The Limits of Decidability for First Order Logic on CPDA Graphs

Christopher H. Broadbent

The Limits of Decidability for First Order Logic on CPDA Graphs*

Christopher H. Broadbent

LIAFA, Université Paris Diderot-Paris 7 and CNRS
Case 7014 F-75205 Paris Cedex 13, France
broadbent@liafa.jussieu.fr

Abstract

Higher-order pushdown automata (n-PDA) are abstract machines equipped with a nested 'stack of stacks of stacks'. Collapsible pushdown automata (n-CPDA) extend these devices by adding 'links' to the stack and are equi-expressive for tree generation with simply typed λY terms. Whilst the configuration graphs of HOPDA are well understood, relatively little is known about the CPDA graphs. The order-2 CPDA graphs already have undecidable MSO theories but it was only recently shown by Kartzow [9] that first-order logic is decidable at the second level. In this paper we show the surprising result that first-order logic ceases to be decidable at order-3 and above. We delimit the fragments of the decision problem to which our undecidability result applies in terms of quantifer alternation and the orders of CPDA links used. Additionally we exhibit a natural sub-hierarchy enjoying limited decidability.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic

Keywords and phrases Collapsible Pushdown Automata, First Order Logic, Logical Reflection

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.589

1 Introduction

Higher-order pushdown automata generalise traditional pushdown automata by allowing the stack to contain other stacks rather than just atomic elements. These devices are closely related to *recursion schemes*, which are essentially simply typed λY terms that generate a single infinite tree. Enjoying decidable μ-calculus theories, the class of trees generated by recursion schemes shows a lot of promise as a model for verifying higher-order functional programs [12, 13]. Unfortunately n-PDA expressively coincide with order-n recursion schemes only when the latter satisfy a property called *safety* [10], unsafe recursion schemes are strictly more expressive [15, 14]. Hence a more powerful automaton is needed, which motivates order-n *collapsible pushdown automata* (n-CPDA) [7]. Inspired by *panic automata* [11], which can be viewed as the special case at order-2, atomic elements in collapsible stacks eminate 'links' that target a component of the stack further below.

We concern ourselves here with configuration *graphs* of these automata with *reachable* states as vertices and transitions as edges. It is particularly fruitful to additionally consider the 'ϵ-closures' of such graphs, 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* [6, 3, 5], which is defined independently in terms of graph transformations. This deep result has as a consequence that every n-PDA graph has decidable MSO theory.

* The author is funded by La Fondation Sciences Mathématiques de Paris.

29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 589–600
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

	2-CPDA /w ϵ-cl.	3_2-CPDA	3_2-CPDA /w ϵ-cl.	n_n-CPDA (/w ϵ-cl.) $n \geq 3$	$n_{n,(n-1)}$-CPDA $n \geq 4$ /wo ϵ-cl. $n \geq 3$ /w ϵ-cl.	n_m-CPDA $n \geq 4$, $m \leq n-2$	n-PDA /w ϵ-cl. all n
Σ_1	Dec [9]	Dec [1]	Dec [1]	**Dec**	?	**Und**	Dec [5]
Σ_2	Dec [9]	Dec [1]	**Und**	**Und**	**Und**	**Und**	Dec [5]
FO	Dec [9]	Dec [1]	**Und**	**Und**	**Und**	**Und**	Dec [5]
MSO	Und [7]	Und [7]	Und [7]	Und [7]	Und [7]	Und [7]	Dec [5]
μ-c.	Dec [11]	Dec [7]	Dec [7]	Dec [7]	Dec [7]	Dec [7]	Dec [10]

Table 1 Summary of (un)decidability results known to date. Those in bold are from this paper. The notation '$n_{m,m'}$-CPDA' means an n-CPDA that only uses links of orders m and m'.

Unfortunately there is even a 2-CPDA graph that has *un*decidable MSO theory [7]. Nevertheless the local nature of first-order logic meant it was widely assumed the first-order theories would still enjoy decidability and indeed Kartzow saw that the ϵ-closures of 2-CPDA graphs are *tree automatic* [9] and so do indeed have decidable first-order theory. Our first contribution is to show that Kartzow's result cannot be extended to higher-orders in full generality. At order-3 we get undecidability when we consider Σ_2-sentences, namely those with quantifier alternation of the form $\forall\exists$. If we allow order-3 links we do not even need ϵ-closure for this. This result is interesting in itself, but we also gain some insight into what links 'mean' in terms of 3-CPDA *graphs*. On the one hand links can act as 'place holders' that allow first-order logic to compare internal components of a single order-3 stack rather than just two order-3 stacks in their entirety. This is the core of the undecidability result, which goes via a reduction from Post's Correspondence Problem [16]. Additionally order-3 links provide edges in the graph that are 'non-local' in nature, allowing ϵ-closure to be eliminated as a requirement for undecidability. At order-4 we get that even the Σ_1-theory is undecidable—*viz.* the theory consisting of sentences *without any* quantifier alternation.

Our second contribution introduces a technique to tackle the Σ_1-theories of CPDA graphs. Making use of *logical reflection* [2], which enables CPDA to 'know' which μ-calculus sentences they satisfy at any given point, we define a notion of *monotonic CPDA* that constructs all its reachable configurations without ever destroying an order-$(n-1)$ stack. This has some parallels with Carayol's work on canonical sequences of operations witnessing the constructibility of n-PDA stacks [4]. If an n-CPDA only has 'order-n links', monotonicity allows us to eliminate them, thereby producing an n-PDA with decidable MSO theory, and leading to the decidability of the Σ_1 theory of the original n-CPDA. This is a graph analogue of Aehlig *et al.'s* [8] in which links are eliminated from 2-CPDA to produce an equivalent word-generating 2-PDA. This decidability result is the first about a fragment of first-order logic on n-CPDA at *all* orders and shows restricting links can recover some decidability.

We emphasise that even the undecidability results without ϵ-closure still require a restriction of the domain to configurations *reachable from the origin*. Unlike word-automatic structures, however, this undecidability is non-trivial, as indicated by the positive results.

