Collapsible Pushdown Automata and Recursion Schemes

Matthew Hague, Andrzej Murawski, Luke Ong, Olivier Serre

Collapsible Pushdown Automata and Recursion Schemes

M. Hague* A. S. Murawski† C.-H. L. Ong‡ O. Serre§

(Preliminary version, 16 January 2007)

Abstract

Collapsible pushdown automata (CPDA) are a new kind of higher-order pushdown automata in which every symbol in the stack has a link to a stack situated somewhere below it. In addition to the higher-order stack operations $push_i$ and pop_i, CPDA have an important operation called $collapse$, whose effect is to "collapse" a stack s to the prefix as indicated by the link from the top_1-symbol of s. Our first result is that CPDA are equi-expressive with *recursion schemes* as generators of node-labelled ranked trees. In one direction, we give a simple algorithm that transforms an order-n CPDA to an order-n recursion scheme that generates the same tree, uniformly for all $n \geq 0$. In the other direction, using ideas from game semantics, we give an effective transformation of order-n recursion schemes (not assumed to be *homogeneously typed*, and hence not necessarily *safe*) to order-n CPDA that compute *traversals* over a certain finite graph determined by the scheme, and hence paths in the tree generated by the scheme. Our equi-expressivity result is the first such automata-theoretic characterization of (general) recursion schemes.

An important consequence of the equi-expressivity result is that it allows us to translate decision problems on trees generated by recursion schemes to equivalent problems on CPDA and *vice versa*. For example, since the Modal Mu-Calculus Model-Checking Problem for trees generated by order-n recursion schemes is n-EXPTIME complete, we show that it follows that the same decidability result holds for the problem of solving a parity game played on an order-n *collapsible pushdown graph* i.e. the configuration graph of a corresponding (order-n) collapsible pushdown system; the latter subsumes several well-known results about the solvability of games over (higher-order) pushdown graphs by (respectively) Walukiewicz, Cachat, and Knapik *et al*. Moreover our approach yields techniques that are radically different from standard methods for solving model-checking problems on infinite graphs generated by finite machines. This transfer of techniques goes both ways. Another innovation of our work is a self-contained proof of the solvability of parity games on collapsible pushdown graphs by generalizing *standard* techniques in the field. By appealing to our equi-expressivity result, we obtain a new proof of the decidability (and optimal complexity) of the Modal Mu-Calculus Model-Checking Problem of trees generated by recursion schemes.

In contrast to higher-order pushdown graphs, we show that Monadic Second-Order (MSO) theories of collapsible pushdown graphs are undecidable. Hence collapsible pushdown graphs are, to our knowledge, the first example of a natural class of finitely-presentable graphs that have undecidable MSO theories while enjoying decidable modal mu-calculus theories.

Keywords: Higher-order (collapsible) pushdown automata, higher-order recursion schemes, ranked and ordered trees, solution of parity games over configuration graphs.

*Oxford University Computing Laboratory (OUCL). `web.comlab.ox.ac.uk/oucl/work/matthew.hague/`
†OUCL. `web.comlab.ox.ac.uk/oucl/work/andrzej.murawski/`
‡OUCL. `users.comlab.ox.ac.uk/luke.ong/`
§LIAFA (CNRS and Université Paris VII). `www.liafa.jussieu.fr/~serre/`

Contents

1 Introduction

Higher-order pushdown automata (PDA) were first introduced by Maslov [17, 18] as accepting devices for word languages. As n varies over the natural numbers, the languages accepted by order-n pushdown automata form an infinite hierarchy. In *op. cit.* Maslov gave an equivalent definition of the hierarchy in terms of *higher-order indexed grammars*. Yet another characterization of Maslow's hierarchy was given by Damm and Goerdt [8, 9]: they studied *higher-order recursion schemes* that satisfy the constraint of *derived types*, and showed that the word languages generated by order-n such schemes coincide with those accepted by order-n PDA. Maslow's hierarchy offers an attractive classification of the semi-decidable languages: orders 0, 1 and 2 are respectively the regular, context-free and indexed languages, though little is known about languages at higher orders.

Higher-order PDA as a generating device for (possibly infinite) labelled ranked trees was first studied by Knapik, Niwiński and Urzyczyn [15]. As in the case of word languages, an infinite hierarchy of trees is defined, according to the order of the generating PDA; lower orders of the hierarchy are well-known classes of trees: orders 0, 1 and 2 are respectively the regular [22], algebraic [7] and hyperalgebraic trees [14]. Knapik *et al.* considered another method of generating such trees, namely, by higher-order (deterministic) recursion schemes that satisfy the constraint of *safety*. A major result in that work is the equi-expressivity of the two methods as tree generators. Open since the early 1980s, a question of fundamental importance in higher-type recursion is to find the class of automata that characterizes the expressivity of higher-order recursion schemes[1]. The results of Damm and Goerdt, and of Knapik *et al.* may be viewed as attempts to answer the question; they have both had to impose syntactic constraints (of derived types and safety respectively[2], which seem awkward and rather unnatural) on recursion schemes in order to establish their results. An exact correspondence with (general) recursion schemes has never been proved before.

