automata [7] were introduced for order 2), and proved that trees generated by n-CPDA coincide with trees generated by all order-n recursion schemes.

In this paper we concentrate on configuration graphs of these automata. In particular we consider their ε-closures, whose edges consist of an unbounded number of transitions rather than just single steps. The ε-closures of n-PDA graphs form precisely the Caucal hierarchy [8–10], which is defined independently in terms of MSO-interpretations and graph unfoldings. These results imply that the graphs have decidable MSO theory, and invite the question about decidability of logics in ε-closures of n-CPDA graphs.

[⋆] The author holds a post-doctoral position supported by Warsaw Center of Mathematics and Computer Science. Work supported by the National Science Center (decision DEC-2012/07/D/ST6/02443).

Unfortunately there is even a 2-CPDA graph that has undecidable MSO theory [6]. Kartzow showed that the ε-closures of 2-CPDA graphs are tree automatic [11], thus they have decidable first-order theory. This topic was farther investigated by Broadbent [12–15]. He proved that for order 3 (and higher) the FO theory starts to be undecidable. In fact it is undecidable already on (where n_m-CPDA denotes an n-CPDA in which we allow collapse links only of one order m):

- n_m-CPDA graphs restricted to reachable configurations,[1] when $n \geq 3$, and $3 \leq m \leq n$, and the formula is Σ_2, and
- n_m-CPDA graphs restricted to reachable configurations,[1] when $n \geq 4$, and $2 \leq m \leq n-2$, and the formula is Σ_1, and
- ε-closures[2] of 3_2-CPDA graphs, when the formula is Σ_2, and
- 3-CPDA graphs not restricted to reachable configurations (nor to stacks which are constructible from the empty one by a sequence of stack operation).

On the other side, he gives some small decidability results:

- for $n = 2$, FO is decidable even when extended by transitive closures of quantifier free formulae;
- FO is decidable on 3_2-CPDA graphs restricted to reachable configurations;
- Σ_1 formulae are decidable on ε-closures of n_n-CPDA graphs (for each n), and of 3_2-CPDA graphs.

In the current paper we complement this picture by three new results (answering questions stated by Broadbent). First, we prove that the existential (Σ_1) FO sentences are decidable on ε-closures of 3-CPDA graphs. This is almost proved in [15]: it holds under the assumption that the 3-CPDA is *luminous*, which means that after removing all order-3 collapse links from a two different reachable configurations, there are still different (that is, the targets of such links are uniquely determined by the structure of the stack). We prove that each 3-CPDA can be turned into an equivalent luminous one. The question whether Σ_1 formulae are decidable for n_{n-1}-CPDA and $n_{n,n-1}$-CPDA (allowing links of orders n and $n-1$) where $n \geq 4$, both with and without ε-closure, remains open.

Second, we prove (oppositely to the Broadbent's conjecture) that first-order logic is undecidable on 4-CPDA graphs not restricted to reachable configurations, but restricted to stacks constructible from the empty one by a sequence of stack operations (although not necessarily ever constructed by the particular CPDA in question). Our reduction is very similar to the one showing undecidability of 3-CPDA graphs not restricted to reachable configurations nor to constructible stacks.

Third, we prove that first-order logic is decidable (for each n) on n-CPDA graphs not restricted to reachable configurations nor to constructible stacks,

[1] Thus for their ε-closures as well.

[2] For ε-closures, it does not change anything whether we restrict to reachable configurations or not.

when stacks are represented as *annotated stacks*. This is an alternative representation of stacks of n-CPDA (defined independently in [16] and [17]), where in an atomic element, instead of an order-k link, we keep an order-k stack; the collapse operation simply recalls this stack stored in the topmost element. In the constructible case, annotated and CPDA stacks amount to the same thing (although the annotated variant offers some conveniences in expressing certain proofs), but in the unconstructible case there is an important difference. Whilst with an unconstructible CPDA stack each link is constrained to point to some stack below its source, in an annotated stack it can point to an arbitrary stack, completely unrelated to the original one. This shows up when we go back through a pop edge: in the classical case links in the appended stack point (potentially anywhere) inside our original stack, so we can use them to inspect any place in the stack. On the other hand, in the annotated case we can append an arbitrary stack, which does not give us any new information: in first-order logic we can refer only locally to some symbols near the top of the stack.

2 Preliminaries

We give a standard definition of an n-CPDA, using the "annotated stack" representation of stacks. We choose this representation because of Section 5, in which we talk about all configurations with such stacks. For Sections 3 and 4 we could choose the standard representation (with links as numbers) as well.

Given a number n (the order of the CPDA) and a stack alphabet Γ, we define the set of stacks as the smallest set satisfying the following. If $1 \leq k \leq n$ and $s_1, s_2, \ldots, s_m$ for $m \geq 1$ are $(k-1, n)$-stacks, then the sequence $[s_1, s_2, \ldots, s_m]$ is a (k, n)-stack. If $a \in \Gamma$, and $1 \leq k \leq n$, and s is a (k, n)-stack or $s = []$ (the "empty stack"), then (a, k, s^k) is a $(0, n)$-stack. We sometimes use "k-stack" instead of "(k, n)-stack" when n is clear from the context or meaningless.

A 0-stack (a, l, t) is also called an *atom*; it has label $\mathsf{lb}((a, l, t)) := a$ and link t of order l. In a k-stack $s = [s_1, s_2, \ldots, s_m]$, the top of the stack is on the right. We define $|s| := m$, called the *height* of s, and $\mathsf{pop}(s) := [s_1, \ldots, s_{m-1}]$ (which is equal to $[]$ if $m = 1$). For $0 \leq i \leq k$, $\mathsf{top}^i(s)$ denotes the topmost i-stack of s.

An n-CPDA has the following operations on an (n, n)-stack s:

- pop^k, where $1 \leq k \leq n$, removes the topmost $(k-1)$-stack (undefined when $|\mathsf{top}^k(s)| = 1$);
- $\mathsf{push}^1_{a,l}$, where $1 \leq l \leq n$ and $a \in \Gamma$, pushes on the top of the topmost 1-stack the atom $(a, l, \mathsf{pop}(\mathsf{top}^l(s)))$;
- push^k, where $2 \leq k \leq n$, duplicates the topmost $(k-1)$-stack inside the topmost k-stack;
- $\mathsf{collapse}$, when $\mathsf{top}^0(s) = (a, l, t)$, replaces the topmost l-stack by t (undefined when $t = []$);
- rew_a, where $a \in \Gamma$, replaces the topmost atom (b, l, t) by (a, l, t).

Denote the set of all these operations as $\Theta^n(\Gamma)$. Operation rew_a is not always present in definitions of CPDA, but we add it following [15].

A *position* is an n-tuple $x = (p_n, \dots, p_1)$ of natural numbers. The atom at position x in an n-stack s is the p_1-th 0-stack in the p_2-th 1-stack in ... in the p_n-th $(n-1)$-stack of s. We say that x is a position of s, if such atom exists. For an n-stack s and a position x in s, we define $s_{\leq x}$ as the stack obtained from s by a sequence of pop operations, in which the topmost atom is at position x.

An (n, n)-stack s is called *constructible* if it can be obtained by a sequence of operations in $\Theta^n(\Gamma)$ from a stack with only one atom $(a, 1, [])$ for some $a \in \Gamma$. It is not difficult to see that when restricted to constructible stacks, our definition of stacks coincides with the classical one.

Proposition 2.1. *Let s be a constructible n-stack, and x a position of an atom (a, l, t) in s. Then t is a proper prefix of $\mathsf{top}^l(s_{\leq x})$, that is, $t = [t_1, \dots, t_m]$ and $\mathsf{top}^l(s_{\leq x}) = [t_1, \dots, t_{m'}]$ with $m < m'$.*

An n-CPDA $\mathcal{A}$ is a tuple $(\Sigma, \Pi, Q, q_0, \Gamma, \perp_0, \Delta, \Lambda)$, where Σ is a finite set of transition labels; Π is a finite set of configuration labels; Q is a finite set of control states containing the initial state q_0; Γ is a finite stack alphabet containing the initial stack symbol $\perp_0$; $\Delta \subseteq Q \times \Gamma \times \Sigma \times \Theta_n(\Gamma) \times Q$ is a transition relation; $\Lambda \subseteq Q \times \Gamma \times \Pi$ is a predicate relation.

A configuration of $\mathcal{A}$ is a pair (q, s) where q is a control state and s is an (n, n)-stack. Such a configuration satisfies a predicate $b \in \Pi$ just in case $(q, \mathsf{lb}(\mathsf{top}^0(s)), b) \in \Lambda$. For $c \in \Sigma$, we say that $\mathcal{A}$ can c-transition from (q, s) to $(q', \theta(s))$, written $(q, s) \xrightarrow{c} (q', \theta(s))$, if and only if $(q, \mathsf{lb}(\mathsf{top}^0(s)), c, \theta, q') \in \Delta$. For a language L over Σ we write $(q, s) \xrightarrow{L} (q', s')$ when (q', s') can be reached from (q, s) by a sequence of transitions such that the word of their labels is in L. The initial configuration of $\mathcal{A}$ is $(q_0, \perp)$, where $\perp$ is the stack containing only one atom which is $(\perp_0, 1, [])$.

We define three graphs with Π-labelled nodes and Σ-labelled directed edges. The graph $\mathcal{G}^{ano}(\mathcal{A})$ has as nodes all configurations, $\mathcal{G}^{con}(\mathcal{A})$ only configurations (q, s) in which s is constructible, and $\mathcal{G}(\mathcal{A})$ only configurations (q, s) such that $(q_0, \perp) \xrightarrow{\Sigma^*} (q, s)$. In all cases we have a c-labelled edge from (q, s) to (q', s') when $(q, s) \xrightarrow{c} (q', s')$. Assuming that $\varepsilon \in \Sigma$, we can define the ε-closure of a graph $\mathcal{G}$: it contains only those nodes of $\mathcal{G}$ which have some incoming edge not labeled by ε, and two nodes are connected by a c-labelled edge (where $c \neq \varepsilon$) when in $\mathcal{G}$ they are related by $\xrightarrow{\varepsilon^* c}$. We denote the ε-closure of $\mathcal{G}(\mathcal{A})$ as $\mathcal{G}_{/\varepsilon}(\mathcal{A})$.

We consider first-order logic (FO) on graphs as it is standardly defined, with a unary predicate for each symbol in Π and a binary relation for each symbol in Σ, together with a binary equality symbol. A formula is Σ_1, if it is of the form $\exists x_1 \dots \exists x_k . \varphi$, where φ is without quantifiers.

3 Luminosity for 3-CPDA

The goal of this section is to prove the following theorem.

Theorem 3.1. *Given a Σ_1 first-order sentence φ and a 3-CPDA $\mathcal{A}$, it is decidable whether φ holds in $\mathcal{G}_{/\varepsilon}(\mathcal{A})$.*

In [15] (Theorem 5, and the comment below) this is proven under the restriction to 3-CPDA $\mathcal{A}$ which are luminous. It remains to show that each 3-CPDA $\mathcal{A}$ can be turned into a luminous 3-CPDA $\mathcal{A}'$ for which $\mathcal{G}^\varepsilon(\mathcal{A}) = \mathcal{G}^\varepsilon(\mathcal{A}')$.

Let us recall the definition of luminosity. For an (n, n)-stack s, we write $stripln(s)$ to denote the (n, n)-stack that results from deleting all order-n links from s (that is, changing atoms (a, n, p) into $(a, n, [])$; of course we perform this stripping also inside all links). An n-CPDA $\mathcal{A}$ is *luminous* whenever for every two configurations (q, s), (q', s') in the ε-closure with $stripln(s) = stripln(s')$ it holds $s = s'$.