2 Preliminaries

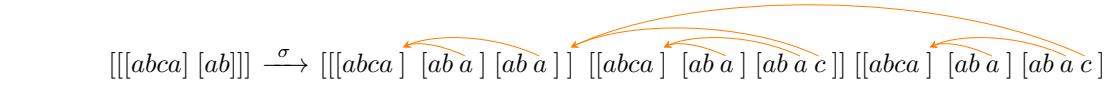
2.1 Higher-Order Stacks

Let us fix a stack-alphabet Γ. For higher-order automata this alphabet must be finite, but it is convenient for definitions to allow it to be infinite. An *order*-1 stack over Γ is just a string of

the form $[\ \gamma\ ]$ where $\gamma \in \Gamma^*$. Let us refer to the set of order-1 stacks over Γ as $stack_1(\Gamma)$. For $n \in \mathbb{N}$ the set of *order*-$(n+1)$ stacks is: $stack_{n+1}(\Gamma) := stack_1(stack_n(\Gamma))$. We allow the following operations on an order-1 stack s for every $a \in \Gamma$: $push_1^a([\ a_1 \cdots a_m\ ]) := [\ a_1 \cdots a_m a\ ]$, $pop_1([\ a_1 \cdots a_m a_{m+1}\ ]) := [\ a_1 \cdots a_m\ ]$, $nop(s) := s$. We allow the following *order*-$(n+1)$ *operations* on an order-$(n+1)$ stack s, where θ is any *order-n operation*: $push_{n+1}([\ s_1 \cdots s_m\ ]) := [\ s_1 \cdots s_m s_m\ ]$, $pop_{n+1}([\ s_1 \cdots s_m s_{m+1}\ ]) := [\ s_1 \cdots s_m\ ]$, $\theta([\ s_1 \cdots s_m\ ]) := [\ s_1 \cdots \theta(s_m)\ ]$. Where s is an $(n+1)$-stack we write $top_{n+1}(s)$ to denote the top-most (right-most) order-n stack (abusing notation with $top_{n+1}(s) := s$ if s is an n-stack) and $top_1(s)$ as the top atomic element. Let us also write $s \sqsubseteq t$ when $s = [\ s_1 \cdots s_m\ ]$ and t is of the form $[\ s_1 \cdots\ s_{m'}\ ]$ for $m' \leq m$, $s \sqsubset t$ when additionally $s \neq t$ and define $|s| := m$.

2.2 Collapsible Pushdown Stacks

A $push_1^{a,k}$ operation in a CPDA attaches a *k-link* from the newly created a element that points to the k-stack below. The targets of these links are preserved during higher-order *push* operations. Consider the sequence $\sigma = push_1^{a,2}; push_2; push_3; push_1^{c,3}; push_3$, then:



We offer fine control over the orders of links that a collapsible stack may contain; an order-n_S stack is an order-n stack equipped with order-i links for each $i \in S$. Formally the *S-collapsible pushdown alphabet* $\Gamma^{[S]}$ (for $S \subseteq \mathbb{N}$) induced by an alphabet Γ is the set $\Gamma \times S \times \mathbb{N}$. The set of order-$n_S$ collapsible stacks $stack_{n_S}^{\mathcal{C}}(\Gamma)$ is defined by: $stack_{n_S}^{\mathcal{C}}(\Gamma) := stack_n(\Gamma^{[S]})$. An atomic element $(a, l, p) \in \Gamma^{[S]}$ has label a with a link of order l. If $l < n$ the target of a link is the pth $(l-1)$-stack of the l-stack in which the element resides.

We thus introduce the operations $push_1^{a,l}(s) := push_1^{(a,l,|top_{l+1}(s)|-1)}(s)$ and $collapse(s) := pop_l^{|top_{l+1}(s)|-p}$ where $top_1(s) = (a, l, p)$ and θ^m represents the operation θ iterated m times. From now on we will abuse notation and consider $top_1(s) := a$. Write Θ_{n_S} to denote the set of order-n_S collapsible stack operations. Write $\bot_n$ for the empty n-stack.

2.3 The Automata and their Graphs

Let $n \in \mathbb{N}$ and let $S \subseteq [1..n]$. An n_S-CPDA (order-n_S collapsible pushdown automaton) $\mathcal{A}$ is a tuple: $\langle\ \Sigma, \Pi, Q, q_0, \Gamma, R_{a_1}, R_{a_2}, \ldots, R_{a_r}, P_{b_1}, P_{b_2}, \ldots, P_{b_{r'}}\ \rangle$ where Σ is a finite set of transition labels $\{a_1, a_2, \ldots, a_r\}$; Π is a finite set of configuration labels $\{b_1, b_2, \ldots, b_{r'}\}$; Q is a finite set of control-states; $q_0 \in Q$ is an initial control-state; Γ is a finite stack alphabet; each R_{a_i} is the a_i-labelled transition relation with $R_{a_i} \subseteq Q \times \Gamma \times \Theta_{n_S} \times Q$; each P_{b_i} is the b_i-labelled unary predicate specified by $P_{b_i} \subseteq Q \times \Gamma$. Remaining consistent with the definitions in the literature, an n-CPDA is an $n_{[n..2]}$-CPDA and an n-PDA is an $n_\emptyset$-CPDA.

A *configuration* of an n_S-CPDA $\mathcal{A}$ is a pair (q, s) where q is a control-state and s is an n_S-stack. Such a configuration *satisfies* the predicate $b_i \in \Pi$ just in case $(q, top_1(s)) \in P_{b_i}$. We say $\mathcal{A}$ can *a_i-transition from (q, s) to $(q', \theta(s))$*, written $(q, s) \xrightarrow{a_i} (q', \theta(a))$, iff $(q, top_1(s), \theta, q') \in R_{a_i}$. Let us further say that *(q', s') can be reached from (q, s) in $\mathcal{A}$ with path labeled in $\mathcal{L}$* for some $\mathcal{L} \subseteq \Sigma^*$ iff $(q, s) \xrightarrow{a_{i_1}} (q_1, s_1) \xrightarrow{a_{i_2}} (q_2, s_2) \xrightarrow{a_{i_3}} \cdots (q_{m-1}, s_{m-1}) \xrightarrow{a_{i_m}} (q', s')$ for some configurations $(q_1, s_1), \ldots, (q_{m-1}, s_{m-1})$ where $a_{i_1} a_{i_2} a_{i_3} \cdots a_{i_m} \in \mathcal{L}$. We write $(q, s) \xrightarrow{\mathcal{L}} (q', s')$ to mean this. The set of *reachable configurations* of $\mathcal{A}$ is given by:

$$\boldsymbol{R}(\mathcal{A}) := \{\ (q, s)\ :\ (q_0, \bot_n) \xrightarrow{\Sigma^*} (q, s)\ \}$$

The *configuration graph* $\mathcal{G}(\mathcal{A})$ of $\mathcal{A}$ has domain $\boldsymbol{R(\mathcal{A})}$, unary predicates Π and directed edges Σ between configurations. The ϵ-closure of such a graph $\mathcal{G}^{\epsilon}(\mathcal{A})$ is induced from $\mathcal{G}(\mathcal{A})$ by defining a-labelled edges between vertices related by $\xrightarrow{\epsilon^* a}$ in $\mathcal{G}(\mathcal{A})$.