A partial answer was recently obtained by Knapik, Niwiński, Urzyczyn and Walukiewicz. In an ICALP'05 paper [16], they proved that order-2 homogeneously-typed (but not necessarily safe) recursion schemes are equi-expressive with a variant class of order-2 pushdown automata called *panic automata*. In this paper, we give a complete answer to the question. We introduce a new kind of higher-order pushdown automata (which generalizes *pushdown automata with links* [2], or equivalently panic automata, to all finite orders), called *collapsible pushdown automata* (CPDA), in which every symbol in the stack has a link to a (necessarily lower-ordered) stack situated somewhere below it. In addition to the higher-order stack operations $push_i$ and pop_i, CPDA have an important operation called $collapse$, whose effect is to "collapse" a stack s to the prefix as indicated by the link from the top_1-symbol of s. The main result of this paper (Theorem 5 and Theorem 9) is that for every $n \geq 0$, order-n recursion schemes and order-n CPDA are equi-expressive as generators of trees.

Our equi-expressivity result has a number of far-reaching consequences. It allows us to translate decision problems on trees generated by recursion schemes to equivalent problems on CPDA and *vice versa*. Chief among them is the Modal Mu-Calculus Model-Checking Problem (equivalently Alternating Parity Tree Automaton Acceptance Problem, or equivalently Monadic Second-Order (MSO) Model-Checking Problem). We observe that all these problems reduce to the problem of solving a parity game played on a *collapsible pushdown graph* i.e. the configuration graph of a corresponding collapsible pushdown system (CPDS). Recently one of us has shown [20] that the above decision problems for trees generated by order-n recursion schemes are n-EXPTIME complete. Thanks to our Equi-Expressivity Theorems, it follows that the same (n-EXPTIME complete) decidability result holds for the corresponding CPDS problems, which subsumes many known results [26, 3, 16]. Moreover our approach yields techniques that are radically different from standard methods for solving model-checking problems on infinite graphs generated by finite machines. We stress that this transfer

[1] Higher-order recursion schemes are essentially simply-typed lambda calculus with general recursion and uninterpreted first-order function symbols

[2] As constraints on recursion schemes, *derived types* and *safety* are actually equivalent; see [10] for a proof.

of techniques goes both ways. Indeed another innovation of our work is a self-contained (and without recourse to game semantics) proof of the solvability of parity games on collapsible pushdown graphs by generalizing *standard* techniques in the field. By appealing to our Equi-Expressivity Theorems, we obtain new proofs for the decidability (and optimal complexity) of model-checking problems of trees generated by recursion schemes as studied in [20].

In contrast to higher-order pushdown graphs (which do have decidable MSO theories [5]), we show that MSO theories of collapsible pushdown graphs are undecidable. Hence collapsible pushdown graphs are, to our knowledge, the first example of a natural class of finitely-presentable graphs that have undecidable MSO theories while enjoying decidable modal mu-calculus theories.

2 Collapsible pushdown automata (CPDA)

We first introduce *(higher-order) collapsible pushdown automata*, assuming that the reader is familiar with the notion of higher-order pushdown automata as presented by Knapik *et al.* in their FOSSACS 2002 paper [15]. An order-n CPDA, or n-CPDA for short, is just an order-n pushdown automata (PDA) in which every non-$\bot$ symbol in the order-n stack has a *link* to a (necessarily lower-ordered) stack situated below it.

2.1 Stacks with links

Fix a stack alphabet Γ and a distinguished *bottom-of-stack symbol* $\bot \in \Gamma$. An ***order-0 stack*** (or simply *0-stack*) is just a stack symbol. An ***order-***$(n+1)$ ***stack*** (or simply $(n+1)$*-stack*) s is a non-null sequence (written $[\, s_1 \cdots s_l \,]$) of n-stacks such that every non-$\bot$ Γ-symbol a that occurs in s has a link to a stack of some order k (say, where $0 \leq k \leq n$) situated below it in s; we call the link a $(k+1)$*-link*. The *order* of a stack s is written $ord(s)$.

As usual, the bottom-of-stack symbol $\bot$ cannot be popped from or pushed onto a stack. Thus we require an *order-1 stack* to be a non-null sequence $[\, a_1 \cdots a_l \,]$ of elements of Γ such that for all $1 \leq i \leq l$, $a_i = \bot$ iff $i = 1$. We define $\bot_k$, the ***empty*** k***-stack***, as follows: $\bot_0 = \bot$ and $\bot_{k+1} = [\, \bot_k \,]$.

We first define the operations pop_i and top_i with $i \geq 1$: $top_i\, s$ returns the top $(i-1)$-stack of s, and $pop_i\, s$ returns s with its top $(i-1)$-stack removed. Precisely let $s = [\, s_1 \cdots s_{l+1} \,]$ be a stack with $1 \leq i \leq ord(s)$:

$$top_i \underbrace{[\, s_1 \cdots s_{l+1} \,]}_{s} = \begin{cases} s_{l+1} & \text{if } i = ord(s) \\ top_i\, s_{l+1} & \text{if } i < ord(s) \end{cases}$$

$$pop_i \underbrace{[\, s_1 \cdots s_{l+1} \,]}_{s} = \begin{cases} [\, s_1 \cdots s_l \,] & \text{if } i = ord(s) \text{ and } l \geq 1 \\ [\, s_1 \cdots s_l\, pop_i s_{l+1} \,] & \text{if } i < ord(s) \end{cases}$$

By abuse of notation, we set $top_{ord(s)+1}\, s = s$. Note that $pop_i\, s$ is undefined if $top_{i+1}\, s$ is a one-element i-stack. For example $pop_2\, [\, [\, \bot\, a\, b \,] \,]$ is undefined.