For example, the two 2-stacks

$$[[(a,1,[]),(b,1,[])],[(a,1,[]),(b,2,s_1)],[(a,1,[]),(b,2,s_1)]] \qquad \text{and}$$
$$[[(a,1,[]),(b,1,[])],[(a,1,[]),(b,2,s_1)],[(a,1,[]),(b,2,s_2)]]$$

with $s_1 = [[(a,1,[]),(b,1,[])]]$ and $s_2 = [[(a,1,[]),(b,1,[])],[(a,1,[]),(b,2,s_1)]]$ become identical if the links are removed. Extra annotations would need to be added to the stack to tell them apart without links.

We explain briefly why luminosity is needed in the decidability proof in [15]. The proof reduces the order of the CPDA by one (a configuration of an n-CPDA is represented as a sequence of configurations in an $(n-1)$-CPDA), in the cost of creating a more complicated formula. This reduction allows to deal with the operational aspect of links (that is, with the collapse operation). However, there is also the problem of preserving identities, to which first-order logic is sensitive. For this reason, the reduction would be incorrect, if by removing links from two different configurations, suddenly they would become equal.

Let us emphasize that we are not trying to simulate the operational behavior of links in a 3-CPDA after removing them. We only want to construct another 3-CPDA with the same $\mathcal{G}_{/\varepsilon}$, which still uses links of order-3, but such that $stripln(s) = stripln(s')$ implies $s = s'$.

Our construction is quite similar to that from [15] (which works for such n-CPDA which only have links of order n). The key idea which allows to extend it to 3-CPDA which also have links of order 2, is to properly assign the value of generation (see below) to atoms with links of order 2.

Fix a 3-CPDA $\mathcal{A}$ with a stack alphabet Γ. W.l.o.g. we assume that $\mathcal{A}$ "knows" what is the link order in each atom: Γ is divided into Γ_1, Γ_2, Γ_3, so that in all reachable configurations of $\mathcal{A}$ we only have atoms (a, k, t) for which $a \in \Gamma_k$. We also assume that $\mathcal{A}$ does not perform $\mathsf{collapse}$ on links of order 1 (as it is equivalent to pop^1 repeated twice). We will construct a luminous 3-CPDA $\mathcal{A}'$ with stack alphabet

$$\Gamma' = \Gamma \times \{1{>}, 1{=}, 1{<}\} \times \{2{>}, 2{=}, 2{<}, \neg 2\} \times \{3{\geq}, 3{<}, \neg 3\}.$$

To obtain luminosity, it would be enough to mark for each atom (in particular for atoms with links of order 3), whether it was created at its position, or copied from the 1-stack below, or copied from the 2-stack below. Of course we cannot do this for each atom independently, since when a whole stack is copied, we cannot change markers in all its atoms; thus some markers are needed also on top of 1-stacks and 2-stacks.

There is an additional difficulty that all markers should be placed as a function of a stack, not depending on how the stack was constructed (otherwise one node in $\mathcal{G}_{/\varepsilon}(\mathcal{A})$ would be transformed into several nodes in $\mathcal{G}_{/\varepsilon}(\mathcal{A}'))$. Thus when an atom is created by $\mathsf{push}^1_{a,l}$ we cannot just mark it as created here, since equally well identical atom could be copied from a stack below. However, an atom with a link pointing to the 3-stack containing all the 2-stacks below cannot be a copy from the previous 2-stack. We can also be sure about this for some atoms with links of order 2, namely those whose link target already contains an atom with such "fresh" link of order 3. For these reasons, for each k-stack s (for $0 \leq k \leq 2$), including $s = []$, we define $gn(s)$, the *generation* of s:

$$gn([]) := 0,$$
$$gn([s_1, \dots, s_m]) := \max(0, \max_{1\leq i\leq m} gn(s_i)),$$
$$gn((a,k,t)) := \begin{cases} |t| + 1 & \text{if } k = 3, \\ gn(t) & \text{if } k = 2, \\ -1 & \text{if } k = 1. \end{cases}$$

Intuitively, $gn(s)$ is a lower bound for the height of the 3-stack of the CPDA in the moment when s was modified last time (or created). For convenience, the generation of an atom with a link of order 1 is smaller than the generation of any k-stack for $k > 0$, and the generation of any atom with a link of order 3 is greater than the generation of the empty stack.

For each constructible 3-stack s over Γ we define its marked variant $mar(s)$, which is obtained by adding markers at each position x of s as follows.

- Let $i \in \{1, 2\}$ and $r \in \{>, =, <\}$, or $i = 3$ and $r \in \{\geq, <\}$. If x is the topmost position in its $(i-1)$-stack (always true for $i = 1$), we put marker ir at x if
$$gn(\mathsf{pop}(\mathsf{top}^i(s_{\leq x})))\ r\ gn(\mathsf{top}^{i-1}(s_{\leq x})).$$
- Assume that x is not topmost in its 1-stack, and the position directly above it has assigned marker $1<$. Let t be the atom just above x, and let y be the highest position in $s_{\leq x}$ such that $gn(\mathsf{top}^2(s_{\leq y})) < gn(t)$. We put marker $2r$ at x if
$$gn(\mathsf{pop}(\mathsf{top}^2(s_{\leq y})))\ r\ gn(\mathsf{top}^1(s_{\leq y}));$$
- If no marker of the form $2r$ (or $3r$) is placed at x, we put there $\neg 2$ (respectively, $\neg 3$).

For example, the marker $2<$ is placed at the top of some 1-stack to say that the generation of this 1-stack is greater than of all the 1-stacks below it, in the same 2-stack.

In the second item, notice that y always will be found, even inside the topmost 2-stack of $s_{\leq x}$ (because s is constructible, the bottommost position of the

2-stack satisfies the inequality, since it has generation -1, and $gn(t) > 0$). Intuitively, when an atom from a new generation is placed above y, in y we keep his $2r$ marker (notice that the formula is the same as in the first item). This is needed to reproduce the $2r$ marker when y becomes again the topmost position. Necessarily, the marker from y will be also present at positions x which are copies of y. Notice however that when we remove an atom at position x using pop^1, and then we reproduce identical atom using $\mathsf{push}^1_{a,k}$, the $2r$ marker has to be written there again (mar should be a function of the stack). For this reason the x containing the $2r$ marker from y is not necessarily a copy of y: we store the marker in the highest atom below an atom from the higher generation. See Figure 1 for an example.

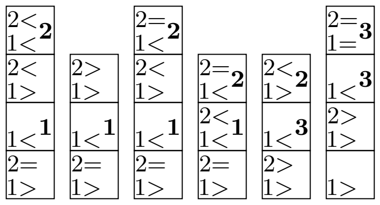

Fig. 1. An example 2-stack (one out of many in a 3-stack). It grows from left to right. We indicate all $1r$ and $2r$ markers, as well as the generation of atoms (bold; no number for generation -1). To calculate the $2<$ marker at positions $(1, 3)$, $(3, 3)$, and $(4, 2)$ we have used position $(1, 3)$ as y. Observe the atom of generation 2 above an atom of generation 3; this is possible for an atom with a link of order 2.

The key property is that the markers can be updated by a CPDA. We will say that a CPDA *defines a path* if from each configuration there is at most one transition available.

Lemma 3.2. *Let $\theta \in \Theta^n(\Gamma)$ be a stack operation. Then there exists a 3-CPDA $\mathcal{A}_\theta$ which defines a path, with stack alphabet Γ' and two distinguished states q_0, q_1, such that for each constructible 3-stack s:*

- *if $\theta(s)$ exists, then there is a unique configuration with state q_1 reachable by $\mathcal{A}_\theta$ from $(q_0, mar(s))$; the stack in this configuration is $mar(\theta(s))$;*
- *if θ cannot be applied to s, no configuration with state q_1 is reachable by $\mathcal{A}_\theta$ from $(q_0, mar(s))$.*

Additionally, $\mathcal{A}_\theta$ does not use the $\mathsf{collapse}$ *operation for $\theta \neq$* $\mathsf{collapse}$.

Proof (sketch). This is a tedious case analysis. In most cases we just have to apply a local change of markers. For a push, we update markers in the previously topmost atom (depending on markers which were previously there), then we perform the push, and then we update markers in the new topmost atom. For pop or $\mathsf{collapse}$, we perform this operation, and then we update markers in the atom which became topmost, depending on markers in this atom, and in the atom which was topmost previously.

There is one exception from this schema, during such $\mathsf{push}^1_{a,k}$ operation which increases the generation of the topmost 1-stack, but not of the topmost 2-stack. In this situation in the previously topmost atom we should place a $2r$ marker, the same as in the atom just below the bottommost atom having the highest generation in the 2-stack. This information is not available locally; to find this atom (and the marker in it), we copy the topmost 2-stack (push^3), we destructively search for this atom (which is easy using the markers), and then we remove the garbage using pop^3. $\square$

Lemma 3.3. *Let s, s' be constructible 3-stacks such that $stripln(mar(s)) = stripln(mar(s'))$. Then $s = s'$.*

Proof (sketch). We prove this by induction, so we can assume that s is equal to s' everywhere except its topmost atom. Only the situation when $\mathsf{top}^0(s)$ has a link of order 3 is nontrivial; then we have to prove that the generation of the topmost atoms of s and s' is the same. We notice that $gn(\mathsf{top}^0(s)) = gn(\mathsf{top}^1(s))$. We have several cases. When the topmost atom is marked by $2{=}$, its generation is $gn(\mathsf{pop}(\mathsf{top}^2(s)))$, which is determined by the part below $\mathsf{top}^0(s)$. When it is marked by $2{<}$ and $3{<}$, its generation is $|s|$. When it is marked by $2{<}$ and by $3{\geq}$, this atom was necessarily copied from the 2-stack below (and has the same generation as the corresponding atom there). Finally, when it is marked by $2{>}$, this atom was necessarily copied from the 1-stack below (here, or in some 2-stack below). $\square$

Having these two lemmas it is easy to conclude. We construct $\mathcal{A}'$ from $\mathcal{A}$ as follows. The initial stack of $\mathcal{A}'$ should be $mar(\perp)$. Whenever $\mathcal{A}$ wants to apply a c-labelled transition with operation θ and final state q, $\mathcal{A}'$ simulates the automaton $\mathcal{A}_\theta$ using ε-transitions, and then changes state to q using a c-labelled transition. Then $\mathcal{G}_{/\varepsilon}(\mathcal{A})$ is isomorphic to $\mathcal{G}_{/\varepsilon}(\mathcal{A}')$: a configuration (q, s) corresponds to $(q, mar(s))$. Moreover, by Lemma 3.3 the CPDA is luminous (notice that the ε-closure contains only configurations with stack of the form $mar(s)$).

4 Unreachable configurations with constructible stack

In this section we prove that the FO theory is undecidable for configuration graphs without the restriction to reachable configurations, but when we allow only constructible stacks (oppositely to the conjecture stated in [15]). On the other hand, in the next section we show decidability when one allows also stacks which are unconstructible. Let us recall that the FO theory was known [15] to be undecidable, when one allows also stacks which are unconstructible, but for the classical definition of stacks (links represented as numbers, pointing to substacks). Our proof goes along a similar line, but additional care is needed to ensure that the stacks used in the reduction are indeed constructible. For this reason we need to use stacks of order 4 (while [15] uses stacks of order 3).

To be precise, we prove our undecidability result for graph $\mathcal{G}^{con}(\mathcal{A})$, for the 4-CPDA $\mathcal{A}$ which has a single-letter stack alphabet $\{\star\}$, one state, and for each stack operation θ a θ-labelled transition performing operation θ. Since there is only one state we identify a configuration with the $(4,4)$-stack it contains.