2.4 Logics

We consider first-order logic $\boldsymbol{FO}$ on graphs as it is standardly defined. A $\Sigma_0 = \Pi_0 = \Delta_0$ formula $\phi(x_1, \ldots, x_k)$ is one without any quantifiers. A Σ_{n+1} formula is one of the form $\exists \vec{y}.\phi(\vec{y}, x_1, \ldots, x_k)$ where $\phi(\vec{y}, x_1, \ldots, x_k)$ is Π_n and a Π_{n+1} formula is one of the form $\forall \vec{y}.\phi(\vec{y}, x_1, \ldots, x_k)$ where ϕ is Σ_n. A Δ_n formula is one that is equivalent to both a Σ_n and Π_n formula on every CPDA graph. MSO is $\boldsymbol{FO}$ extended with variables ranging over sets. Transitive closure logic $\boldsymbol{FO(TC)}$ is $\boldsymbol{FO}$ together with a binary predicate $\overline{\phi(x,y)}$ for every formula of $\boldsymbol{FO}$ $\phi(x, y)$ with two variables, which denotes the transitive closure of the relation defined by $\phi(x, y)$. When the formulae transitively closed are restricted to Δ_1 we call the logic $\boldsymbol{FO(TC[\Delta_1])}$. A *sentence* is a formula with no free variables.

Note that we often allow ourselves to assume additional edges in the graph when writing formulae. For example $z \xrightarrow{pop_3;pop_2} z'$ means that we perform pop_3 and then pop_2 on the stack of z and move into a fixed distinguished control-state to result in a configuration z'. Thus $x \xrightarrow{pop_3;pop_2} z \ \wedge \ y \xrightarrow{pop_3;pop_2} z$ mean that pop_3 followed by pop_2 on the stack of x results in the same stack as pop_3 followed by pop_2 on the stack of y, regardless of whether x and y have different control-states. Provided that the number of operations constituting the edge is bounded, this can be done without ϵ-closure.

3 Undecidability

3.1 Post's Correspondence Problem

All of the undecidability results go via a reduction from the Post Correspondence Problem [16], which is well known to be undecidable. Consider a finite-alphabet Σ with $|\Sigma| \geq 2$. An instance of the Post Correspondence Problem (PCP) consists of two finite sequences of strings over Σ: $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$. The question to be decided is whether there is a finite sequence $i_1 \ldots i_k$ consisting of integers $1 \leq i_j \leq m$ such that $u_{i_1}.u_{i_2}.\cdots.u_{i_k} = v_{i_1}.v_{i_2}.\cdots.v_{i_k}$.

▶ **Example 1.** Consider the following two sequences of strings over the alphabet $\{a, b, c\}$:

$$u_1 := ab \quad u_2 := cababcabb \quad u_3 := ca \qquad v_1 := ababc \quad v_2 := ab \quad v_3 := bca$$

Then the Post Correspondence Problem has answer 'yes' as witnessed by the solution 1123:

$$u_1.u_1.u_2.u_3 = ababcababcabbca = v_1.v_1.v_2.v_3$$

Given an instance of the PCP P we define a pushdown automaton $\mathcal{A}_1^P$ that pushes elements of Σ onto the stack together with indices indicating a partition into the u_i and v_i.

▶ **Definition 2.** The automaton $\mathcal{A}_1^P$ has stack alphabet: $\Sigma \uplus \{1_u, 2_u \ldots m_u\} \uplus \{1_v, 2_v \ldots m_v\}$ and behaves by non-deterministically choosing one of the following options:

- Push any member of Σ onto the stack.
- If the Σ symbols in the stack since the last symbol of the form i_u (or the bottom of the stack if there is no such symbol) form the word u_j, then it may push j_u onto the stack.
- If the Σ symbols in the stack since the last symbol of the form i_v (or the bottom of the stack if there is no such symbol) form the word v_j, then it may push j_v onto the stack.

P has a solution just in case $\mathcal{A}_1^P$ can generate a stack with 'matching' i_u and i_v subsequences.

▶ **Lemma 3.** *Let P be an instance of Post's Correspondence Problem. P has a solution just in case the automaton $\mathcal{A}_1^P$ can generate a stack s such that:*

- *$s_u = s_v$ where s_u is the subsequence of s consisting of elements of the form i_u and s_v of elements of the form i_v and equality is interpreted with respect to the indices i only.*
- *The top two elements of s form the set $\{i_u, i_v\}$ for some $1 \leq i \leq m$.*

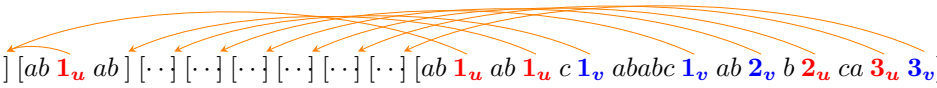
▶ **Example 4.** To continue the running example from Example 1, which we call P, the solution as represented by a stack of $\mathcal{A}_1^P$ is: $[\, ab\mathbf{1_u}ab\mathbf{1_u}c\mathbf{1_v}ababc\mathbf{1_v}ab\mathbf{2_v}b\mathbf{2_u}ca\mathbf{3_u}\mathbf{3_v}\,]$

3.2 Post's Correspondence Problem and 2-CPDA

Hague *et al.* [7] showed that the model-checking problem for MSO on 2-CPDA graphs is undecidable; indeed the 2-CPDA graph they exhibit witnesses the undecidability of transitive closure logic $\boldsymbol{FO(TC)}$. To introduce our basic technique, we first reprove the undecidability of $\boldsymbol{FO(TC)}$ on 2-CPDA graphs—in fact we get the undecidability of $\boldsymbol{FO(TC[\Delta_1])}$.

The 2-CPDA $\mathcal{A}_2^P$ is like $\mathcal{A}_1^P$ except it ensures each index (elements of the form i_u or i_v) has a pointer to a distinct 1-stack in the 2-stack. This enables first-order logic to 'ascertain corresponding positions' in two instances of a 1-stack by comparing the results of collapsing. More specifically, $\mathcal{A}_2^P$ behaves in the same way as $\mathcal{A}_1^P$ except that it performs $push_2; push_1^{j_u,2}$ or $push_2; push_1^{j_v,2}$ whenever $\mathcal{A}_1^P$ would have performed $push_1^{j_u}$ or $push_1^{j_v}$ for some $1 \leq j \leq m$.

▶ **Example 5.** To continue Example 1, the solution as represented by a stack of $\mathcal{A}_2^P$ is:



We adapt $\mathcal{A}_2^P$ further so that it may non-deterministically enter a distinguished control-state $\boldsymbol{guess}$ when it has formed a stack whose top two elements are those in a set $\{i_u, i_v\}$ for some $1 \leq i \leq m$. Lemma 3 then implies:

▶ **Lemma 6.** *Let P be an instance of Post's Correspondence Problem. P has a solution iff the automaton $\mathcal{A}_2^P$ can reach a configuration $(\boldsymbol{guess}, s)$ such that $s_u = s_v$, where s_u is the subsequence of $top_2(s)$ consisting of elements of the form i_u, and s_v of elements of the form i_v, and equality is interpreted with respect to the indices i only.*

Let us say that a stack s is a u-stack if its top element is of the form i_u and a v-stack if its top element is of the form i_v. Consider the compound operations $\boldsymbol{pop}_u(s)$ (resp. $\boldsymbol{pop}_v(s)$) that perform pop_1 on s, stopping only when the next element of the form i_u (resp. i_v) is found or else the top 1-stack is rendered empty if no such element exists. It is also useful to define $!u := v$ and $!v := u$. Note that a configuration $(\boldsymbol{guess}, s)$, where s is a w-stack for $w \in \{u, v\}$ (so that $pop_1(s)$ is a $!w$-stack), meets the criterion of Lemma 6 precisely when $top_1(\boldsymbol{pop}_w^k(s)) = i_w$ iff $top_1(\boldsymbol{pop}_{!w}^k(pop_1(s))) = i_{!w}$ for every $k \in \mathbb{N}$. This is because $\boldsymbol{pop}_u$ and $\boldsymbol{pop}_v$ step (backwards) through the sequences s_u and s_v.