There are two kinds of $push$ operations. First the first-order $push$. Let a be a non-$\bot$ stack symbol and $1 \leq k \leq ord(s)$, we define a new stack operation $push_1^{a,k}$ that, when applied to s, first attaches a link from a to the $(k-1)$-stack *immediately* below the top $(k-1)$-stack of s, then pushes a (with its link) onto the top

1-stack of s. Formally for $1 \leq k \leq ord(s)$ and $a \in (\Gamma \setminus \{ \bot \})$, we define

$$push_1^{a,k} \underbrace{[s_1 \cdots s_{l+1}]}_{s} = \begin{cases} [s_1 \cdots s_l \, push_1^{a,k} \, s_{l+1}] & \text{if } k < ord(s) \\ [s_1 \cdots s_l \, s_{l+1} \, a^\dagger] & \text{if } k = ord(s) = 1 \\ [s_1 \cdots s_l \, push_1^{\widehat{a}} \, s_{l+1}] & \text{if } k = ord(s) \geq 2 \text{ and } l \geq 1 \end{cases}$$

where

- $a^\dagger$ denotes the symbol a with a link to the 0-stack s_{l+1}
- $\widehat{a}$ denotes the symbol a with a link to the $(k-1)$-stack s_l; and we define

$$push_1^{\widehat{a}} \underbrace{[t_1 \cdots t_{r+1}]}_{t} = \begin{cases} [t_1 \cdots t_r \, push_1^{\widehat{a}} \, t_{r+1}] & \text{if } ord(t) > 1 \\ [t_1 \cdots t_{r+1} \, \widehat{a}] & \text{otherwise i.e. } ord(t) = 1 \end{cases}$$

The higher-order $push_j$, where $j \geq 2$, simply duplicates the top $(j-1)$-stack of s. Precisely, let $s = [s_1 \cdots s_{l+1}]$ be a stack with $2 \leq j \leq ord(s)$:

$$push_j \underbrace{[s_1 \cdots s_{l+1}]}_{s} = \begin{cases} [s_1 \cdots s_{l+1} \, s_{l+1}] & \text{if } j = ord(s) \\ [s_1 \cdots s_l \, push_j s_{l+1}] & \text{if } j < ord(s) \end{cases}$$

Note that in case $j = ord(s)$ above, the link structure of s_{l+1} is preserved by the copy that is pushed on top by $push_j$.

Finally there is an important operation called $collapse$. We say that the n-stack s_0 is a ***prefix*** of an n-stack s, written $s_0 \leq s$, just in case s_0 can be obtained from s by a sequence of (possibly higher-order) pop operations. Take an n-stack s whose top_1-element has a link to (a particular copy of) k-stack u somewhere in s, such that $top_{k+1} \, s_0$ is (that copy of) u for some $s_0 \leq s$. Then $collapse \; s$ is defined to be s_0.

2.2 A formal definition of CPDA stack operations

One way to give a formal semantics of the stack operations is to work with appropriate numeric representations of the links. Knapik *et al.* have shown how this can be done in the order-2 case – called panic automata – in [16]. Here we introduce a different encoding of stacks with links that works for all orders. The idea is simple: take an order-n stack s and suppose there is a link from (a particular occurrence of) a symbol a in s to some $(j-1)$-stack. First denote by s' the unique prefix of s whose top_1-element is the occurrence of a. Then there exists a unique k such that $collapse \; s' = \underbrace{pop_j \, ; \cdots ; \, pop_j}_{k} \; s$. In s, we represent the occurrence of a with its link as $a^{(j,k)}$, where the superscipt (j,k) is a shorthand for the iterated stack operation $pop_j^k = \underbrace{pop_j \, ; \cdots ; \, pop_j}_{k}$.

In the formal definition, a *symbol-with-link* of an order-n CPDA is written $a^{(j,k)}$, where $a \in \Gamma$, $1 \leq j \leq n$ and $k \geq 1$, such that[3] if $j = 1$ then $k = 1$. Further, purely for convenience, we require that if $a = \bot$ then $j = k = 1$.

[3] Thus 1-links are invariant – they always point to the preceding symbol and no stack operation will change that.

The set Op_n of order-n CPDA *stack operations* comprises four types of operations:

1. pop_k for each $1 \leq k \leq n$
2. $push_j$ for each $2 \leq j \leq n$
3. $push_1^{a,k}$ for each $1 \leq k \leq n$ and each $a \in (\Gamma \setminus \{\perp\})$, and
4. $collapse$.