Theorem 4.1. *FO is undecidable on $\mathcal{G}^{con}(\mathcal{A})$.*

We reduce the first-order theory of finite graphs, which is well-known to be undecidable [18]. A finite graph $G = (V, E)$ consists of a finite domain V of nodes over which there is a binary irreflexive and symmetric relation E of edges. We will use the domain of $\mathcal{G}^{con}(\mathcal{A})$ to represent all possible finite graphs.

First we observe that in first-order logic we can determine the order of the link in the topmost atom. That is, for $1 \leq k \leq 4$ we have a formula $link^k(s)$ which is true in configurations s such that $\mathsf{top}^0(s) = (\star, k, t)$ with $t \neq []$. The formulae are defined by

$$link^k(s) := \bigwedge_{1\leq i<k} \neg link^i(s) \wedge \exists t.(s \xrightarrow{\mathsf{collapse}} t \wedge eq^k(s,t)),$$

where $eq^k(s,t)$ states that s and t differ only in their topmost k-stacks, that is $eq^4(s,t) := true$, and for $1 \leq k \leq 3$,

$$eq^k(s,t) := \exists u.(s \xrightarrow{\mathsf{pop}^{k+1}} u \wedge t \xrightarrow{\mathsf{pop}^{k+1}} u) \vee (eq^{k+1}(s,t) \wedge \neg\exists u.(s \xrightarrow{\mathsf{pop}^{k+1}} u)).$$

Next, we define two sets of substacks of a 4-stack s which can be easily accessed in FO. The set $vis^4(s)$ contains s and the stacks t for which in $\mathsf{top}^3(s)$ there is the atom $(\star, 4, t)$. The set $vis^3(s)$ contains s and the stacks t for which $\mathsf{pop}(s) = \mathsf{pop}(t)$ and in $\mathsf{top}^2(s)$ there is the atom $(\star, 3, \mathsf{top}^3(t))$. When s is constructible, the property that $t \in vis^k(s)$ (for $k \in \{3,4\}$) can be expressed by the FO formula

$$vis^k(s,t) := \exists u.(u \xrightarrow{\mathsf{pop}^k} s \wedge link^k(u) \wedge u \xrightarrow{collapse} t).$$

To every constructible 4-stack s we assign a finite graph $G(s)$ as follows. Its nodes are $V := vis^4(s)$. Two nodes $t, u \in V$ are connected by an edge when $\mathsf{top}^0(v) = (\star, 4, u)$ for some $v \in vis^3(t)$, or $\mathsf{top}^0(v) = (\star, 4, t)$ for some $v \in vis^3(u)$.

Lemma 4.2. *For each non-empty finite graph G there exists a constructible $(4,4)$-stack s_G (in the domain of $\mathcal{G}^{con}(\mathcal{A})$) such that G is isomorphic to $G(s_G)$.*

Proof. Suppose that $G = (V, E)$ where $V = \{1, 2, \ldots, k\}$. The proof is by induction on k. If $k = 1$, as s_G we just take the (constructible) 4-stack consisting of one atom $(\star, 1, [])$. Assume that $k \geq 2$. For $1 \leq i < k$, let G_i be the subgraph of G induced by the subset of nodes $\{1, 2, \ldots, i\}$, and let $s_i := s_{G_i}$ be the stack corresponding to G_i obtained by the induction assumption. We will have $\mathsf{pop}^4(s_G) = s_{k-1}$, and $\mathsf{top}^4(s_G) = t_k$, where 3-stacks t_i for $0 \leq i \leq k$ are defined by induction as follows. We take $t_0 = []$. For $i > 0$ we take take $\mathsf{pop}(t_i) = t_{i-1}$, and the topmost 2-stack of t_i consists of one or two 1-stacks. Its first 1-stack is

$$[(\star, 1, []), (\star, 4, s_1), (\star, 4, s_2), \dots, (\star, 4, s_{k-1}), (\star, 3, t_0), (\star, 3, t_1), \dots, (\star, 3, t_{i-1})].$$

If $(i, k) \notin E$ we only have this 1-stack; if $(i, k) \in E$, in $\mathsf{top}^2(t_i)$ we also have the 1-stack

$$[(\star, 1, []), (\star, 4, s_1), (\star, 4, s_2), \dots, (\star, 4, s_i)].$$

We notice that all substacks are available in vis^i: $vis^4(s_G)$ contains stacks $s_1, s_2, \dots, s_{k-1}, s_G$, and $vis^3(s_G)$ contains all stacks obtained from s_G by replacing its topmost 3-stack by t_i for some $i \geq 1$. It follows that $G(s_G)$ is isomorphic to G.

It is also easy to see that s_G is constructible. We create it out of s_{k-1} by performing push^4 and appropriately changing the topmost 3-stack. Notice that the bottommost 1-stack of $\mathsf{top}^3(s_{k-1})$ starts with $(\star, 1, []), (\star, 4, s_1), (\star, 4, s_2), \dots,$ $(\star, 4, s_{k-2})$. We uncover this prefix using a sequence of pop operations. We append $(\star, 4, s_{k-1})$ and $(\star, 3, t_0)$ by $\mathsf{push}^1_{\star,4}$ and $\mathsf{push}^1_{\star,3}$. If $(1, k) \in E$, we create the second 1-stack using push^2 and a sequence of pop^1. This already gives the first 2-stack. To append each next (i-th) 2-stack, we perform push^3; we remove the second 1-stack if it exists using pop^2; we append $(\star, 3, t_{i-1})$ using $\mathsf{push}^1_{\star,3}$; if necessary we create the second 1-stack using push^2 and a sequence of pop^1. □

We have a formula stating that two nodes x, y of $G(s)$ are connected by an edge:

$$\begin{aligned} E(x, y) :=& \exists z.(vis^3(x, z) \wedge link^4(z) \wedge z \xrightarrow{\mathsf{collapse}} y) \vee \\ &\vee \exists z.(vis^3(y, z) \wedge link^4(z) \wedge z \xrightarrow{\mathsf{collapse}} x). \end{aligned}$$

Given any sentence φ over finite graphs, we construct a formula $\varphi'(s)$ by replacing all occurrences of the atomic binary predicate xEy with the formula $E(x, y)$ from above, and relativising all quantifiers binding a variable x to $vis^4(s, x)$. Then for each constructible $(4, 4)$-stack s, φ holds in $G(s)$ if and only if $\varphi'(s)$ holds in $\mathcal{G}^{con}(\mathcal{A})$. Thus φ holds in some finite graph if and only if it holds in the empty graph or $\exists s.\varphi'(s)$ holds in $\mathcal{G}^{con}(\mathcal{A})$. This completes the reduction and hence the proof of Theorem 4.1, since it is trivial to check whether φ holds in the empty graph.

5 Unreachable configurations with annotated stack

In this section we prove decidability of first order logic in the graph of all configurations, not restricted to constructible stacks.

Theorem 5.1. *Given a first-order sentence φ and a CPDA $\mathcal{A}$, it is decidable whether φ holds in $\mathcal{G}^{ano}(\mathcal{A})$.*

For the rest of the section fix a CPDA $\mathcal{A}$ of order n, with stack alphabet Γ. The key idea of the proof is that an FO formula can inspect only a small topmost part of the stack, and check equality of the parts below. Thus instead of valuating variables into stacks, it is enough to describe how the top of the stack looks like, and which stacks below are equal. To formalize this we define generalized stacks.

Consider the following operations on stacks:

– for each $k \in \{1, \dots, n\}$ operation $\mathsf{first}^k(\cdot)$ which takes a $(k-1)$-stack s and returns the k-stack $[s]$,
– for each $k \in \{1, \dots, n\}$ operation $\mathsf{app}^k(\cdot, \cdot)$ which takes a k-stack $[s_1, \dots, s_m]$ and a $(k-1)$-stack s, and returns the k-stack $[s_1, \dots, s_m, s]$,
– for each $a \in \Gamma$ and $k \in \{1, \dots, n\}$ operation $\mathsf{cons}(a, k, [])$ (without arguments) which returns the 0-stack $(a, k, [])$,
– for each $a \in \Gamma$ and $k \in \{1, \dots, n\}$ operation $\mathsf{cons}(a, k, \cdot)$ which takes a k-stack s and returns the 0-stack (a, k, s).

We notice that stacks can be seen as elements of the free multisorted algebra with these operations and no generators (we have $n+1$ sorts, one for each order of stacks). In the proof we need elements of the free multisorted algebra with these operations and some generators: for each sort k we have an infinite set of constants, denoted $x_1^k, x_2^k, \dots$. Elements of this algebra will be called *generalized stacks*. Thus a generalized stack is a stack in which we have replaced some prefixes of some stacks by constants. Generalized stacks will be denoted by uppercase letters.

For each generalized stack S and each $d \in \mathbb{N}$ we define the set $\mathsf{ts}_{=d}(S)$ of stacks. These are substacks of S which are at "distance" exactly d from the top. The definition is inductive: we take $\mathsf{ts}_{=0}(S) := \{S\}$,

$$\mathsf{ts}_{=1}(S) := \begin{cases} \{T\} & \text{if } S = \mathsf{first}^k(T), \\ \{T, U\} & \text{if } S = \mathsf{app}^k(T, U), \\ \emptyset & \text{if } S = \mathsf{cons}(a, k, []), \\ \{T\} & \text{if } S = \mathsf{cons}(a, k, T), \\ \emptyset & \text{if } S \text{ is a constant,} \end{cases}$$

and $\mathsf{ts}_{=d+1}(S) := \bigcup_{T \in \mathsf{ts}_{=d}(S)} \mathsf{ts}_{=1}(T)$ for $d \geq 1$. Moreover we define $\mathsf{ts}_{\leq d}(S) := \bigcup_{e \leq d} \mathsf{ts}_{=e}(S)$ and for $d \in \mathbb{N} \cup \{\infty\}$, $\mathsf{ts}_{<d}(S) := \bigcup_{e<d} \mathsf{ts}_{=e}(S)$.

A *valuation* is a (partial) function v mapping constants to stacks, preserving the order. Such v can be extended to a homomorphism $\overline{v}$ mapping generalized stacks to stacks. Obviously, to compute $\overline{v}(S)$ it is enough to define v only on constants appearing in S.

In the logic we can talk also about equality of stacks, so we are interested in valuations which applied to different generalized stacks give different stacks. This is described by the $\hookrightarrow_d$ relation. Let $S_1, \dots, S_m$ (for $m \geq 0$) be extended stacks, and $s_1, \dots, s_m$ stacks, and $d \in \mathbb{N}$. Then we say that $(S_1, \dots, S_m) \hookrightarrow_d (s_1, \dots, s_m)$ if there exists a valuation v such that

– $s_i = \overline{v}(S_i)$ for each i, and

– no element of $\bigcup_i \mathsf{ts}_{<d}(S_i)$ is a constant (that is, all constants are at depth at least d), and
– for each $T, U \in \bigcup_i \mathsf{ts}_{\leq d}(S_i)$ such that $\overline{v}(T) = \overline{v}(U)$, it holds $T = U$.