Bearing this in mind, we extend $\mathcal{A}_2^P$ to an automaton $\mathcal{A}_{2+}^P$ that allows additional behaviour after reaching a $\boldsymbol{guess}$-configuration so that it does its best to verify the condition above. We will then use $\boldsymbol{FO(TC[\Delta_1])}$ to assert the constraints that the automaton is unable to enforce by itself. $\mathcal{A}_{2+}^P$ is defined to generate any sequence of configurations of the form:

$$(\boldsymbol{guess}, s) \xrightarrow{\boldsymbol{init}} (\boldsymbol{test}, t_1) \xrightarrow{\boldsymbol{next}} (\boldsymbol{test}, t_2) \xrightarrow{\boldsymbol{next}} \cdots \xrightarrow{\boldsymbol{next}} (\boldsymbol{test}, t_{k-1}) \xrightarrow{\boldsymbol{next}} (\boldsymbol{final}, t_k)$$

where the top three 1-stacks of each t_i include a 1-stack $\boldsymbol{u}(t_i)$ that is either a u-stack or empty and a 1-stack $\boldsymbol{v}(t_i)$ that is either a v-stack or empty. The automaton will abort if ever $top_1(\boldsymbol{u}(t_i)) = i_u$ but $top_1(\boldsymbol{v}(t_i)) = j_v$ with $i \neq j$ and will enter a control-state $\boldsymbol{final}$ iff $\boldsymbol{u}(t_k)$ and $\boldsymbol{v}(t_k)$ are both empty. Let us say that t_i is *w-dominant* for $w \in \{u, v\}$ if $\boldsymbol{w}(t_i) \sqsupset \boldsymbol{!w}(t_i)$, which occurs iff $\boldsymbol{w}(t_i)$ is below $\boldsymbol{!w}(t_i)$ in t_i. The third of the top three 1-stacks in t_i is the auxiliary *conditional stack* $\boldsymbol{cond}(t_i) = \boldsymbol{pop}_{!w}(\boldsymbol{!w}(t_i))$ where t_i is w-dominant. This will be of technical use in avoiding the need for ϵ-closure. Where s is a w-stack, the transition $\xrightarrow{\boldsymbol{init}}$ consists of just $push_2; pop_1; push_2; \boldsymbol{pop}_{!w}$, which sets up an initial $\boldsymbol{u}$, $\boldsymbol{v}$ and conditional stack, one of which will be the original $top_2(s)$.

The idea is that $\mathcal{A}^P_{2^+}$ can always enforce one of $\boldsymbol{u}(t_{i+1}) = \boldsymbol{pop}_u(\boldsymbol{u}(t_i))$ or $\boldsymbol{v}(t_{i+1}) = \boldsymbol{pop}_v(\boldsymbol{v}(t_i))$ but not both—the one that is not enforced in t_{i+1} must be guessed. It is easy to see that this is possible. A transition $(\boldsymbol{test}, t) \xrightarrow{\boldsymbol{next}} (\boldsymbol{test}, t')$ will employ a sequence of transitions of the form $pop_2; pop_2; pop_1^*; push_2; pop_1^*; push_2; pop_1^*$, enforcing $\boldsymbol{w}(t') = \boldsymbol{pop}_w(\boldsymbol{w}(t))$, guessing $\boldsymbol{!w}(t')$ but still enforcing $\boldsymbol{cond}(t') = \boldsymbol{pop}_{!w'}(\boldsymbol{!w'}(t'))$ when t is w-dominant and t' is w'-dominant. It may or may not be the case that $w = w'$ depending on the guess and the ordering of the i_u and i_v.

We now define a relation stronger than $\xrightarrow{\boldsymbol{next}}$ that $\mathcal{A}^P_{2^+}$ cannot impose by itself. We say that $c' = (\boldsymbol{test}, t')$ is a *successor* of $c = (\boldsymbol{test}, t)$, written $c' = c^+$, iff $c \xrightarrow{\boldsymbol{test}} c'$ and $\boldsymbol{!w}(t') = \boldsymbol{pop}_{!w}(\boldsymbol{w!}(t)) = \boldsymbol{cond}(t)$ where t is w-dominant. We then reformulate Lemma 6:

▶ **Lemma 7.** *Let $(\boldsymbol{guess}, s)$ be a reachable configuration of $\mathcal{A}^P_{2^+}$ for some instance P of Post's Correspondence Problem. Then s represents a solution to P in the sense of Lemma 6 if and only if there exists a chain of configurations $c_1, c_2, \ldots, c_k$ such that $c_{i+1} = c_i^+$ for $1 \leq i < k$, $(\boldsymbol{guess}, s) \xrightarrow{\boldsymbol{init}} c_1$ and c_k has control-state $\boldsymbol{final}$.*

Let us now observe that the relation $x = y^+$ is definable by a Δ_1 formula. Let w range over $\{u, v\}$ and consider configurations $(\boldsymbol{test}, t)$ and $(\boldsymbol{test}, t')$. Define $\boldsymbol{to_w}(t)$ to be $pop_2(t)$ or $(pop_2; pop_2)(t)$ as necessary to render $\boldsymbol{w}(t)$ the topmost 1-stack in t. Since the number of stack operations are bounded, we may safely view this as an edge in $\mathcal{G}(\mathcal{A}^P_{2^+})$ without resorting to ϵ-closure.

Recall that we have ensured that each instance of i_u or i_v in the representation of the postulated solution to P has a 2-link with a distinct target. It is also the case that the conditional stack of any t is the top-most 1-stack. This means that the equality $\boldsymbol{cond}(t) = \boldsymbol{w}(t')$ between *internal* 1*-stacks* holds iff the equality $collapse(t) = (\boldsymbol{to_w}; collapse)(t')$ holds, which may be viewed as an equality between *configurations* since t and t' are 2-stacks. Hence $x = y^+$ can be expressed by a Δ_1 formula $\psi_+(x, y) \wedge (x \xrightarrow{\boldsymbol{next}} y)$ where $\psi_+(x, y)$ asserts for each $w \in \{u, v\}$ that if y is $!w$-dominant, then:

$$\exists z.(x \xrightarrow{collapse} z \wedge y \xrightarrow{\boldsymbol{to_w}; collapse} z) \text{ equivalently } \forall z.(x \xrightarrow{collapse} z \rightarrow y \xrightarrow{\boldsymbol{to_w}; collapse} z)$$

With the aid of transitive closure, the existence of the $(\_)^+$-sequence in the condition in Lemma 7 can be asserted in Σ_1-$\boldsymbol{FO(TC[\Delta_1])}$, completing the reduction.

