First-Order Logic on CPDA Graphs

Paweł Parys[⋆]

University of Warsaw, Warsaw, Poland
`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 the 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 trees generated by deterministic n-PDA coincide with trees generated by *safe* order-n recursion schemes (safety is a syntactic restriction on the recursion scheme). Driven by the question of 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 further investigated by Broadbent [12–15]. He proved that for order 3 (and higher) the FO theory starts to be undecidable. This can be made more precise. Let n_m-CPDA denote an n-CPDA in which we allow collapse links only of one order m. First-order theory is undecidable already on:

– 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, Broadbent 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 two different reachable configurations, they 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 (contrarily 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 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, when stacks are represented as *annotated stacks*. This is an alternative representation of stacks of n-CPDA (defined independently in [16] and [17]), where

[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.

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", which, according to our definition, is not a stack), then (a, k, s) 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 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 2. *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. One has to add extra annotations 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), at 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, and that it does not perform $\mathsf{collapse}$ on links of order 1. 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 an 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:

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

Intuitively, $gn(s)$ is a lower bound for the height of the 3-stack of the CPDA at the moment when s was last modified (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 $s_{\leq x}$ (in the lexicographic order) 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, $\neg 3$).
- Recall that when the atom at x is (a, l, t), then t is a proper prefix of $\mathsf{top}^l(s_{\leq x})$. We attach the markers in t so that this property is preserved, than is in the same way as in $\mathsf{top}^l(s_{\leq x})$.

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 $s_{\leq x}$. Intuitively, when an atom from a new generation is placed above y, in y we keep his $2r$ marker. This is needed to reproduce the $2r$ marker when

y again becomes 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 an 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.

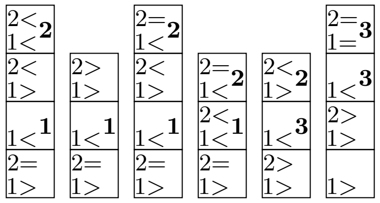

Fig. 1. An example 2-stack (one out of many in a 3-stack). It grows from left to right. We indicate all $1r$ and $2r$ markers, as well as the generation of atoms (bold; no number for generation -1). To calculate the $2<$ marker at positions $(1,3)$, $(3,3)$, and $(4,2)$ we have used position $(1,3)$ 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. *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^k 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 4. *Let s and 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 4 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 (contrarily to the conjecture stated in [15]). On the other hand, in the next section we show decidability when one also allows stacks which are unconstructible. Let us recall that the FO theory is known [15] to be undecidable, if one also allows 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 the graph $\mathcal{G}^{con}(\mathcal{A})$, where $\mathcal{A}$ is the 4-CPDA 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 5. *FO is undecidable on $\mathcal{G}^{con}(\mathcal{A})$.*

We reduce from 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 at most 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 6. *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 $\mathsf{top}^3(s_G) = t_k$, 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

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

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

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

We notice that $vis^4(s_G)$ 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^i 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 5, 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 7. *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. When the size of the described top part of the stack is fixed, there are only finitely many such descriptions. For each quantifier in the FO sentence we will be checking all possible descriptions of fixed size (of course the size of the described part has to decrease with each next variable). 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 generalized 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 FO we can also talk about equality of stacks, so we are interested in valuations which applied to different generalized stacks give different stacks. This is described by a relation $\hookrightarrow_d$ which is defined as follows. Let $S_1, \dots, S_m$ (for $m \geq 0$) be generalized 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 8. 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 that $(\mathsf{app}^2(\mathsf{first}^2(y^1),\mathsf{app}^1(x^1,\mathsf{cons}(c,1,[])))) \hookrightarrow_2 (s)$; 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 just cut our stack at one, fixed depth, and place constants in all places at this depth. In fact, for some stacks, we need to place some constants exponentially deeper than other constants. As a consequence, our algorithm will be nonelementary (we have to increase d exponentially with each quantifier).

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 a 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 9. *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 $S \in \bigcup_{i \le m} \mathsf{ts}_{\le d'}(S_i)$, 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 10. *Let $S_1, \dots, S_m$ and S_{m+1} (for $m \ge 0$) 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 $i \le m$ into "fresh" stacks, such that none of them is a substack of any other nor of any s_i for $i \le m$ (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 formulae can evaluated on the level of generalized stacks related by the $\hookrightarrow_{n+1}$ to the actual stacks.

Lemma 11. *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:*

– $\mathsf{lb}(\mathsf{top}^0(s))$,
– *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 we deal with atomic formulae using Lemma 11.

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. In Czumaj, A., Mehlhorn, K., Pitts, A.M., Wattenhofer, R., eds.: ICALP (2). Volume 7392 of Lecture Notes in Computer Science., Springer (2012) 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. In Czumaj, A., Mehlhorn, K., Pitts, A.M., Wattenhofer, R., eds.: ICALP (2). Volume 7392 of Lecture Notes in Computer Science., Springer (2012) 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