We define them in terms of the standard stack operations of an order-n PDA (namely, pop_j's, top_j's and $push_1^-$'s) as follows, where $1 \leq i \leq ord(s)$ and $2 \leq j \leq ord(s)$:

$$\begin{aligned} push_1^{b,i}\, s &= push_1^{b^{(i,1)}}\, s \\ collapse\, s &= pop_e^f\, s \quad \text{where } top_1\, s = a^{(e,f)} \\ push_j\, \underbrace{[\, s_1 \cdots s_{l+1}\,]}_{s} &= \begin{cases} [\, s_1 \cdots s_{l+1}\, s_{l+1}^{\langle j \rangle}\,] & \text{if } j = ord(s) \\ [\, s_1 \cdots s_l\, push_j s_{l+1}\,] & \text{if } j < ord(s) \end{cases} \end{aligned}$$

where $\Theta^{\langle j \rangle}$ is the operation of replacing every superscript occurring in Θ of the form (j, k_j) (for some k_j) by $(j, k_j + 1)$. The meaning of pop_j is the standard one.

Example 2.1 Take the 3-stack $s = [[[\perp a]]\ [[\perp][\perp a]]]$. (To save writing, we omit the superscript $(1,1)$.) We have

$$\begin{aligned} push_1^{b,2}\, s &= [[[\perp a]]\ [[\perp][\perp a\, b^{(2,1)}]]] \\ push_1^{c,3}(push_1^{b,2}\, s) &= [[[\perp a]]\ [[\perp][\perp a\, b^{(2,1)}\, c^{(3,1)}]]] \\ push_2(push_1^{c,3}(push_1^{b,2}\, s)) &= [[[\perp a]]\ [[\perp][\perp a\, b^{(2,1)}\, c^{(3,1)}][\perp a\, b^{(2,2)}\, c^{(3,1)}]]] \\ push_3(push_1^{c,3}(push_1^{b,2}\, s)) &= [[[\perp a]]\ [[\perp][\perp a\, b^{(2,1)}\, c^{(3,1)}]]\ [[\perp][\perp a\, b^{(2,1)}\, c^{(3,2)}]]] \end{aligned}$$

and we have

$$collapse(push_2(push_1^{c,3}(push_1^{b,2}\, s))) = collapse(push_3(push_1^{c,3}(push_1^{b,2}\, s))) = [[[\perp a]]]$$

2.3 Tree-generating CPDA

Collapsible pushdown automata are a generalization (to all finite orders) of *pushdown automata with links* [2, 1], which are essentially the same as *panic automata* [16]. There are various versions of CPDA, depending on whether the automaton is used as an accepting or generating device (for word language, or a labelled tree or graph, in which case there is an accompanying input alphabet of an appropriate kind) or to describe a computational process. Here we consider the version for generating Σ-labelled trees, for a given ranked alphabet Σ. (When it is used to generate graphs – see Section 6, we shall refer to the device as collapsible pushdown *systems*.)

Definition 2.2 A tree generating ***order-n collapsible pushdown automaton*** (n-CPDA) is given by a 5-tuple $\mathcal{A} = \langle\, \Sigma, \Gamma, Q, \delta, q_0 \,\rangle$ where Σ is a ranked alphabet, Γ is a stack alphabet, Q is a finite set of control states, q_0 is the initial state, and δ is the transition function

$$\delta \;:\; Q \times \Gamma \;\longrightarrow\; (Q \times Op_n \;+\; \{\, (f; q_1, \cdots, q_{ar(f)}) : f \in \Sigma, q_i \in Q \,\})$$

(We require δ to respect the standard convention that $\bot$ cannot be pushed onto or popped from the stack.)

Configurations of an n-CPDA are pairs of the form (q, s) where $q \in Q$ and s is an n-stack over Γ; we call $(q_0, \bot_n)$ the *initial configuration*. A *generalized configuration* (ranged over by γ, γ_i, etc.) is either a configuration or a triple of the form $(f; q_1, \cdots, q_{ar(f)}; s)$. We define a *one-step labelled transition relation* of $\mathcal{A}$ over generalized configuration by clauses, one for each of the three kinds[4] of label ***I***, ***P*** and ***O***:

I. $(q, s) \overset{(q',\theta)}{>} (q', s')$, if for some $\theta \in Op_n$, we have $\delta(q, top_1\, s) = (q', \theta)$ and $s' = \theta(s)$

P. $(q, s) \overset{(f;\overline{q})}{>} (f; q_1, \cdots, q_{ar(f)}; s)$, if $\delta(q, top_1\, s) = (f; q_1, \cdots, q_{ar(f)})$

O. $(f; q_1, \cdots, q_{ar(f)}; s) \overset{(f,i)}{>} (q_i, s)$, for each $1 \leq i \leq ar(f)$.

The labelled transition relation is *deterministic* in the sense that for any generalized configuration γ and for any label ℓ, if $\gamma \overset{\ell}{>} \gamma_1$ and $\gamma \overset{\ell}{>} \gamma_2$ then $\gamma_1 = \gamma_2$. We write $\gamma_1 > \gamma_2$ just if $\gamma_1 \overset{\ell}{>} \gamma_2$ for some label ℓ. Note that the unlabelled $>$ is not deterministic in general, but it is when restricted to configurations (because δ is).