▶ **Lemma 8.** *There exists a Σ_1 sentence ϕ of $\boldsymbol{FO(TC[\Delta_1])}$ such that for all instances P of Post's Correspondence Problem we have: $\mathcal{G}(\mathcal{A}^P_{2^+}) \models \phi$ iff P has a solution.*

3.3 Undecidability for 3_2-CPDA and 4_2-CPDA

A 3_2-CPDA can record the chain of 2-stacks mentioned in Lemma 7 directly in its 3-stack—the members of the chain are piled on top of each other. This allows ϵ-closure in the graph and

universal quantification to replace transitive closure in the logic. As with $\mathcal{A}^P_{2^+}$ the key idea in the proof is to use first-order logic to assert the coherence of guesses that the automaton makes, ensuring that the 2-stacks making up the alleged $(\_)^+$-chain stored in the stack really do form a chain. The 3_2-CPDA $\mathcal{A}^P_{3_2}$ begins by behaving in the same way as $\mathcal{A}^P_{2^+}$, performing only 2-stack operations until just after it has performed an $\boldsymbol{init}$ transition. It then proceeds to perform $\boldsymbol{next}$ transitions, perfoming a $push_3$ operation after completing each one. It may stop and enter a distinguished control-state $\boldsymbol{guess}_{3_2}$ as soon as it has completed a $(push_3; \boldsymbol{next})$-transition that would have resulted in a $\boldsymbol{final}$-control state in $\mathcal{A}^P_{2^+}$. We thus have a solution to P iff $\mathcal{A}^P_{3_2}$ has a reachable configuration of the form:

$$(\boldsymbol{guess}_{3_2}, [s\ s_1\ s_2\ \cdots\ s_k]) \text{ such that } s_{i+1} = s_i^+ \text{ for every } 1 \leq i < k$$

Since the $s_i \xrightarrow{\boldsymbol{next}} s_{i+1}$ is guaranteed by the definition of $\mathcal{A}^P_{3_2}$, it suffices to assert that $\psi_+(s_i, s_{i+1})$ for every $1 \leq i \leq k$ (abusing notation by ignoring control-states). But observe how ψ_+ makes sense on 3-stacks provided that the 3-stacks only differ in their topmost 2-stacks. This is thus equivalent to $\psi_+([s\ s_1\ s_2\ \cdots\ s_{i-1}\ s_i\ s_i], [s\ s_1\ s_2\ \cdots\ s_{i-1}\ s_i\ s_{i+1}])$. Given ϵ-closure we are able to have a single edge labelled $\boldsymbol{toPre}$ going from a configuration of the form $(\boldsymbol{guess}_{3_2}, S)$ to any one of the form $(\boldsymbol{test}_{3_2}, S_i)$ where S_i is the prefix of S with its s_i on top for $2 \leq i \leq k$. Thus the following asserts correctness:

$$\exists x.(\boldsymbol{guess}_{3_2}(x) \ \wedge\ \forall y.(x \xrightarrow{\boldsymbol{toPre}} y \ \rightarrow\ \psi_+((pop_3; push_3)(y), y)))$$

▶ **Lemma 9.** *There exists a Σ_2 sentence ϕ of $\boldsymbol{FO}$ such that for all instances P of Post's Correspondence Problem we have: $\mathcal{G}^\epsilon(\mathcal{A}^P_{3_2}) \vDash \phi$ iff P has a solution.*

Observe that the sentence asserting the correctness of a chain suggested by $\mathcal{A}^P_{3_2}$ essentially says that every S_i generated by $\boldsymbol{toPre}$ from $(\boldsymbol{guess}_{3_2}, S)$ satisfies $\theta(S_i) = \theta'(S_i)$ for some particular sequences of operations θ and θ'. With the aid of an order-4_2 stack we are able to go from a $(\boldsymbol{guess}_{3_2}, [S])$ configuration to create both a configuration of the form $(\boldsymbol{test}_\theta, [S\ \theta(S_k)\ \theta(S_{k-1})\ \cdots\ \theta(S_2)])$ and following an alternative path a configuration of the form $(\boldsymbol{test}_{\theta'}, [S\ \theta'(S_k)\ \theta'(S_{k-1})\ \cdots\ \theta'(S_2)])$. In other words we create two 4_2-stacks containing in corresponding positions the 3_2-stacks that have to be compared to verify the chain. Given a 4_2-CPDA $\mathcal{A}^P_{4_2}$ implementing this, we can assert the existence of a correct chain with a Σ_1-sentence asserting the existence of a $\boldsymbol{test}_\theta$-configuration and a $\boldsymbol{test}_{\theta'}$-configuration both with the same stack. Whilst ϵ-closure is not required here, *the restriction to reachable configurations* is necessary for the control-states $\boldsymbol{test}_\theta$ and $\boldsymbol{test}_{\theta'}$ to be significant.

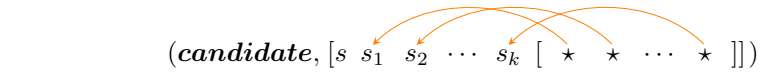
▶ **Lemma 10.** *There exists a Σ_1-sentence ϕ such that for every instance P of Post's Correspondence Problem we have $\mathcal{G}(\mathcal{A}^P_{4_2}) \vDash \phi$ iff P has a solution.*

3.4 The Non-Locality of 3_3-CPDA

Adapting $\mathcal{A}^P_{3_2}$ to become a 3_3-CPDA is straightforward—the role of 2-links, giving each element i_u and i_v a link with a distinct target, can easily be taken by 3-links; of course these distinct targets will be distinct 2-stacks rather than distinct 1-stacks. The challenge is that we also want to remove the need for ϵ-closure, although we still have to restrict the domain to configurations reachable from the origin. The idea is that 3-links are 'non-local' with respect to a 3-stack so that whilst with $\mathcal{A}^P_{3_2}$ it was necessary to use ϵ-closure in order to provide access to all elements of the alleged chain, this can instead, in some sense, be performed by a single *collapse* edge on a 3-link.