Example 5.2. Consider the following 2-stack:

$$s := [[(a,1,[]),(b,1,[]),(c,1,[])],[(a,1,[]),(b,1,[]),(c,1,[])]].$$

It can be written as:

$$\mathsf{app}^2\Big(\mathsf{first}^2\Big(\mathsf{app}^1(\mathsf{app}^1(\mathsf{first}^1(\mathsf{cons}(a,1,[])),\mathsf{cons}(b,1,[])),\mathsf{cons}(c,1,[]))\Big),$$
$$\mathsf{app}^1(\mathsf{app}^1(\mathsf{first}^1(\mathsf{cons}(a,1,[])),\mathsf{cons}(b,1,[])),\mathsf{cons}(c,1,[]))\Big).$$

It holds

$$(\mathsf{app}^2(\mathsf{first}^2(\mathsf{app}^1(x^1,\mathsf{cons}(c,1,[]))),\mathsf{app}^1(x^1,\mathsf{cons}(c,1,[])))) \hookrightarrow_2 (s),$$

where the valuation maps x^1 into $\mathsf{app}^1(\mathsf{first}^1(\mathsf{cons}(a,1,[])),\mathsf{cons}(b,1,[]))$. On the other hand it does not hold $(\mathsf{app}^2(\mathsf{first}^2(y^1),\mathsf{app}^1(x^1,\mathsf{cons}(c,1,[])))) \hookrightarrow_2 (s)$; the problem is that the two 1-stacks of s were equal, while they are different in this generalized 2-stack. This shows that we cannot place all constants at one, fixed depth.

When a formula (having already some generalized stacks assigned to its free variables) starts with a quantifier, as a value of the quantified variable we want to try all possible generalized stacks which are of the special form, as described by the following definition. Let $S_1,\dots,S_m,S_{m+1}$ (for $m \geq 0$) be generalized stacks, let $d \in \mathbb{N}$, and $d' := d + 2^{d+1}$. We say that S_{m+1} is *d-normalized* with respect to $(S_1,\dots,S_m)$ if

– no element of $\mathsf{ts}_{<d}(S_{m+1})$ is a constant, and
– each element of $\mathsf{ts}_{=d}(S_{m+1})$ is
  • a "fresh" constant, i.e. not belonging to $\bigcup_{i\leq m} \mathsf{ts}_{<\infty}(S_i)$, or
  • an element of $\bigcup_{i\leq m} \mathsf{ts}_{\leq d'}(S_i)$, or
  • an element of $\mathsf{ts}_{<d}(S_{m+1})$.

The key point is that for fixed $S_1,\dots,S_m$ there are only finitely many d-normalized generalized stacks S_{m+1} (up to renaming of fresh constants), so we can try all of them. The next two lemmas say that to consider d-normalized generalized stacks is exactly what we need.

Lemma 5.3. *Let $S_1,\dots,S_m$ (for $m \geq 0$) be generalized stacks, let $s_1,\dots,s_m$ and s_{m+1} be stacks, let $d \in \mathbb{N}$ and $d'' := d+2^{d+2}$. Assume that $(S_1,\dots,S_m) \hookrightarrow_{d''} (s_1,\dots,s_m)$. Then there exists a generalized stack S_{m+1} d-normalized with respect to $(S_1,\dots,S_m)$ and such that $(S_1,\dots,S_m,S_{m+1}) \hookrightarrow_d (s_1,\dots,s_m,s_{m+1})$.*

Proof (sketch). Let $d' := d + 2^{d+1}$ (this is the d' used in the definition in d-normalization, and is smaller than d''). Let v be a valuation witnessing that $(S_1,\dots,S_m) \hookrightarrow_{d''} (s_1,\dots,s_m)$, i.e. such that $s_i = \overline{v}(S_i)$ for each $i \leq m$. For each stack $s \in \mathsf{ts}_{\leq d}(s_{m+1})$ we define by induction a generalized stack $\mathsf{repl}(s)$:

– if $s \in \mathsf{ts}_{<d}(s_{m+1})$, we take

$$\mathsf{repl}(s) := \begin{cases} \mathsf{first}^k(\mathsf{repl}(t)) & \text{if } s = \mathsf{first}^k(t), \\ \mathsf{app}^k(\mathsf{repl}(t), \mathsf{repl}(u)) & \text{if } s = \mathsf{app}^k(t, u), \\ \mathsf{cons}(a, k, []) & \text{if } s = \mathsf{cons}(a, k, []), \\ \mathsf{cons}(a, k, \mathsf{repl}(t)) & \text{if } s = \mathsf{cons}(a, k, t); \end{cases}$$

– otherwise, if $s = \overline{v}(S)$ for some $S \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'}(S_i)$, we take $\mathsf{repl}(s) := S$;
– otherwise, we take $\mathsf{repl}(s) := x_s$, where x_s is a fresh constant.

At the end we take $S_{m+1} := \mathsf{repl}(s_{m+1})$. It remains to check in detail that such S_{m+1} satisfies all parts of the definitions. Notice that there can exist stacks s which are simultaneously in $\mathsf{ts}_{<d}(s_{m+1})$ and $\mathsf{ts}_{=d}(s_{m+1})$ (so it is not true that we apply one of the last two cases to each stack at depth d). □

Lemma 5.4. *Let $S_1, \dots, S_m$ and S_{m+1} (for $m \geq 0$) be generalized stacks, let $s_1, \dots, s_m$ be stacks, let $d \in \mathbb{N}$ and $d'' := d+2^{d+2}$. Assume that $(S_1, \dots, S_m) \hookrightarrow_{d''} (s_1, \dots, s_m)$, and that S_{m+1} is d-normalized with respect to $(S_1, \dots, S_m)$. Then there exists a stack s_{m+1} such that $(S_1, \dots, S_m, S_{m+1}) \hookrightarrow_d (s_1, \dots, s_m, s_{m+1})$.*

Proof (sketch). It is enough to map the constants appearing in S_{m+1} but not in S_i for $i \leq m$ into "fresh" stacks, such that none of them is a substack of any other nor of any s_i for $i \leq m$ (the latter is easy to obtain by taking these stacks to be bigger than all s_i). □

We also easily see that the atomic FO formulas can evaluated on the level of generalized stacks related by the $\hookrightarrow_{n+1}$ to the actual stacks.

Lemma 5.5. *Let S, T be generalized n-stacks, and let s, t be n-stacks. Assume that $(S, T) \hookrightarrow_{n+1} (s, t)$. Then taking as input S and T (even not knowing s and t) one can compute:*

– $\mathsf{lb}(\mathsf{top}^0(s))$*, and*
– *whether $s = t$, and*
– *for any stack operation $\theta \in \Theta^n(\Gamma)$, whether it holds $\theta(s) = t$.*

Using the last three lemmas we can check whether an FO sentence holds in $\mathcal{G}^{ano}(\mathcal{A})$. Indeed, for each quantifier we check all possible generalized stacks which are d-normalized with respect to the previously fixed variables, for big enough d (depending on the quantifier rank of the formula, so that the induction works fine), and with atomic formulas we deal using Lemma 5.5.

References

1. Maslov, A.N.: The hierarchy of indexed languages of an arbitrary level. Soviet Math. Dokl. **15** (1974) 1170–1174
2. Maslov, A.N.: Multilevel stack automata. Problems of Information Transmission **12** (1976) 38–43

3. Knapik, T., Niwinski, D., Urzyczyn, P.: Higher-order pushdown trees are easy. In Nielsen, M., Engberg, U., eds.: FoSSaCS. Volume 2303 of Lecture Notes in Computer Science., Springer (2002) 205–222
4. Parys, P.: Collapse operation increases expressive power of deterministic higher order pushdown automata. In Schwentick, T., Dürr, C., eds.: STACS. Volume 9 of LIPIcs., Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011) 603–614
5. Parys, P.: On the significance of the collapse operation. In: LICS, IEEE (2012) 521–530
6. Hague, M., Murawski, A.S., Ong, C.H.L., Serre, O.: Collapsible pushdown automata and recursion schemes. In: LICS, IEEE Computer Society (2008) 452–461
7. Knapik, T., Niwinski, D., Urzyczyn, P., Walukiewicz, I.: Unsafe grammars and panic automata. In Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M., eds.: ICALP. Volume 3580 of Lecture Notes in Computer Science., Springer (2005) 1450–1461
8. Caucal, D.: On infinite terms having a decidable monadic theory. In Diks, K., Rytter, W., eds.: MFCS. Volume 2420 of Lecture Notes in Computer Science., Springer (2002) 165–176
9. Cachat, T.: Higher order pushdown automata, the caucal hierarchy of graphs and parity games. In Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J., eds.: ICALP. Volume 2719 of Lecture Notes in Computer Science., Springer (2003) 556–569
10. Carayol, A., Wöhrle, S.: The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In Pandya, P.K., Radhakrishnan, J., eds.: FSTTCS. Volume 2914 of Lecture Notes in Computer Science., Springer (2003) 112–123
11. Kartzow, A.: Collapsible pushdown graphs of level 2 are tree-automatic. Logical Methods in Computer Science **9**(1) (2013)
12. Broadbent, C.H.: On collapsible pushdown automata, their graphs and the power of links. PhD thesis, University of Oxford (2011)
13. Broadbent, C.H.: Prefix rewriting for nested-words and collapsible pushdown automata. [19] 153–164
14. Broadbent, C.H.: The limits of decidability for first order logic on CPDA graphs. In Dürr, C., Wilke, T., eds.: STACS. Volume 14 of LIPIcs., Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012) 589–600
15. Broadbent, C.H.: On first-order logic and CPDA graphs. Accepted to Theory of Computing Systems
16. Broadbent, C.H., Carayol, A., Hague, M., Serre, O.: A saturation method for collapsible pushdown systems. [19] 165–176
17. Kartzow, A., Parys, P.: Strictness of the collapsible pushdown hierarchy. In Rovan, B., Sassone, V., Widmayer, P., eds.: MFCS. Volume 7464 of Lecture Notes in Computer Science., Springer (2012) 566–577
18. Trachtenbrot, B.: Impossibility of an algorithm for the decision problem in finite classes. Doklady Akad. Nauk. **70** (1950) 569–572
19. Czumaj, A., Mehlhorn, K., Pitts, A.M., Wattenhofer, R., eds.: Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II. In Czumaj, A., Mehlhorn, K., Pitts, A.M., Wattenhofer, R., eds.: ICALP (2). Volume 7392 of Lecture Notes in Computer Science., Springer (2012)

A Missing proofs for Section 3

We begin by a proposition which will be useful also in Appendix B

Proposition A.1. *Let $2 \leq k \leq n$, let s be a constructible (n,n)-stack, and x a position in s. If in $\mathsf{top}^{k-1}(s_{\leq x})$ we have an atom (a,k,t), then either $t = \mathsf{pop}(\mathsf{top}^k(s_{\leq x}))$, or an atom (b,k,t) is also present in $\mathsf{top}^{k-1}(\mathsf{pop}^k(s_{\leq x}))$.*

Proof. It is enough to prove this lemma in the case when the atom (a,k,t) is at position x; then the general statement follows immediately by taking as x the position of this atom. The proof is by induction on the smallest length of a sequence of operations used to construct s. Consider a stack u which can be constructed using less operations, and such that $s = \theta(u)$ for some operation θ. If x is present also in u, nothing below x is changed by the operation, so the assumptions are satisfied also for u and x, so we can just use the induction assumption (notice that the label of the atom at position x can be changed by a rew operation). Otherwise, x was created in s by some push operation. If it was $\mathsf{push}^1_{a,k}$, we have $t = \mathsf{pop}(\mathsf{top}^k(s_{\leq x}))$. Otherwise it was push^i for some i. Then x is in the topmost $(i-1)$-stack of s, and the atom at this position was copied from the corresponding position y in the topmost $(i-1)$-stack of u. If $i \neq k$, we conclude using the induction assumption for u and y (notice that $\mathsf{pop}(\mathsf{top}^k(s_{\leq x}))$ and $\mathsf{pop}(\mathsf{top}^k(u_{\leq y}))$ are identical). If $i = k$, we see that y (which contains the atom (a,k,t)) is inside $\mathsf{top}^{k-1}(\mathsf{pop}^k(s_{\leq x}))$. □

An immediate consequence of this proposition (used for $k = n = 3$) is that in the topmost 2-stack of a constructible $(3,3)$-stack s there are no atoms with generation strictly between $gn(\mathsf{pop}(s))$ and $|s|$.