A *computation path* of $\mathcal{A}$ is a finite or infinite transition sequence of the form

$$\rho \quad = \quad \gamma_0 \overset{\ell_0}{>} \gamma_1 \overset{\ell_1}{>} \gamma_2 \overset{\ell_2}{>} \cdots$$

where γ_0 is the initial configuration. Every computation path is uniquely determined by the associated *label sequence*, namely, $\ell_0\, \ell_1\, \ell_2 \cdots$. Observe that such label sequences satisfy the regular expression $(I^*\, P\, O)^\omega + (I^*\, P\, O)^*\, I^\omega$ if the sequence is infinite, and $(I^*\, P\, O)^*\, I^*(\varepsilon + P + P\, O)$ if the sequence is finite. The Σ-*projection* of ρ is the subsequence $\ell_{r_1}\, \ell_{r_2}\, \ell_{r_3}\, \cdots$ of labels of the shape (f, i) (in which case $ar(f) \geq 1$) or of the shape $(f; \varepsilon)$ (in which case $ar(f) = 0$, and the label occurs at the end of the Σ-projection). We say the CPDA $\mathcal{A}$ generates the Σ-labelled tree t just in case the *branch language*[5] of t coincides with the Σ-projection of computation paths of $\mathcal{A}$.

Remark 2.3 Order-2 collapsible pushdown automata (2-CPDA) are a slight variant of *pushdown automata with links* in [2, 1]. They are essentially the same as *panic automata* in the sense of Knapik *et al.* [16], which use numeric indices to represent links.

3 Recursion schemes

Types are generated by the grammar $A ::= o \mid A \to A$. Every type $A \neq o$ can be written uniquely as $A_1 \to \cdots \to A_n \to o$ (by convention arrows associate to the right) which we shall abbreviate to $(A_1, \cdots, A_n, o)$, for some $n \geq 1$ which is called its *arity*; the base type o has arity 0. We define the *order* of a type by: $ord(o) = 0$ and $ord(A \to B) = \max(ord(A) + 1, ord(B))$. Let Σ be a *ranked alphabet* i.e. each Σ-symbol f has an arity

[4] ***I*** for internal or hidden Player-move, ***P*** for Player-move, and ***O*** for Opponent-move.

[5] The ***branch language*** of $t : \mathsf{Dom}(t) \longrightarrow \Sigma$ consists of infinite words $(f_1, d_1)(f_2, d_2) \cdots$ just if for $0 \leq i < n$, we have $t(d_1 \cdots d_i) = f_{i+1}$; and of finite words $(f_1, d_1) \cdots (f_n, d_n)a$ just if for $0 \leq i < n$, we have $t(d_1 \cdots d_i) = f_{i+1}$ and $t(d_1 \cdots d_n) = a$.

$ar(f) \geq 0$ which determines its type $\underbrace{o \to \cdots \to o \to}_{ar(f)} o$. Further we shall assume that each symbol $f \in \Sigma$ is assigned a finite set $\mathsf{Dir}(f) = \{ 1, \cdots, ar(f) \}$ of *directions*, and we define $\mathsf{Dir}(\Sigma) = \bigcup_{f \in \Sigma} \mathsf{Dir}(f)$. Let D be a set of directions; a *D-tree* is just a prefix-closed subset of D^*, the free monoid of D. A ***Σ-labelled tree*** is a function $t : \mathsf{Dom}(t) \longrightarrow \Sigma$ such that $\mathsf{Dom}(t)$ is a $\mathsf{Dir}(\Sigma)$-tree, and for every node $\alpha \in \mathsf{Dom}(t)$, the Σ-symbol $t(\alpha)$ has arity k if and only if α has exactly k children and the set of its children is $\{ \alpha\, 1, \cdots, \alpha\, k \}$ i.e. t is a *ranked* (and ordered) tree. We write $\mathcal{T}^\infty(\Sigma)$ for the set of (finite and infinite) Σ-labelled ranked and ordered trees.

3.1 The tree generated by a (deterministic) recursion scheme

Let Ξ be a set of typed symbols. The set of *applicative terms of type A generated from Ξ*, written $\mathcal{T}_A(\Xi)$, is defined by induction over the following rules: if $f : A$ is an element of Ξ then $f \in \mathcal{T}_A(\Xi)$; if $s \in \mathcal{T}_{A \to B}(\Xi)$ and $t \in \mathcal{T}_A(\Xi)$ then $st \in \mathcal{T}_B(\Xi)$. For simplicity we write $\mathcal{T}(\Xi)$ to mean $\mathcal{T}_o(\Xi)$, the set of ground terms. In case Ξ is a ranked alphabet (and so every Ξ-symbol has an order-0 or order-1 type determined by its arity) we identify terms in $\mathcal{T}(\Xi)$ with the finite trees in $\mathcal{T}^\infty(\Xi)$.