The 3_3-CPDA $\mathcal{A}^P_{3_3}$ begins by behaving in a similar way to $\mathcal{A}^P_{3_2}$ (using 3-links instead of 2-links). Recall a ***next***-transition in $\mathcal{A}^P_{2+}$ takes the form $pop_2; pop_2; pop_1^*; push_2; pop_1^*; push_2; pop_1^*$ (the lack of ϵ-closure is not a concern here as we will not need to refer to ***next***-transitions in the logic). By implication $\mathcal{A}^P_{3_2}$ will perform a ***next***-transition of the form: $push_3; pop_2; pop_2; pop_1^*; push_2; pop_1^*; push_2; pop_1^*$. We make $\mathcal{A}^P_{3_3}$ differ again from $\mathcal{A}^P_{3_2}$ by performing instead: $push_3; pop_2; pop_2; push_1^{\star,3}; push_2; pop_1^*; push_2; pop_1^*; push_2; pop_1^*$ for each ***next***-transition, where $\star$ is a new distinct stack symbol. After the ***next***-transitions are complete, and $\mathcal{A}^P_{3_2}$ would have moved into a $\boldsymbol{guess}_{3_2}$ control-state, we perform final $push_3; push_1^{\star,3}$ operations and move into a distinguished ***candidate*** control-state.

This has the effect of creating a configuration of the form:



which corresponds to an $\mathcal{A}^P_{3_2}$-configuration $(\boldsymbol{guess}_{3_2}, [s\ s_1\ s_2\ \cdots\ s_k\ ])$. From here we further extend $\mathcal{A}^P_{3_3}$ to be allowed to perform arbitarily long sequences of pop_2 operations from a ***candidate***-configuration with the option of ending in a distinguished control-state ***ready*** if it reaches a $\star$ symbol. Thus to assert the existence of a ***candidate***-configuration x such that $\chi(y)$ holds of all y that are prefixes of its stack ending in an s_i, we may say:

$$\exists x.\exists x'.\forall z.\forall y.(\boldsymbol{candidate}(x) \wedge x \xrightarrow{pop_3} x' \wedge ((\boldsymbol{ready}(z) \wedge z \xrightarrow{pop_3} x' \wedge z \xrightarrow{collapse} y) \rightarrow \chi(y)))$$

In particular we may take χ to be the assertion about prefixes used with $\mathcal{A}^P_{3_2}$ (noting that this does not require ϵ-closure). Again the restriction to reachable configurations is necessary to give ***candidate*** and ***ready*** their significance.

▶ **Lemma 11.** *Let P be an instance of Post's Correspondence Problem. Then there exists a Σ_2 sentence ϕ of **FO** such that: $\mathcal{G}(\mathcal{A}^P_{3_3}) \models \phi$ (note* no *ϵ-closure) if and only if P has a solution.*

As a consequence of all these reductions:

▶ **Theorem 12.** ***1.*** *For every $n \geq 4$ and $2 \leq m \leq n-2$ the Σ_1-**FO** model-checking problem for n_m-CPDA graphs (even without ϵ-closure, albeit restricted to configurations reachable from the origin) is undecidable.*

2. *For every $n \geq 3$ and $m \geq 3$ the Σ_2-**FO** model-checking problem for n_m-CPDA graphs (even w/o ϵ-closure, albeit restricted to configurations reachable from the origin) is undecidable.*

3. *For every $n \geq 3$ and $m \geq 2$ the Σ_2-**FO** model-checking problem for the ϵ-closures of n_m-CPDA graphs is undecidable.*

4 Σ_1 Decidability on n_n-CPDA

We reduce checking a Σ_1-sentence on the ϵ-closure of an n_n-CPDA graph to MSO model-checking on n-PDA, which is known to be decidable. The idea is to 'eliminate the n-links' and instead 'simulate' them in an n-PDA. The high level idea could be viewed as the graph analogue of Aehlig *et al.*'s link-elimination at order-2 for word generation [8]; however our technique for graphs is necessarily quite different. When generating a word one only needs to (non-deterministically) simulate a single run at a time, whilst for graphs one sometimes needs to verify the *non*-existence of edges which in some sense requires checking *all possible* runs.

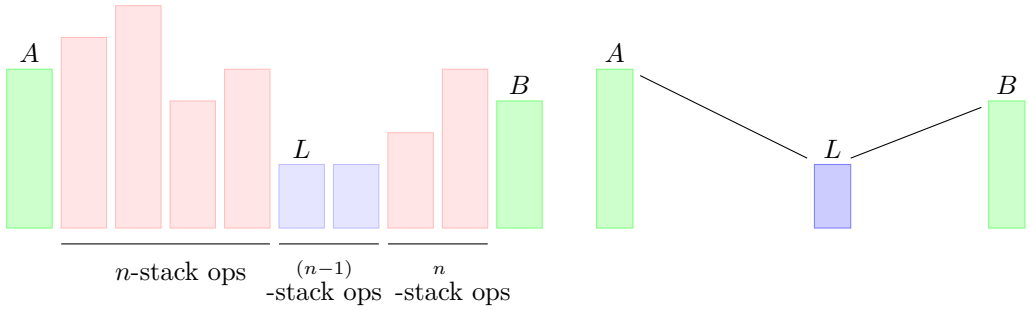


Figure 1 The anatomy of a run and eliminating the destruction of $(n-1)$-stacks.

Another issue is that for graphs links not only serve an operational purpose but additionally distinguish otherwise identical stacks. Due to space constraints we will not detail the solution to this challenge as it is largely technical. But the basic idea is that every atomic element (0-stack) and every constituent k-stack is assigned one of three colours $c_<$, $c_=$ and $c_>$ which specifies whether the highest target of an n-link that it contains is below, the same as or above the highest target of an n-link contained in a k-stack below it in the $(k+1)$-stack. The colouring of every prefix of a stack is sufficient to distinguish stacks even when links are deleted. It is also possible for an n_n-CPDA to keep track of sufficient information to dynamically maintain colour annotations.

We instead focus on the more interesting question of accounting for the operational aspect of links after they have been eliminated—that is how we express what a run involving *collapse* 'would have done' in the simulating n-PDA. Again due to space constraints we only sketch the main ideas. Our strategy is illustrated in Figure 1. Consider an a-labelled edge in the ϵ-closure of the configuration graph of an n_n-CPDA $\mathcal{A}$ between a configuration with stack A and a configuration with stack B, witnessed by an ϵ^*a-labelled run. Such a run can be divided into two parts with a configuration bearing stack L at the boundary. Suppose that no n-stack containing fewer than m $(n-1)$-stacks is used, then L is deemed to be the first configuration in the run with a stack containing m $(n-1)$-stacks.

We describe the ϵ^*-labelled first part of the run from A to L as an ϵ^*-*fall* and the ϵ^*a-labelled component from L to B as an ϵ^*a-*climb*. These two components are simulated by an n-PDA using two different methods. First consider the climb. Due to L being the 'lowest stack' in the run, it is possible to construct the stack of B from L without destroying any $(n-1)$-stacks and in particular without having to perform *collapse*. Of course the original n-CPDA might have a more complex run between L and B that does need to use *collapse*, but we show that it is possible to adjust the automaton so that it 'can smooth out' its run during the ϵ^*a-climb and avoid destroying any $(n-1)$-stacks. This means that climbs can be implemented *directly* by an n-PDA.