Proof (Lemma 3.2). If $\theta = \mathsf{rew}_a$, no markers need to be changed. Otherwise, we analyze separately each of the kinds of markers.

First observe that marker $1r$ at position x depends only on $\mathsf{top}^1(t_{\leq x})$. Thus, at positions which are present in both s and $\theta(s)$, it is not affected by any of the stack operations; moreover the markers are still set correctly inside a stack copied by push^2 or push^3. We only have to set it correctly in a new atom created by a $\mathsf{push}^1_{a,k}$ operation. We have several cases. For $k = 1$ the pushed atom has generation -1, which is smaller than $gn(\mathsf{top}^1(s))$, so we should put marker 1> in the new atom. For $k = 2$ the pushed atom has generation $gn(\mathsf{pop}(\mathsf{top}^2(s)))$. Its relation with $gn(\mathsf{top}^1(s))$ is described by the $2r$ marker in $\mathsf{top}^0(mar(s))$. The relation should be reversed: for 2< we should put 1>, for 2= we should put 1=, and for 2> we should put 1<. For $k = 3$ the pushed atom has generation $|s|$. If $\mathsf{top}^0(mar(s))$ contains 3≥, we have $|s| > |\mathsf{pop}(s)| \geq gn(\mathsf{pop}(s)) \geq gn(\mathsf{top}^2(s)) \geq gn(\mathsf{top}^1(s))$ (notice that the generation of an atom in a 3-stack cannot be greater than the height of this stack), so we should put 1<. If $\mathsf{top}^0(mar(s))$ contains 3<, the atom with the greatest generation in $\mathsf{top}^2(s)$ could not be copied from the 2-stack below, so $gn(\mathsf{top}^2(s)) = |s|$. If additionally $\mathsf{top}^0(mar(s))$ contains 2>, we have $gn(\mathsf{top}^1(s)) < |s|$, so we should put 1<. Otherwise (marker 3< and one of 2=, 2<) we have $gn(\mathsf{top}^1(s)) = |s|$ and we should put 1=.

Next, concentrate on the $2r$ marker. First recall that, at a position x, the marker depends only on $\mathsf{top}^2(s_{\leq x})$, and on the atom directly above x in its 1-stack (if such exists). Thus markers at positions present simultaneously in s and $\theta(s)$ need not to be changed after operations push^3, push^2, pop^3, pop^2, and $\mathsf{collapse}$ (both of order 2 or 3); after push^3 the markers are still set correctly in the copied stack.

Consider the push^2 operation. In the new topmost position we should put $2>$ if $\mathsf{top}^0(mar(s))$ contains $2>$; otherwise $2=$. This is correct, since this marker is intended to compare $gn(\mathsf{top}^2(s)) = \max(gn(\mathsf{pop}(\mathsf{top}^2(s))), gn(\mathsf{top}^1(s)))$ with $gn(\mathsf{top}^1(s))$, while the marker in $\mathsf{top}^0(mar(s))$ compares $gn(\mathsf{pop}(\mathsf{top}^2(s)))$ with $gn(\mathsf{top}^1(s))$. Next, notice that the location of the $\neg 2$ markers in the copied 1-stack will be correct, since the $1<$ markers need not to be changed in the copy. Next, take a position x in the newly created 1-stack, such that the atom directly above it exists and has marker $1<$; let x_0 be the corresponding position in the 1-stack below. Then y (as in the definition of the markers) found for x cannot be above x_0 (the atom just above x_0 has the same generation as t, the atom just above x), so it will be the same as for x_0. Thus the copied marker remains correct.

After pop^1 we only have to compute the marker in the topmost atom of $\mathsf{pop}^1(s)$. If in $\mathsf{top}^0(mar(s))$ we have marker $1>$ or $1=$, we rewrite the $2r$ marker from $\mathsf{top}^0(mar(s))$; in this case the generation of the topmost 1-stack remains unchanged, since below $\mathsf{top}^0(s)$ we have an atom with the same or higher generation. Similarly, if in $\mathsf{top}^0(mar(s))$ we have marker $2>$, we rewrite it to the atom which became topmost; removing an atom can only decrease the generation of the topmost 1-stack, so it remains smaller than $gn(\mathsf{pop}(\mathsf{top}^2(s)))$. If in $\mathsf{top}^0(mar(s))$ we have markers $2=$ and $1<$, we set the marker to $2>$; the generation of the topmost 1-stack decreases, so it becomes smaller than $gn(\mathsf{pop}(\mathsf{top}^2(s)))$. The remaining case is when in $\mathsf{top}^0(mar(s))$ we have markers $2<$ and $1<$. Then in $\mathsf{top}^0(\mathsf{pop}^1(mar(s)))$ we already had a $2r$ marker; we leave it there. Notice that, due to the $2<$ and $1<$ markers, x will be found as y (in the definition of the $2r$ marker at this position x), so this marker remains correct.

In the remaining case of $\mathsf{push}^1_{a,k}$ we have to compute the $2r$ marker in the atom which was topmost till now, and in the newly created topmost atom. Concentrate first on the newly created topmost atom. For $k = 1$ we just rewrite the marker from $\mathsf{top}^0(mar(s))$: the new atom has generation -1, so the generation of the topmost 1-stack remains unchanged. Assume that $k = 2$. Then the generation of the new atom is equal to $gn(\mathsf{pop}(\mathsf{top}^2(s)))$. Thus if the marker in $\mathsf{top}^0(mar(s))$ is $2<$ or $2=$, we rewrite it (the generation of the topmost 1-stack remains unchanged); for $2>$ the marker becomes $2=$. Finally, assume that $k = 3$. The generation of the new atom is $|s|$, and $gn(s) \leq |s|$. If $\mathsf{top}^0(mar(s))$ contains $2<$, we have $gn(\mathsf{pop}(\mathsf{top}^2(s))) < |s|$, so we should use $2<$. Also when $\mathsf{top}^0(mar(s))$ contains $3\geq$, we have $gn(\mathsf{pop}(\mathsf{top}^2(s))) \leq gn(\mathsf{top}^2(s)) \leq gn(\mathsf{pop}(s)) < |s|$, so we should use $2<$. Otherwise, $\mathsf{top}^0(mar(s))$ contains $3<$ and one of $2=$, $2>$; then $gn(\mathsf{pop}(\mathsf{top}^2(s))) = |s|$ (because an atom copied from the lower 2-stack cannot give the $3<$ relation), so we should use $2=$.

Now we describe how to find the $2r$ marker in the atom below the topmost one, after the $\mathsf{push}^1_{a,k}$ operation. Let x be the position of this atom. As described above, we can determine whether the new topmost atom will have the $1<$ marker or not, so whether we should place some $2r$ marker at x, or just $\neg 2$. Assume that we should place some $2r$. Then t (from the definition of the markers) will be equal to $\mathsf{top}^0(\theta(s))$. If $k = 3$ and marker $3\geq$ is present in $\mathsf{top}^0(mar(s))$, then t has generation $|s|$ while all atoms in s have smaller generation; thus y will be equal to x, and we can just leave the $2r$ marker which was present in $\mathsf{top}^0(mar(s))$. If $k = 1$, we cannot place marker $1<$ in the new topmost atom. If $k = 2$, we will place marker $1<$ in the new topmost atom only if $\mathsf{top}^0(mar(s))$ contained marker $2>$, so we have $gn(t) = gn(\mathsf{pop}(\mathsf{top}^2(s))) = gn(\mathsf{top}^2(s))$. Similarly, if $k = 3$ and marker $3<$ is present in $\mathsf{top}^0(mar(s))$, then we have $gn(t) = |s| = gn(\mathsf{top}^2(s))$. In both these cases, the $2r$ marker at x should be the same as in the atom (which has position y) just below the bottommost atom having generation $gn(\mathsf{top}^2(s))$.

To find this atom (and the marker in it), we copy the topmost 2-stack (push^3), we destructively search for this atom, and then we remove the garbage using pop^3. The searching works as follows. As long as the topmost atom is marked by $2>$ or $2=$, the bottommost atom with the highest generation is not in the topmost 1-stack, so we apply pop^2. If it is marked by $2<$, we have to find the bottommost atom with the highest generation in the topmost 1-stack. Thus we just perform pop^1, until we have an atom marked by $1<$. Notice that there is no such problem that some pop could not be executed due to empty stack: we start this procedure only when some atom has a positive generation.

Finally, we describe how to update the $3r$ marker. After pop^3, or a $\mathsf{collapse}$ of order 3, these markers remain correct. After push^3, the generation of the new 2-stack is the same as of the 2-stack below, so in the new topmost atom we should put $3\geq$. Operation push^2 does not change the generation of the topmost 2-stack, so we should just move the $3r$ marker from the previously topmost atom (where we put $\neg 3$) to the new topmost atom. Similarly, operation $\mathsf{push}^1_{a,k}$ for $k = 1$ or $k = 2$. On the other hand, operation $\mathsf{push}^1_{a,3}$ causes that the generation of the topmost 2-stack becomes $|s|$, while $gn(\mathsf{pop}(s)) < |s|$, so we should use the $3<$ marker in the new atom.

For operations pop^2, pop^1, and $\mathsf{collapse}$ of order 2, we have to determine whether the generation of the topmost 2-stack decreases. If it does not decrease, we should just move the $3r$ marker from the previously topmost atom to the new topmost atom. If it decreases, necessarily it becomes smaller or equal to $gn(\mathsf{pop}(s))$ (in the topmost 2-stack we cannot have atoms of generation strictly between $gn(\mathsf{pop}(s))$ and $|s|$); in this case we should put the $3\geq$ marker. When a pop^2 is performed, the generation of the topmost 2-stack decreases if and only if the topmost atom contained marker $2<$. When a pop^1 is performed, the generation of the topmost 2-stack decreases if and only if the topmost atom contained markers $2<$ and $1<$. When a $\mathsf{collapse}$ of order 2 is performed, the generation of the topmost 2-stack decreases if and only if the topmost atom contained a marker $2>$ or $1>$ (notice that the generation of the topmost 2-stack of $\theta(s)$ is the same as of the topmost atom of s). $\square$

Next we repeat the proof of Lemma 3.3 in more details. We start by an auxiliary proposition.

Proposition A.2. *Let s be a constructible $(3,3)$-stack, and x a position in it such that the atom at x has a link of order 3. Assume that $gn(\mathsf{pop}(\mathsf{top}^2(s_{\leq x}))) > gn(\mathsf{top}^0(s_{\leq x}))$. Then x is not in the bottommost 1-stack of its 2-stack, and the atom at the corresponding position in the 1-stack below exists and has the same generation as the atom at position x.*

Proof. The proof is by induction on the smallest length of a sequence of operations used to construct s. Consider a stack t which can be constructed using less operations, and such that $s = \theta(t)$ for some operation θ. If x is present also in t, nothing below x is changed by the operation, so the assumptions are satisfied also for t and x, so we can just use the induction assumption. Otherwise, x was created in s by some push operation. It cannot be $\mathsf{push}^1_{a,3}$, since the created atom would have the greatest generation in the whole s, which contradicts our assumption. If x created by push^2, we are done: atom at x is a copy of the corresponding atom in the 1-stack below. The remaining case is that x was created by push^3. Then the 2-stack containing x and the topmost 2-stack of t are identical; let y be the position corresponding to x in the topmost 2-stack of t. We conclude using the induction assumption for t and y. □

Lemma A.3. *Let s, s' be constructible 3-stacks such that $stripln(mar(s)) = stripln(mar(s'))$, and the topmost atom of s (and s') has a link of order 3, and $\mathsf{pop}^1(s) = \mathsf{pop}^1(s')$. Then $s = s'$.*

Notice that $\mathsf{pop}^1(s)$ and $\mathsf{pop}^1(s')$ exist, since in constructible 3-stack the bottommost atom of each 1-stack has a link of order 1.