For each type A, we assume an infinite set Var_A of variables of type A, such that Var_A and Var_B are disjoint whenever $A \neq B$; and we write Var for the union of Var_A as A ranges over types. We use letters $x, y, \varphi, \psi, \chi, \xi$ etc. to range over variables. We write $s : A$ to mean "the expression s has type A". A (deterministic) ***recursion scheme*** is a 4-tuple $G = \langle \Sigma, \mathcal{N}, \mathcal{R}, S \rangle$ where

- Σ is a ranked alphabet of *terminals* (including a distinguished symbol $\bot : o$)
- $\mathcal{N}$ is a finite set of typed *non-terminals*; we use upper-case letters F, H, etc. to range over non-terminals
- $S \in \mathcal{N}$ is a distinguished *start symbol* of type o
- $\mathcal{R}$ is a finite set of rewrite rules, one for each non-terminal $F : (A_1, \cdots, A_n, o)$, of the form

  $$F\, \xi_1 \cdots \xi_n \;\to\; e$$

  where each ξ_i is a variable of type A_i, and e is an applicative term in $\mathcal{T}(\Sigma \cup \mathcal{N} \cup \{ \xi_1, \cdots, \xi_n \})$. Note that the expressions on either side of the arrow are terms of ground type.

The *order* of a recursion scheme is defined to be the highest order of (the types of) its non-terminals.

Value tree $[\![G]\!]$ of a recursion scheme G

In this paper we use recursion schemes as generators of Σ-labelled trees. Informally the *value tree*[6] of (or the tree *generated* by) a recursion scheme G, $[\![G]\!]$, is a possibly infinite applicative term (of ground type), *constructed from the terminals in* Σ, that is obtained by unfolding the rewrite rules of G *ad infinitum*, replacing formal by actual parameters each time, starting from the start symbol S.

To define $[\![G]\!]$, we first introduce a map $(\cdot)^\bot : \mathcal{T}(\Sigma \cup \mathcal{N}) \longrightarrow \mathcal{T}(\Sigma)$ that takes an applicative term and replaces each non-terminal, together with its arguments, by $\bot$. We define $(\cdot)^\bot$ by structural recursion as

[6] We would like to refer to the Σ-labelled tree generated by a recursion scheme as its *value* tree, because the name is a good counterpoint to *computation tree*. We have in mind here the distinction between *value* and *computation* emphasized by Moggi [19]. The idea is that the value tree is obtained from the computation tree by a (possibly infinite) process of evaluation.

follows: we let f range over Σ-symbols, and F over non-terminals in $\mathcal{N}$

$$\begin{array}{rcl} f^{\perp} & = & f \\ F^{\perp} & = & \perp \\ (st)^{\perp} & = & \begin{cases} \perp & \text{if } s^{\perp} = \perp \\ (s^{\perp}t^{\perp}) & \text{otherwise.} \end{cases} \end{array}$$

Clearly if $s \in \mathcal{T}(\Sigma \cup \mathcal{N})$ is of ground type then $s^{\perp} \in \mathcal{T}(\Sigma)$ is of ground type. Henceforth we shall identify ground-type terms in $\mathcal{T}(\Sigma)$ with finite trees in $\mathcal{T}^{\infty}(\Sigma)$.

Next we define a one-step reduction relation $\rightarrow_G$ which is a binary relation over terms in $\mathcal{T}(\Sigma \cup \mathcal{N})$. Informally $s \rightarrow_G s'$ just if s' is obtained from s by replacing some occurrence of a non-terminal F by the right-hand side of its rewrite rule in which all formal parameters are in turn replaced by their respective actual parameters, subject to the proviso that the F must occur at the head of a subterm of ground type. Formally $\rightarrow_G$ is defined by induction over the following rules:

- (*Substitution*). $Ft_1 \cdots t_n \rightarrow_G e[t_1/\xi_1, \cdots, t_n/\xi_n]$ where $F\xi_1 \cdots \xi_n \rightarrow e$ is a rewrite rule of G.
- (*Context*). If $t \rightarrow_G t'$ then $(st) \rightarrow_G (st')$ and $(ts) \rightarrow_G (t's)$.

The relation $\downarrow_G$ between terms and trees is then defined as follows: We say that $s \downarrow_G t$ where $s \in \mathcal{T}(\Sigma \cup \mathcal{N})$ and $t \in \mathcal{T}^{\infty}(\Sigma)$ just if

- there is a finite reduction sequence $s = t_0 \rightarrow_G \ \cdots \ \rightarrow_G t_n = t$, and t is a finite tree, none of whose node is labelled $\perp$; or
- there is an infinite reduction sequence $s = t_0 \rightarrow_G t_1 \rightarrow_G t_2 \cdots$ such that $t = \lim\langle t_i^{\perp} : i \in \omega \rangle$, and t may be a finite tree (in which case, some of t's nodes are labelled $\perp$) or an infinite tree.

Recall that $\mathcal{T}^{\infty}(\Sigma)$ is a complete partial order with respect to the approximation ordering $\sqsubseteq$ as defined by: $t \sqsubseteq t'$ just if $\mathsf{Dom}(t) \subseteq \mathsf{Dom}(t')$ and for all $w \in \mathsf{Dom}(t)$, we have $t(w) = \perp$ or $t(w) = t'(w)$ (i.e. t' is obtained from t by replacing some $\perp$-labelled nodes by Σ-labelled trees). We can finally define the Σ-labelled ranked tree $[\![G]\!]$, called the ***value tree*** of (or the tree *generated* by) G, as follows:

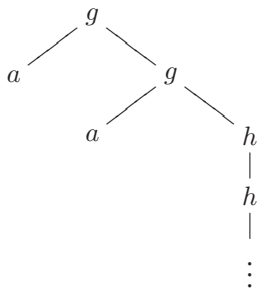
$$[\![G]\!] \ = \ \sup\{ t \in \mathcal{T}^{\infty}(\Sigma) : S \downarrow_G t \}.$$

The supremum is well-defined because the set in question is directed, which is a consequence of the Church-Rosser property of G viewed as a rewrite system. We write ***RecTree***$_n\Sigma$ for the class of value trees $[\![G]\!]$ where G ranges over order-n recursion schemes.