For the fall we exploit the fact that the stack of L is a prefix of the stack of A. The idea is that each $(n-1)$-stack t in an n-stack s will be annotated with information about the climbs that could be performed from the prefix of s ending in t. We then adjust the automaton further so that it is always aware of those annotations to which it has an ϵ^*-fall. This means that it can exploit the information in the annotations *indirectly* without actually having to 'physically' perform the ϵ^*-fall. Again this means that physical use of the *collapse* operation can be avoided.

In order to keep the annotations concerning climbs finite, it is necessary to fix in advance

a finite number of stacks to which one might eventually want to climb. These stacks will be possible witnesses to our Σ_1-sentence and so there will be one for each variable therein; the requirement to fix these is the reason why the proof does not generalise to more complex sentences. It is also impossible for an n-PDA to correctly provide such annotations itself and so, as with the undecidability proofs, it is necessary for it to guess these annotations and then use MSO to verify correctness externally.

So let us fix an n_n-CPDA $\mathcal{A}$ and a Σ_1-sentence $\exists x_1 \exists x_2 \cdots \exists x_k . \chi(x_1, x_2, \ldots, x_k)$ to be interpreted over $\mathcal{G}^{\epsilon}(\mathcal{A})$ where $\chi(x_1, x_2, \ldots, x_k)$ is quantifier free.

4.1 Simulating the Climb

An $\epsilon^* a$-climb is a run of the form: $(q, [s_1\ s_2\ \cdots\ s_k\ t]) \xrightarrow{\epsilon^* a} (q', [s_1\ s_2\ \cdots\ s_k\ t'\ t_1\ t_2\ \cdots\ t_m])$ that *never descends below* t so that for all stacks s occurring in the run, $[s_1\ s_2\ \cdots\ s_k] \sqsubset s$. Our first step is to construct a modified *'monotonic'* automaton $\mathcal{A}^{\uparrow}$ that can climb from the configuration transition to the last *without ever destroying any* $(n-1)$ *stacks* in the process.

We are able to construct $\mathcal{A}^{\uparrow}$ using *logical reflection* [2]. Given μ-calculus sentences $\phi_1, \ldots, \phi_k$, logical reflection allows a CPDA to be augmented so that it 'knows' (by reference to its control-state and the top element of the stack) whether a particular ϕ_i holds in its graph at its current configuration. In particular we can take the ϕ_i to be assertions of the form: 'If I were to to mark the current top_n $(n-1)$-stack and then perform a $push_n$ into control-state q, then I would be able to perform ϵ-transitions and end back at the marker in control-state q''. Given an automaton that is aware of the truth of such assertions, it can be adapted so that instead of actually performing these ϵ-transitions, which grow the stack before returning to the starting point, it can simply transition from control-state q to control-state q' without touching the stack. In this way it is able to carry out the climb without creating an $(n-1)$-stack only to destroy it later on, which is exactly what we want.

The CPDA produced by an application of logical reflection generates a graph isomorphic to the original in a particularly strong way. That is if a CPDA $\mathcal{B}'$ is produced from a CPDA $\mathcal{B}$ by an application of logical reflection, then there is an isomorphism $L : \mathcal{G}(\mathcal{B}) \cong (\mathcal{B}')$ that can be defined as a map on configurations of the form $L(q, s) = (q, L(s))$ where the control-state is preserved and L preserves the structure of the stack (albeit adding information to the atomic elements therein). Even though the evaluation of a μ-calculus formula will be a function of both q and $top_1(L(s))$ it is $top_1(L(s))$ that contains all of the additional information provided by logical reflection. (In actual fact this is a slight abuse of notation as $\mathcal{B}'$ will need to store information concerning the stack s in its control-state as well. However, since this information depends completely on the stack and not the control-state, it is safe for our purposes to suppress it notationally and assume that it is 'absorbed' into the $L(s)$ component). This means that when $\mathcal{A}^{\uparrow}$ transitions from control-state q to control-state q' without touching the stack, the information that the stack contains remains correct.

Using a similar technique we also allow $\mathcal{A}^{\uparrow}$ to construct all of its reachable configurations from its initial state without destroying $(n-1)$-stacks via transitions given a fresh label r.

4.2 Meta-Annotations—Towards simulating the Fall

Let Σ be the set of non-ϵ edge labels of $\mathcal{A}$ and let Q be its set of control-states, both of which are shared by $\mathcal{A}^{\uparrow}$. A *k-meta-annotation* (where k is the number of variables in the formula) is a $k.|\Sigma|$-tuple $M := ((Q_1^a)_{a\in\Sigma}, (Q_2^a)_{a\in\Sigma}, \ldots, (Q_k^a)_{a\in\Sigma})$ where $Q_i^a \subseteq Q$ for every i and a with $1 \leq i \leq k$ and $a \in \Sigma$. We further adapt $\mathcal{A}^{\uparrow}$ to non-deterministically choose some

meta-annotation M to push onto the stack before performing any $push_n$ operation. Hence every $(n-1)$-stack in any reachable configuration is decorated with a meta-annotation.

The '*meaning*' of meta-annotations is given with respect to a k-tuple of $\mathcal{A}^{\uparrow}$ configurations $c_1 = (q_1, s_1), c_2 = (q_2, s_2), \ldots, c_k = (q_k, s_k)$. Suppose that $t \sqsubseteq s_i$ is a prefix of one of the stacks of these configurations, then a 'correct' meta-annotation on top of t is one such that its set Q^a_j contains a control-state p iff there is an $\epsilon^* a$-climb from (p, t) to (q_j, s_j). Note in particular that if $pop_n(t) \not\sqsubseteq s_j$ then such a climb would be impossible and so $Q^a_j = \emptyset$.

If the meta-annotations contained in the stacks of $c_1, c_2, \ldots, c_k$ are all correct in this sense, then we say that this k-tuple of configurations is *consistent*. Consistency is not something that can be guaranteed by the automaton itself, but will be asserted using MSO.

We now consider how to handle ϵ^*-falls. Formally an ϵ^*-fall is a run of the form: $(q, [s_1\ s_2\ \cdots\ s_k]) \xrightarrow{\epsilon^*} (q', [s_1\ s_2\ \cdots\ s_{k'}])$ where $k' < k$ and $[s_1\ s_2\ \cdots\ s_{k'}]$ is a prefix of every stack occurring in the run.

We make one final adjustment to $\mathcal{A}^{\uparrow}$ by again applying logical reflection, this time with μ-calculus assertions of the form $\psi_{M,q}$ for each possible meta-annotation M and control-state q. The sentence $\psi_{M,q}$ says: 'From my current configuration I could behave as $\mathcal{A}$, without deploying any new meta-annotations, and eventually reach a stack with the meta-annotation M on top in control-state q'. Since no new meta-annotations are deployed, $\psi_{M,q}$ effectively says: 'I have an ϵ^*-fall from my current configuration to a configuration with control-state q and M on top of the stack'. Logical reflection gives the CPDA graph a unary predicate $\boldsymbol{M_q}^{\downarrow}$ that holds whenever $\psi_{M,q}$ would hold.