Proof. It is enough to prove that $gn(\mathsf{top}^0(s)) = gn(\mathsf{top}^0(s'))$ (then the links in the topmost atoms point to fragments of the stacks of the same size, which are equal by the assumption $\mathsf{pop}^1(s) = \mathsf{pop}^1(s')$).

First notice that $gn(\mathsf{top}^0(s)) = gn(\mathsf{top}^1(s))$ (e.g. by inspecting the proof of Lemma 3.2 we see that an atom with a link of order 3 always has marker 1< or 1=, never 1>), and $gn(\mathsf{top}^0(s')) = gn(\mathsf{top}^1(s'))$. We have several cases.

Assume first that the topmost atom (of s and of s') is marked by 2=. Then $gn(\mathsf{top}^0(s)) = gn(\mathsf{pop}(\mathsf{top}^2(s)))$, and we know that $\mathsf{pop}(\mathsf{top}^2(s)) = \mathsf{pop}(\mathsf{top}^2(s'))$.

Next, assume that the topmost atom is marked by 3< and by 2<. Then necessarily its generation is $|s|$, both for s and for s' (in the topmost 2-stack we cannot have atoms of generation strictly between $gn(\mathsf{pop}(s))$ and $|s|$).

The next case is when the topmost atom is marked by 3≥ and by 2<. Let us consider the sequence of operations used to construct s; let t be the stack at the moment when the atom which is now topmost was placed at its position x. Notice that $t_{\leq x} = s$. The last operation before t, which created the atom, could not be $\mathsf{push}^1_{a,3}$, since it would create an atom of generation $|s|$, while our atom has smaller generation (due to the 3≥ marker). It could not be done by a push^2 operation as well, since the previous 1-stack has only atoms of smaller generation (due to the 2< marker). It was thus done by a push^3 operation, so the generation of our atom is the same as of the corresponding atom in the 2-stack below. And this atom is identical for s and for s'.

The remaining case is that the topmost atom is marked by 2>. Then the atom at the topmost position of s has smaller generation than $\mathsf{pop}(\mathsf{top}^2(s))$, so using Proposition A.2 we obtain that the atom $\mathsf{top}^0(s)$ has the same generation as the corresponding atom in the 1-stack below. The same is true for s', so we are done. □

Proof (Lemma 3.3). The proof is by induction on the number of atoms in s. If s consists of only one atom (whose link by definition has order 1), we are done. Otherwise, consider the operation θ which removes the topmost atom from s (typically this is pop^1, but we should use pop^2 or pop^3 to remove the only atom from a 1-stack or a 2-stack). First, we will prove that $\theta(s) = \theta(s')$. To see this, consider the automaton $\mathcal{A}_\theta$ from Lemma 3.2, and its state q_0. We start the automaton simultaneously from $(q_0, mar(s))$ and $(q_0, mar(s'))$. Observe that at each moment the corresponding configurations (q, u) and (q', u') of these two runs satisfy $q = q'$ and $stripln(u) = stripln(u')$. Indeed, this is true at the beginning. At each moment the state and the topmost stack symbol are equal, so the same transition will be used. This gives the equality $q = q'$ in the next step; also the same operation will be applied to the two stacks, which implies equality $stripln(u) = stripln(u')$ of obtained stacks (here it is important that $\mathcal{A}_\theta$ does not perform $\mathsf{collapse}$; this is ensured by Lemma 3.2). It follows that $stripln(mar(\theta(s))) = stripln(mar(\theta(s')))$. Since $\theta(s)$ and $\theta(s')$ are constructible 3-stacks, and $\theta(s)$ has less atoms than s, by induction assumption we have $\theta(s) = \theta(s')$.

Let (a, k, t) and (a, k, t') be the topmost atom of s and s', respectively. For $k = 3$ we conclude that $s = s'$ using Lemma A.3. Otherwise the equality $stripln(mar(s)) = stripln(mar(s'))$ implies that $|t| = |t'|$. Recall that (since s is constructible) t is a proper prefix of $\mathsf{top}^k(s)$, and t' is a proper prefix of $\mathsf{top}^k(s')$, of the same length, so they are equal due to $\theta(s) = \theta(s')$. □

B Missing proofs for Section 4

Let us prove (for $k \in \{3, 4\}$) that $t \in vis^k(s)$ if and only if in $\mathcal{G}^{con}(\mathcal{A})$ it holds

$$\exists u.(u \xrightarrow{\mathsf{pop}^k} s \wedge link^k(u) \wedge u \xrightarrow{collapse} t).$$

Assume that $t \in vis^k(s)$. One possibility is that $t = s$. Then as u we can take $\mathsf{push}^1_{\star,k}(\mathsf{push}^k(s))$. Otherwise, by definition, s contains the atom $(\star, k, \mathsf{top}^k(t))$ in its topmost $(k-1)$-stack, so $\mathsf{push}^k(s)$ contains this atom in its topmost $(k-1)$-stack as well; let x be its position. Then the formula is satisfied for $u := (\mathsf{push}^k(s))_{\leq x}$. Notice that the stack u is constructible: it can be constructed from $\mathsf{push}^k(s)$ by a sequence of pop operations.

Oppositely, assume that the formula is satisfied. Since $link^k(u)$ holds and $t = \mathsf{collapse}(u)$, we have $\mathsf{top}^0(u) = (\star, k, \mathsf{top}^k(t))$, and for $k = 3$ additionally

$\mathsf{pop}^4(u) = \mathsf{pop}^4(t)$ (which gives $\mathsf{pop}^4(s) = \mathsf{pop}^4(t)$ since $s = \mathsf{pop}^3(u)$). We apply Proposition A.1 to stack u and the position of the topmost atom in u. It gives us that either $\mathsf{top}^k(t) = \mathsf{pop}(\mathsf{top}^k(u)) = \mathsf{top}^k(\mathsf{pop}^k(u))$, or there is an atom $(\star, k, \mathsf{top}^k(t))$ in $\mathsf{top}^{k-1}(\mathsf{pop}^k(u))$. Because $s = \mathsf{pop}^k(u)$, either $\mathsf{top}^k(t) = \mathsf{top}^k(s)$ (so $t = s$), or there is an atom $(\star, k, \mathsf{top}^k(t))$ in $\mathsf{top}^{k-1}(s)$. By definition this implies that $t \in vis^k(s)$.

C Missing proofs for Section 5

Proof (Lemma 5.3). Let $d' := d + 2^{d+1}$. Let v be a valuation witnessing that $(S_1, \dots, S_m) \hookrightarrow_{d''} (s_1, \dots, s_m)$, i.e. such that $s_i = \overline{v}(S_i)$ for each $i \leq m$. W.l.o.g. assume that v is defined only on the finitely many constants appearing in all the S_i. For each stack s let x_s be a fresh constant of the same order as s (that is, v is undefined for x_s, and $x_s \neq x_{s'}$ for $s \neq s'$). We take w equal to v on the constants on which v is defined, and such that $w(x_s) = s$ for each s.

For each stack $s \in \mathsf{ts}_{\leq d}(s_{m+1})$ we define a generalized stack $\mathsf{repl}(s)$, by induction on the size of s (e.g. on $|\mathsf{ts}_{<\infty}(s)|$):

– if $s \in \mathsf{ts}_{<d}(s_{m+1})$, we take

$$\mathsf{repl}(s) := \begin{cases} first^k(\mathsf{repl}(t)) & \text{if } s = \mathsf{first}^k(t), \\ \mathsf{app}^k(\mathsf{repl}(t), \mathsf{repl}(u)) & \text{if } s = \mathsf{app}^k(t, u), \\ \mathsf{cons}(a, k, []) & \text{if } s = \mathsf{cons}(a, k, []), \\ \mathsf{cons}(a, k, \mathsf{repl}(t)) & \text{if } s = \mathsf{cons}(a, k, t); \end{cases}$$

– otherwise, if $s = \overline{w}(S)$ for some $S \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'}(S_i)$, we take $\mathsf{repl}(s) := S$;
– otherwise, we take $\mathsf{repl}(s) := x_s$.

At the end we take $S_{m+1} := \mathsf{repl}(s_{m+1})$. Notice that the second case in the definition does not introduce ambiguity: there is at most one such S since $(S_1, \dots, S_m) \hookrightarrow_{d''} (s_1, \dots, s_m)$ holds, and $d' \leq d''$.

First observe that repl is a bijection between $\mathsf{ts}_{\leq d}(s_{m+1})$ and $\mathsf{ts}_{\leq d}(S_{m+1})$ preserving depth, and that $\overline{w}$ is the inverse bijection, that is for $e \leq d$ it holds $\mathsf{ts}_{=e}(S_{m+1}) = \{\mathsf{repl}(s) \mid s \in \mathsf{ts}_{=e}(s_{m+1})\}$, and for $s \in \mathsf{ts}_{\leq d}(s_{k+1})$ it holds $\overline{w}(\mathsf{repl}(s)) = s$.

It follows that no element of $\mathsf{ts}_{<d}(S_{m+1})$ is a constant (since we use the first rule of the definition of repl for $s \in \mathsf{ts}_{<d}(s_{m+1})$). Consider an element of $\mathsf{ts}_{=d}(S_{m+1})$; it is of the form $\mathsf{repl}(s)$ for some $s \in \mathsf{ts}_{=d}(s_{m+1})$. If the first case of the definition of $\mathsf{repl}(s)$ is used ($s \in \mathsf{ts}_{<d}(s_{m+1})$), we have $\mathsf{repl}(s) \in \mathsf{ts}_{<d}(S_{m+1})$. In the second case we have $\mathsf{repl}(s) \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'}(S_i)$. In the last case, $\mathsf{repl}(s) = x_s$ is a constant not belonging to $\bigcup_{i \leq m} \mathsf{ts}_{<\infty}(S_i)$. It follows that S_{m+1} is d-normalized with respect to $(S_1, \dots, S_m)$.

The fact that $(S_1, \dots, S_m, S_{m+1}) \hookrightarrow_d (s_1, \dots, s_m, s_{m+1})$ will be witnessed by valuation w. It still holds $s_i = \overline{w}(S_i)$ for $i \leq m$; we also have $s_{m+1} = \overline{w}(S_{m+1})$. No element of $\bigcup_{i \leq m+1} \mathsf{ts}_{<d}(S_i)$ is a constant. It remains to prove for $T, U \in \bigcup_{i \leq m+1} \mathsf{ts}_{\leq d}(S_i)$ such that $\overline{w}(T) = \overline{w}(U)$ that it holds $T = U$. It

is true by assumption when $T, U \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d}(S_i)$. We also have this when $T, U \in \mathsf{ts}_{\leq d}(S_{m+1})$, since $\overline{w}$ is a bijection on $\mathsf{ts}_{\leq d}(S_{m+1})$. Thus assume that $T \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d}(S_i)$ and $U \in \mathsf{ts}_{\leq d}(S_{m+1})$ (the fourth, remaining case is symmetric). In this case $U = \mathsf{repl}(\overline{w}(U)) = \mathsf{repl}(\overline{w}(T))$ (assuming that $\overline{w}(T) = \overline{w}(U)$), so we should prove that $T = \mathsf{repl}(\overline{w}(T))$. For our inductive proof it is important to consider the parameter

$$r(T) := |\{t \in \mathsf{ts}_{\leq d}(s_{m+1}) \mid \overline{w}(T) \in \mathsf{ts}_{<\infty}(t)\}|,$$

that is the number of stacks in $\mathsf{ts}_{\leq d}(s_{m+1})$ which contain $\overline{w}(T)$ as a substack. It is enough to prove the following statement

> Let T be a generalized stack such that $\overline{w}(T) \in \mathsf{ts}_{\leq d}(s_{m+1})$ and $T \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d + r(T)}(S_i)$. Then $T = \mathsf{repl}(\overline{w}(T))$.