Example 3.1 Let G be the order-2 (unsafe[7]) recursion scheme with rewrite rules:

$$\begin{array}{rcl} S & \to & H\,a \\ H\,z & \to & F\,(g\,z) \\ F\,\varphi & \to & \varphi\,(\varphi\,(F\,h)) \end{array}$$

where $z : o$ and $\varphi : (o, o)$, and the arities of the terminals g, h, a are $2, 1, 0$ respectively, and the type of a variable is written as its superscript. The value tree $[\![\, G \,]\!]$ is the Σ-labelled tree representing the infinite term $g\,a\,(g\,a\,(h\,(h\,(h\,\cdots))))$:



The only infinite *path* in the tree is the node-sequence $\varepsilon \cdot 2 \cdot 22 \cdot 221 \cdot 2211 \cdots$.

3.2 Graph representing a recursion scheme

We write $[n]$ as a shorthand for $\{\, 1, \cdots, n \,\}$ and $[n]_0$ for $\{\, 0, \cdots, n \,\}$. Fix a ranked alphabet Σ. Typically[8] $\mathsf{Dir}(f) = [ar(f)]$ (but we always have $|\mathsf{Dir}(f)| = ar(f)$ for each Σ-symbol f). We define $\mathsf{Dir}(\Sigma) = \bigcup_{f \in \Sigma} \mathsf{Dir}(f)$.

We recall the long transform of a recursion scheme as introduced in [21]. Fix a recursion scheme G. Rules of the new recursion scheme $\overline{G}$ (which, we shall see, can be regarded as order 0) are obtained from those of G by applying the following four operations in turn, which we call ***long transform***. For each G-rule:

1. *Expand the RHS to its η-long form.* I.e. we hereditarily η-expand every subterm – even if it is of ground type – provided it occurs in an *operand position* (i.e. it is the second argument of some occurrence of the application operator). Note that each applicative term $s \in \mathcal{T}(\Sigma \cup \mathcal{N} \cup \{\, \xi_1, \cdots, \xi_l \,\})$ can be written uniquely as $\dagger\, s_1 \cdots s_m$ where $\dagger$ is either a variable (i.e. some ξ_j) or a non-terminal or a terminal. Suppose $\dagger\, s_1 \cdots s_m : (A_1, \cdots, A_n, o)$. First we define

$$\ulcorner \dagger\, s_1 \cdots s_m \urcorner \quad = \quad \boldsymbol{\lambda}\overline{\varphi}.\dagger \ulcorner s_1 \urcorner \cdots \ulcorner s_m \urcorner \ulcorner \varphi_1 \urcorner \cdots \ulcorner \varphi_n \urcorner$$

   where $\overline{\varphi}$ is a list $\varphi_1 \cdots \varphi_n$ of (fresh) pairwise-distinct variables (which is a null list iff $n = 0$) of types $A_1, \cdots, A_n$ respectively, none of which occurs free in $\dagger \ulcorner s_1 \urcorner \cdots \ulcorner s_m \urcorner$. Take any $e = \dagger\, s_1 \cdots s_m$ of ground type. The η-***long form*** of e is defined to be $\dagger \ulcorner s_1 \urcorner \cdots \ulcorner s_m \urcorner$.

[7] The *safety* constraint (on lambda terms) may be regarded as a reformulation of the constraint on lambda terms imposed by Damm's *derived types*, first introduced in his major study on the semantics of Algol-like languages [8]. To define ***safety***, we first need to introduce ***homogeneous types***: The base type o is *homogeneous*; a function type $A_1 \to (A_2 \to \cdots \to (A_n \to o) \cdots)$ is *homogeneous* just if each A_i is homogeneous, and $ord(A_1) \geq ord(A_2) \geq \cdots \geq ord(A_n)$. We say that a term (or a rewrite rule or a recursion scheme) is *homogeneously typed* just if all types that occur in it are homogeneous. Knapik *et al.* [15] define safety as follows: A homogeneously-typed term of order $k > 0$ is said to be *unsafe* if it contains an occurrence of a parameter of order strictly less than k, otherwise the term is *safe*. An occurrence of an unsafe term t, as a subexpression of a term t', is ***safe*** if it occurs in an operand position (i.e. it is in the context $\cdots (ts) \cdots$), otherwise the occurrence is *unsafe*. A recursion scheme is ***safe*** if no unsafe term has an unsafe occurrence in the right-hand side of any rewrite rule. Note that it follows from the definition that all recursion schemes of order at most 1 are safe.

[8] The only exception is the symbol $@_A$ of the auxiliary alphabet Λ_G, where we have $\mathsf{Dir}(@_A) = [ar(@_A) - 1]_0$.