The *n-PDA* $\mathcal{A}^{\uparrow\downarrow}$ is formed from $\mathcal{A}^{\uparrow}$, adjusted as above, by removing all of the transitions performing a pop_n or *collapse* operation. Note that the $\boldsymbol{M_q}^{\downarrow}$ predicates depend only on the top_1 element of the stack and current control-state—these thus remain in $\mathcal{A}^{\uparrow\downarrow}$. Moreover since $\mathcal{A}^{\uparrow}$ was adapted to perform all $\epsilon^* a$-climbs without performing pop_n or *collapse*, then despite the deletion of edges, if $\mathcal{A}^{\uparrow}$ was able to perform an $\epsilon^* a$-climb from a configuration c to a configuration c', then $\mathcal{A}^{\uparrow\downarrow}$ can also perform such a climb. Hence consistency is preserved. The reachable configurations will also be unaffected by this deletion due to the r-transitions.

4.3 Σ_1-Model Checking

Any run of the form $(q, s) \xrightarrow{\epsilon^* a} (q', s')$, which witnesses an a-labelled edge in the ϵ-closure, is either a climb itself or else can be decomposed into $(q, s) \xrightarrow{\epsilon^*} (p, t) \xrightarrow{\epsilon^* a} (q', s')$ where the component from (q, s) to (p, t) is an ϵ^*-fall and the component from (p, t) to (q', s') is an $\epsilon^* a$-climb. We can thus assert the existence of an a-edge in $\mathcal{G}^{\epsilon}(\mathcal{A})$ by asserting in $\mathcal{G}(\mathcal{A})$ the existence of either a climb or else such a fall followed by a climb.

Given the quantifier-free first-order formula $\chi(x_1, x_2, \ldots, x_k)$, construct $\chi'(x_1, x_2, \ldots, x_k)$ by replacing every occurence of an atomic formula $x_i \xrightarrow{a} x_j$ with an MSO formula:

$$x_i \xrightarrow{\epsilon^* a} x_j \ \vee \bigvee_{M=((Q^a_1)_{a\in\Sigma},(Q^a_2)_{a\in\Sigma},\ldots,(Q^a_k)_{a\in\Sigma}),\ \ p\in Q^a_j} \boldsymbol{M_p}^{\downarrow}(x_i)$$

where the reachability assertion $x_i \xrightarrow{\epsilon^* a} x_j$ is MSO definable. Observe that in the n-PDA $\mathcal{A}^{\uparrow\downarrow}$ the relation $\xrightarrow{\epsilon^* a}$ asserts precisely the existence of an $\epsilon^* a$-climb and so if $c'_1, c'_2, \ldots, c'_k$ are *consistent* configurations of $\mathcal{A}^{\uparrow\downarrow}$ corresponding (via the isomorphism) to configurations $c_1, c_2, \ldots, c_k$ of $\mathcal{A}$, then $\mathcal{G}(\mathcal{A}^{\uparrow\downarrow}) \vDash \chi'(c'_1, c'_2, \ldots, c'_k)$ iff $\mathcal{G}^{\epsilon}(\mathcal{A}) \vDash \chi(c_1, c_2, \ldots, c_k)$. But note further that since for every k-tuple of configurations of $\mathcal{A}$ there exists *precisely one*

corresponding *consistent* k-tuple of configurations of $\mathcal{A}^{\uparrow\downarrow}$ we have:

$$\mathcal{G}(\mathcal{A}^{\uparrow\downarrow}) \models \exists x_1 x_2 \cdots x_k.(\textbf{\textit{con}}(x_1, x_2, \ldots, x_k) \wedge \chi'(x_1, x_2, \ldots, x_k)) \quad \text{iff } \mathcal{G}^{\epsilon}(\mathcal{A}) \models \exists x_1 x_2 \cdots x_k.\chi(x_1, x_2, \ldots, x_k)$$

where $\textbf{\textit{con}}(x_1, x_2, \ldots, x_k)$ asserts consistency. Note that $\textbf{\textit{con}}(x_1, x_2, \ldots, x_k)$ is also MSO definable in $\mathcal{G}(\mathcal{A}^{\uparrow\downarrow})$; one can quantify over prefixes of a stack by iterated pop_n, and reachability in $\mathcal{G}(\mathcal{A}^{\uparrow\downarrow})$ captures precisely the climbs in $\mathcal{A}$. This completes the reduction.

5 Further Directions

Whilst it is very natural to restrict to configurations reachable from the origin of the graph, it would also be interesting to investigate removing this restriction. For collapsible stacks there would be two versions of this problem: allowing arbitrary stack contents in configurations or restricting to stacks that could be constructed from the empty stack using stack operations. We conjecture that the former is undecidable but that the latter is decidable.

Acknowledgement We thank the anonymous reviewers for helpful comments.

References

1. C.H Broadbent. *On Collapsible Pushdown Automata, their Graphs and the Power of Links.* PhD thesis, 2011.
2. C.H. Broadbent, A. Carayol, C.-H.L. Ong, and O. Serre. Recursion Schemes and Logical Reflection. In *LICS*, 2010.
3. T. Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *ICALP*, 2003.
4. A. Carayol. Regular Sets of Higher-Order Pushdown Stacks. In *MFCS*, 2005.
5. A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *FSTTCS*, 2003.
6. D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theoretical Computer Science*, 290(1):79–115, 2003.
7. M. Hague, A.S. Murawski, C.-H. L. Ong, and O. Serre. Collapsible Pushdown Automata and Recursion Schemes. In *LICS*, 2008.
8. K. Aehlig, J. G. de Miranda, and C.-H. L. Ong. Safety is not a restriction at level 2 for string languages. In *FoSSaCS*, 2005.
9. A. Kartzow. Collapsible Pushdown Graphs of Level 2 are Tree-Automatic. In *STACS*, 2010.
10. T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-Order Pushdown Trees are Easy. In *FoSSaCS*, 2002.
11. T. Knapik, D. Niwinski, P. Urzyczyn, and I. Walukiewicz. Unsafe Grammars and Panic Automata. In *ICALP*, 2005.
12. N. Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *POPL*, 2009.
13. C.-H.L. Ong and S.J. Ramsay. Verifying Higher-Order Functional Programs with Pattern-Matching Algebraic Data Types. In *POPL*, 2011.
14. P. Parys. Collapse Operation Cannot Be Simulated Even By Using Higher Levels (Unpublished). 2011.
15. P. Parys. Collapse Operation Increases Expressive Power of Deterministic Higher Order Pushdown. In *STACS*, 2011.
16. E. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52(4):264–268, 1946.