This is proved by induction on the structure of T. First observe that $r(T) \leq |\mathsf{ts}_{\leq d}(s_{m+1})| \leq 2^{d+1}$ (since s_{m+1} can be seen as a tree, in which each node has at most two children, and we count the nodes at distance at most d from the root), so $d + r(T) \leq d' \leq d''$. One case is that $\overline{w}(T) \in \mathsf{ts}_{<d}(s_{m+1})$. Notice that T is not a constant (there are no constants in $\bigcup_{i \leq m} \mathsf{ts}_{<d''}(S_i) \supseteq \bigcup_{i \leq m} \mathsf{ts}_{\leq d + r(T)}(S_i)$). To fix attention consider the case that T is of the form $\mathsf{first}^k(S)$ for some S; the other cases ($\mathsf{app}^k(S, U)$, $\mathsf{cons}(a, k, [])$, $\mathsf{cons}(a, k, S)$) are similar. We have $\overline{w}(T) = \mathsf{first}^k(\overline{w}(S))$, and $\overline{w}(S) \in \mathsf{ts}_{\leq d}(s_{m+1})$. Moreover $r(S) > r(T)$, because stacks containing $\overline{w}(T)$ as substack also contain $\overline{w}(S)$, and $\overline{w}(S)$ itself contains $\overline{w}(S)$ but not $\overline{w}(T)$; thus $S \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d + r(S)}(S_i)$. From the induction assumption we obtain $S = \mathsf{repl}(\overline{w}(S))$, so

$$T = \mathsf{first}^k(S) = \mathsf{first}^k(\mathsf{repl}(\overline{w}(S))) = \mathsf{repl}(\mathsf{first}^k(\overline{w}(S))) = \mathsf{repl}(\overline{w}(T))$$

(in the third equality we use that $\overline{w}(T) \in \mathsf{ts}_{<d}(s_{m+1})$). The other case is that $\overline{w}(T) \notin \mathsf{ts}_{<d}(s_{m+1})$. Then $\mathsf{repl}(\overline{w}(T)) = T$, because we use the second case in the definition of repl. $\square$

Proof (Lemma 5.4). Let $d' := d + 2^{d+1}$. Let v be a valuation witnessing that $(S_1, \dots, S_m) \hookrightarrow_{d''} (s_1, \dots, s_m)$, i.e. such that $s_i = \overline{v}(S_i)$ for each $i \leq m$. W.l.o.g. assume that v is defined only on the finitely many constants appearing in all the S_i. We will extend v to a valuation w defined also on constants appearing in S_{m+1}. The key point is to map each of these constants into a stack which does not appear anywhere else. One way of realizing that is the following. Let N be a "big" number, such that $\mathsf{ts}_{=N}(s_i) = \emptyset$ for each $i \leq m$, and $\mathsf{ts}_{=N}(S_{m+1}) = \emptyset$.

Consider the following stacks (for some fixed $a \in \mathit{\Gamma}$):

$$\begin{aligned}
&t_0^0 := (a, 1, []),\\
&t_1^0 = (a, 1, [t_0^0]),\\
&t_i^k := [t_i^{k-1}] \qquad \text{for } k \in \{1, \dots, n\}, i \in \{0, 1\},\\
&u_j^0 := (a, 1, [\underbrace{t_0^{k-1}, \dots, t_0^{k-1}}_{2N+j}]) \qquad \text{for } j \in \mathbb{N},\\
&u_j^k := [\underbrace{t_0^{k-1}, \dots, t_0^{k-1}}_{2N+j}, t_1^{k-1}] \qquad \text{for } k \in \{1, \dots, n\}, j \in \mathbb{N}.
\end{aligned}$$

For each constant x^k (of order k) appearing in S_{m+1} but not in any S_i for $i \leq m$, we define $w(x^k)$ to be some u_j^k, different for each constant.

We take $s_{m+1} := \overline{w}(S_{m+1})$. By assumption $s_i = \overline{w}(S_i)$ for each $i \leq m+1$, and no element of $\mathsf{ts}_{\leq d}(S_i)$ is a constant, and for each $T, U \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d}(S_i)$ such that $\overline{w}(T) = \overline{w}(U)$ it holds $T = U$. As in the previous proof, we define

$$r(U) := |\{t \in \mathsf{ts}_{\leq d}(s_{m+1}) \mid \overline{w}(U) \in \mathsf{ts}_{<\infty}(t)\}|,$$

We will prove that for each $T \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'+r(U)}(S_i)$ and $U \in \mathsf{ts}_{\leq d}(S_{m+1})$ such that $\overline{w}(T) = \overline{w}(U)$ it holds $T = U$ (in particular it holds for all $T \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d}(S_i)$, and even for $T \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'}(S_i)$). This is proved by induction on the structure of U. Assume first that $U \in \mathsf{ts}_{<d}(S_{m+1})$. Then U is not a constant. To fix attention consider the case that U is of the form $\mathsf{first}^k(U')$ for some $U' \in \mathsf{ts}_{\leq d}(S_{m+1})$; the other cases ($\mathsf{app}$, cons) are similar. Notice that $r(U) < r(U') \leq 2^{d+1}$. In particular $T \in \bigcup_{i \leq m} \mathsf{ts}_{<d''}(S_i)$, so T is not a constant. Due to $\overline{w}(T) = \overline{w}(U)$ we have $T = \mathsf{first}^k(T')$ for some T' such that $\overline{w}(T') = \overline{w}(U')$. Since $T' \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'+r(U')}(S_i)$, by induction assumption we have $T' = U'$, which implies $T = U$. Another case is that $U \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'}(S_i) \subseteq \bigcup_{i \leq m} \mathsf{ts}_{\leq d''}(S_i)$. Then we have $T = U$ thanks to $(S_1, \dots, S_m) \hookrightarrow_{d''} (s_1, \dots, s_m)$. Because of d-normalization, the only remaining case is that U is a constant not appearing in any S_i for $i \leq m$. But then $\overline{w}(U)$ is one of u_j^k, so it cannot be equal to $\overline{w}(T)$ (we have $\mathsf{ts}_{=N}(\overline{w}(U)) \neq \emptyset = \mathsf{ts}_{=N}(\overline{w}(T))$).

Finally, we prove that for each $T, U \in \mathsf{ts}_{\leq d}(S_{m+1})$ such that $\overline{w}(T) = \overline{w}(U)$ it holds $T = U$. This is again induction on the structure of T (or U). One possibility is that both T, U are in $\mathsf{ts}_{<d}(S_{m+1})$. Then these are not constants, so they use the same constructors (thanks to $\overline{w}(T) = \overline{w}(U)$), e.g. $T = \mathsf{first}^k(T')$ and $U = \mathsf{first}^k(U')$ with $\overline{w}(T') = \overline{w}(U')$ (similarly for app and cons). We have $T', U' \in \mathsf{ts}_{\leq d}(S_{m+1})$, so induction assumption implies $T' = U'$ which gives $T = U$. Other possibility is that $T \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'}(S_i)$. Then $T = U$ by the previous paragraph; similarly when $U \in \bigcup_{i \leq m} \mathsf{ts}_{\leq d'}(S_i)$. Thanks to d-normalization, the only remaining case is that one of T, U, say U, is a constant not appearing in any S_i for $i \leq m$. Then $w(U) = u_j^k$ for some j, k. Assume first that $\mathsf{ts}_{<\infty}(T)$ also contains a constant not appearing in any S_i for $i \leq m$. If T itself is such a constant, we necessarily have $T = U$. If this constant appears as a proper substack of T, some $u^{k'}_{j'}$ is a proper substack of u^k_j, which is impossible by construction of these stacks. The opposite case is that $\mathsf{ts}_{<\infty}(T)$ does not contain such constants. Then $\overline{w}(T)$ is "smaller" than u^k_j. More precisely, we have $\mathsf{ts}_{=N}(T) = \emptyset$; in T at depth $< N$ there might be some constants which are replaced in $\overline{w}(T)$ by some stacks; these stacks are however substacks of some s_i, so their $\mathsf{ts}_{=N}$ is empty, and thus $\mathsf{ts}_{=2N}(\overline{w}(T)) = \emptyset \neq \mathsf{ts}_{=2N}(u^k_j)$.

This proves that $(S_1, \dots, S_m, S_{m+1}) \hookrightarrow_d (s_1, \dots, s_m, s_{m+1})$. □

Proof (Lemma 5.5). Let v be a valuation such that $s = \overline{v}(S)$ and $t = \overline{v}(T)$. We recall that there are no constants in S or T on depth $\leq n$. Hence, for $0 \leq k \leq n$, we can find an extended k-stack $S^k_{\mathsf{top}} \in \mathsf{ts}_{\leq n-k}(S)$ such that $\overline{v}(S^k_{\mathsf{top}}) = \mathsf{top}^k(s)$:

$$S^n_{\mathsf{top}} := S,$$
$$S^k_{\mathsf{top}} := \begin{cases} U & \text{if } S^{k+1}_{\mathsf{top}} = \mathsf{first}^{k+1}(U), \\ U' & \text{if } S^{k+1}_{\mathsf{top}} = \mathsf{app}^{k+1}(U, U') \end{cases} \qquad \text{for } k < n.$$

For $1 \leq k \leq n$ we can also find $S^k_{\mathsf{pop}} \in \mathsf{ts}_{\leq n-k+1}(S)$ or $S^k_{\mathsf{pop}} = []$ such that $\overline{v}(S^k_{\mathsf{pop}}) = \mathsf{pop}(\mathsf{top}^k(s))$:

$$S^k_{\mathsf{pop}} := \begin{cases} [] & \text{if } S^k_{\mathsf{top}} = \mathsf{first}^k(U), \\ U & \text{if } S^k_{\mathsf{top}} = \mathsf{app}^k(U, U'). \end{cases}$$

Additionally, S^0_{top} is of the form $\mathsf{app}(a_S, l_S, S_{\mathsf{link}})$ where $S_{\mathsf{link}} \in \mathsf{ts}_{\leq n+1}(S)$ or $S_{\mathsf{link}} = []$. Observe that a_S is the label of the topmost atom in S. Similarly, we can define T^k_{top}, T^k_{pop}, T_{link}, a_T, l_T, starting from T instead of S.

The key property of the $\hookrightarrow_{n+1}$ relation which we exploit here is that for $U, U' \in \mathsf{ts}_{\leq n+1}(S) \cup \mathsf{ts}_{\leq n+1}(T)$ we have $\overline{v}(U) = \overline{v}(U')$ if and only if $U = U'$. In particular $s = \overline{v}(S) = \overline{v}(T) = t$ if and only if $S = T$. Now we consider each possible operation as θ.

We observe that $t = \mathsf{pop}^k(s)$ if and only if $\mathsf{pop}(\mathsf{top}^i(s)) = \mathsf{pop}(\mathsf{top}^i(t))$ for $i \in \{k+1, \dots, n\}$, and $\mathsf{pop}(\mathsf{top}^k(s)) = \mathsf{top}^k(t)$; this can be expressed equivalently as $S^i_{\mathsf{pop}} = T^i_{\mathsf{pop}}$ for $i \in \{k+1, \dots, n\}$, and $S^k_{\mathsf{pop}} = T^k_{\mathsf{top}}$.

When a $\mathsf{collapse}$ operation is performed from s, it is of order l_S. Thus we have $t = \mathsf{collapse}(s)$ if and only if $\mathsf{pop}(\mathsf{top}^i(s)) = \mathsf{pop}(\mathsf{top}^i(t))$ for $i \in \{l_S + 1, \dots, n\}$, and the link target in $\mathsf{top}^0(s)$ is $\mathsf{top}^{l_S}(t)$; this holds if and only if $S^i_{\mathsf{pop}} = T^i_{\mathsf{pop}}$ for $i \in \{l_S + 1, \dots, n\}$, and $S_{\mathsf{link}} = T^{l_S}_{\mathsf{top}}$ (in particular $S_{\mathsf{link}} \neq []$).

We have $t = \mathsf{rew}_a(s)$ if and only if $\mathsf{pop}(\mathsf{top}^i(s)) = \mathsf{pop}(\mathsf{top}^i(t))$ for $i \in \{1, \dots, n\}$, and $\mathsf{lb}(\mathsf{top}^0(t)) = a$, and the links (including the link order) in $\mathsf{top}^0(s)$ and in $\mathsf{top}^0(t)$ are the same; this holds if and only if $S^i_{\mathsf{pop}} = T^i_{\mathsf{pop}}$ for $i \in \{1, \dots, n\}$, and $a_T = a$, and $S_{\mathsf{link}} = T_{\mathsf{link}}$.

We have $t = \mathsf{push}^k(s)$ if and only if $s = \mathsf{pop}^k(t)$ and $\mathsf{top}^{k-1}(s) = \mathsf{top}^{k-1}(t)$; thus to the (reversed) conditions for pop^k we add that $S^{k-1}_{\mathsf{top}} = S^{k-1}_{\mathsf{top}}$.

Finally, $t = \mathsf{push}^1_{a,k}(s)$ holds if and only if $s = \mathsf{pop}^1(t)$, and $\mathsf{lb}(\mathsf{top}^0(t)) = a$, and the link in $\mathsf{top}^0(t)$ is $\mathsf{pop}(\mathsf{top}^k(s))$; the latter two conditions can be expressed as $a_T = a$ and $T_{\mathsf{link}} = S^k_{\mathsf{pop}}$. □

Now we prove that there are only finitely many d-normalized generalized stacks, up to renaming of constants. For that, fix some order on the set of constants of each order k: $x^k_1, x^k_2, \dots$. We say that a generalized stack S_{m+1} is *fully d-normalized* with respect to $(S_1, \dots, S_m)$ if it is d-normalized with respect to $(S_1, \dots, S_m)$, and a constant x^k_j is used in S_{m+1} only when either it was used in some S_i for $i \le m$, or all constants $x^k_1, \dots, x^k_{j-1}$ are also used in S_{m+1}.

Lemma C.1. *Let $S_1, \dots, S_m$ be extended stacks, and let $d \in \mathbb{N}$. Then there are finitely many generalized stacks S_{m+1} which are fully d-normalized with respect to $(S_1, \dots, S_m)$.*

Proof. We will bound the size of S_{m+1}. The condition that S_{m+1} is fully d-normalized determines which fresh constants can be used, so there are only finitely many S_{m+1} of fixed size.

Fix some S_{m+1} which is d-normalized with respect to $(S_1, \dots, S_m)$. Let $N \ge 1$ be such that $\bigcup_{i \le m} \mathsf{ts}_{=N}(S_i) = \emptyset$. We define

$$r(T) := |\{U \in \mathsf{ts}_{\le d}(S_{m+1}) \mid T \notin \mathsf{ts}_{<\infty}(U)\}|.$$

We will prove for $T \in \mathsf{ts}_{\le d}(S_{m+1})$ that $\mathsf{ts}_{=N+r(T)}(T) = \emptyset$. This is proved by induction on the structure of T. First case is that $T \in \bigcup_{i \le m} \mathsf{ts}_{<\infty}(S_i)$. Then by definition of N we have $\mathsf{ts}_{=N+r(T)}(T) = \emptyset$. Second case is that T is a constant; then as well $\mathsf{ts}_{=N+r(T)}(T) = \emptyset$. According to the definition of d-normalization, the only remaining case is that $T \in \mathsf{ts}_{<d}(S_{m+1})$. Assume for example that $T = \mathsf{first}^k(T')$ (the cases of app and cons are similar). By induction $\mathsf{ts}_{=N+r(T')}(T') = \emptyset$. We conclude that $\mathsf{ts}_{=N+r(T)}(T) = \emptyset$ since $r(T) > r(T')$ (stacks not containing T' as substack also do not contain T, and T' itself contains T' but not T).

In particular $\mathsf{ts}_{=N+r(S_{m+1})}(S_{m+1}) = \emptyset$, so as well $\mathsf{ts}_{=N+2^{d+1}}(S_{m+1}) = \emptyset$, since $r(S_{m+1}) \le 2^{d+1}$. $\square$

Next, we describe in details the decision procedure. Assume that in the logic we only have the existential quantifier, the $\wedge$ and $\neg$ connectives, and atomic formulae. For a formula φ we define a number $d(\varphi)$ as follows:

$$d(\varphi) = \begin{cases} n+1 & \text{if } \varphi \text{ atomic,} \\ \max(d(\psi_1), d(\psi_2)) & \text{if } \varphi = \psi_1 \wedge \psi_2, \\ d(\psi) & \text{if } \varphi = \neg\psi, \\ d(\psi) + 2^{d(\psi)+2} & \text{if } \varphi = \exists x.\psi. \end{cases}$$

Next, we will define a procedure $\mathsf{Check}_{m,\mathcal{A}}(\varphi, q_1, S_1, \dots, q_m, S_m)$, which returns True or False, and whose arguments are states $q_1, \dots, q_m$, extended stacks $S_1, \dots, S_m$, and an FO formula $\varphi(x_1, \dots, x_m)$. The procedure will be such that for any stacks $(s_1, \dots, s_m)$ such that $(S_1, \dots, S_m) \hookrightarrow_{d(\varphi)} (s_1, \dots, s_m)$, it holds $\mathcal{G}^{ano}(\mathcal{A}) \vdash \varphi((q_1, s_1), \dots, (q_m, s_m))$ if and only if $\mathsf{Check}_{m,\mathcal{A}}(\varphi, q_1, S_1, \dots, q_m, S_m)$ returns True. Then we can evaluate a sentence φ by just calling $\mathsf{Check}_{0,\mathcal{A}}(\varphi)$, since it holds $() \hookrightarrow_{d(\varphi)} ()$. We define the procedure as follows, by induction on the structure of φ:

– If $\varphi = \neg\psi$ we just negate $\mathsf{Check}_{m,\mathcal{A}}(\psi, q_1, S_1, \dots, q_m, S_m)$.
– If $\varphi = \psi_1 \wedge \psi_2$ we return True if and only if $\mathsf{Check}_{m,\mathcal{A}}(\psi_i, q_1, S_1, \dots, q_m, S_m)$ returns True for some $i \in \{1, 2\}$. This is correct since $(S_1, \dots, S_m) \hookrightarrow_{d(\varphi)} (s_1, \dots, s_m)$ implies $(S_1, \dots, S_m) \hookrightarrow_{d(\psi_i)} (s_1, \dots, s_m)$ (thanks to $d(\psi_i) \leq d(\varphi)$).
– Assume that $\varphi = (x_i = x_j)$. Notice that $(S_1, \dots, S_m) \hookrightarrow_{d(\varphi)} (s_1, \dots, s_m)$ implies $(S_i, S_j) \hookrightarrow_{n+1} (s_i, s_j)$. We check whether $s_i = s_j$ using Lemma 5.5, and whether $q_i = q_j$.
– Assume that $\varphi = (x_i \xrightarrow{c} x_j)$. We should return True if for some transition (q_i, a, c, θ, q_j) of $\mathcal{A}$ it holds $a = \mathsf{lb}(\mathsf{top}^0(s_i))$ and $s_j = \theta(s_i)$. We check this for all transitions using Lemma 5.5 (again we use that $(S_i, S_j) \hookrightarrow_{n+1} (s_i, s_j)$).
– Assume that $\varphi = b(x_i)$ for some predicate b of $\mathcal{A}$. We should return True if for some triple (q_i, a, b) in the predicate relation of $\mathcal{A}$ it holds $a = \mathsf{lb}(\mathsf{top}^0(s_i))$. We check this for all triples using Lemma 5.5, observing that $(S_i, S_i) \hookrightarrow_{n+1} (s_i, s_i)$.
– Finally, assume that $\varphi = \exists x_{i+1}.\psi$ (w.l.o.g. we assume that the variable is called x_{i+1}). Then we return True if for some state q_{m+1} and for some generalized stack S_{m+1} which is fully $d(\psi)$-normalized with respect to $(S_1, \dots, S_m)$ procedure $\mathsf{Check}_{m+1,\mathcal{A}}(\psi, q_1, S_1, \dots, q_m, S_m, q_{m+1}, S_{m+1})$ returns True. Recall that due to Lemma C.1 there are only finitely many S_{m+1} to check. To see that this is correct take some stacks $s_1, \dots, s_m$ such that $(S_1, \dots, S_m) \hookrightarrow_{d(\varphi)} (s_1, \dots, s_m)$. Assume first that $\mathsf{Check}_{m,\mathcal{A}}(\varphi, q_1, S_1, \dots, q_m, S_m)$ returns True, so that there exists a state q_{m+1} and a generalized stack S_{m+1} which is fully $d(\psi)$-normalized with respect to $(S_1, \dots, S_m)$, and such that $\mathsf{Check}_{m+1,\mathcal{A}}(\psi, q_1, S_1, \dots, q_m, S_m, q_{m+1}, S_{m+1})$ returns True. Then Lemma 5.4 gives us a stack s_{m+1} such that

$$(S_1, \dots, S_m, S_{m+1}) \hookrightarrow_{d(\psi)} (s_1, \dots, s_m, s_{m+1}).$$

By induction assumption $\mathcal{G}^{ano}(\mathcal{A}) \vdash \psi((q_1, s_1), \dots, (q_m, s_m), (q_{m+1}, s_{m+1}))$, which implies that $\mathcal{G}^{ano}(\mathcal{A}) \vdash \varphi((q_1, s_1), \dots, (q_m, s_m))$. Oppositely, assume that $\mathcal{G}^{ano}(\mathcal{A}) \vdash \varphi((q_1, s_1), \dots, (q_m, s_m))$, so that there exists a configuration (q_{m+1}, s_{m+1}) such that $\mathcal{G}^{ano}(\mathcal{A}) \vdash \psi((q_1, s_1), \dots, (q_m, s_m), (q_{m+1}, s_{m+1}))$. Then Lemma 5.3 gives us a generalized stack S_{m+1} which is $d(\psi)$-normalized with respect to $(S_1, \dots, S_m)$, and such that

$$(S_1, \dots, S_m, S_{m+1}) \hookrightarrow_{d(\psi)} (s_1, \dots, s_m, s_{m+1}).$$

Notice that we can safely (it does not influence the $\hookrightarrow_{d(\psi)}$ relation and the $d(\psi)$-normalization) rename constants in S_{m+1} which do not appear in any S_i for $i \leq m$; thus w.l.o.g. we can assume that S_{m+1} is fully $d(\psi)$-normalized. By induction assumption $\mathsf{Check}_{m+1,\mathcal{A}}(\psi, q_1, S_1, \dots, q_m, S_m, q_{m+1}, S_{m+1})$ returns True, so also $\mathsf{Check}_{m,\mathcal{A}}(\varphi, q_1, S_1, \dots, q_m, S_m)$ returns True.