For example the η-long form of $g\,a : o$ is $g\,(\boldsymbol{\lambda}.a)$; we shall see that the "dummy lambda-abstraction"[9] $\boldsymbol{\lambda}.a$ (that binds a *null* list of variable) plays a useful rôle in the syntactic representation of the game semantics of a recursion scheme.

2. *Insert long-apply symbols* $@_A$: Replace each ground-type subterm of the shape $D\,e_1 \cdots e_n$, where $D : (A_1, \cdots, A_n, o)$ is a non-terminal and $n \geq 1$ (i.e. D has order at least 1), by $@_A\,D\,e_1 \cdots e_n$ where $A = ((A_1, \cdots, A_n, o), A_1, \cdots, A_n, o)$. In the following, we shall often omit the type tag A from $@_A$.

3. *Curry the rewrite rule*. I.e. we transform the rule $F\,\varphi_1\,\cdots\,\varphi_n \;\rightarrow\; e'$ to

$$F \;\rightarrow\; \boldsymbol{\lambda}\varphi_1\,\cdots\,\varphi_n.e'.$$

   In case $n = 0$, note that the curried rule has the form $F \;\rightarrow\; \boldsymbol{\lambda}.e'$.

4. *Rename bound variables afresh*, so that any two variables that are bound by different lambdas have different names.

For every recursion scheme G, the system of transformed rules in $\overline{G}$ defines an order-0 recursion scheme – called the *long transform* of G – with respect to an enlarged ranked alphabet Λ_G, which is Σ augmented by certain variables and lambdas (of the form $\boldsymbol{\lambda}\overline{\xi}$ which is a short hand for $\boldsymbol{\lambda}\xi_1 \cdots \xi_n$ where $n \geq 0$) *but regarded as terminals*. The alphabet Λ_G is a finite subset of the set

$$\underbrace{\Sigma \;\cup\; \mathit{Var} \;\cup\; \{\, @_A : A \in \mathit{ATypes} \,\}}_{\textbf{\textit{Non-lambdas}}} \quad\cup\quad \underbrace{\{\, \boldsymbol{\lambda}\overline{\xi} : \overline{\xi} \subseteq \mathit{Var} \,\}}_{\textbf{\textit{Lambdas}}}$$

where *ATypes* is the set of types of the shape $((A_1, \cdots, A_n, o), A_1, \cdots, A_n, o)$ with $n \geq 1$. We rank the symbols in Λ_G as follows:

- variable symbol $\varphi : (A_1, \cdots, A_n, o)$ in Var has arity n
- long-apply symbol $@_A$ where $A = ((A_1, \cdots, A_n, o), A_1, \cdots, A_n, o)$ has arity $n + 1$
- lambda symbol $\boldsymbol{\lambda}\overline{\xi}$ has arity 1, for every list of variables $\overline{\xi} \subseteq \mathit{Var}$.

Further, for $f \in \Lambda_G$, we define

$$\mathsf{Dir}(f) \;=\; \begin{cases} [ar(@_A) - 1]_0 & \text{if } f = @_A \\ [ar(f)] & \text{otherwise} \end{cases}$$

For technical reasons (to be clarified shortly), the leftmost child of an @-labelled node α is in direction 0 (i.e. it is α's 0-child); for all other nodes, the leftmost child is in direction 1. The *non-terminals* of $\overline{G}$ are exactly those of G, except that each is assigned a new type, namely, o. We can now define the ***computation tree*** $\lambda(G)$ to be the value tree $[\![\,\overline{G}\,]\!]$ of the order-0 recursion scheme $\overline{G}$. It follows that $\lambda(G)$ is a regular tree.

A Λ-***labelled deterministic digraph*** (or DDG, for short) is a quadruple

$$\mathcal{K} \quad=\quad \langle\, V, \quad E \subseteq V \times V, \quad l : V \longrightarrow \Lambda, \quad v_0 \in V \,\rangle$$

where the underlying digraph $\langle\, V, E \,\rangle$ is vertex-labelled by the function $l : V \longrightarrow \Lambda$ (where Λ is a ranked alphabet), and edge-labelled by $\mathsf{Dir}(\Lambda)$ in that $E = \bigcup_{i \in \mathsf{Dir}(\Lambda)} E_i$, such that

[9] To my knowledge, Colin Stirling was the first to use a tree representation of lambda terms in which "dummy lambdas" are employed; see his CSL 2005 paper [23]. Motivated by property-checking games in Verification, he has introduced a game that is played over such trees as a characterization of higher-order matching [24].

(i) for each $i \in \mathsf{Dir}(\Lambda)$, we have $E_i \subseteq V \times V$ is a partial function

(ii) for each $v \in V$, and each $i \in \mathsf{Dir}(l(v))$, we have $E_i(v)$ is defined i.e. $\{\, v' : (v, v') \in E_i \,\}$ is a singleton set.

In the following we shall assume that V is finite. It is easy to see that every finite Λ-labelled tree can be presented as a DDG, and the unfolding of a Λ-labelled DDG is a Λ-labelled tree.

Fix a higher-order recursion scheme G and an associated long transform $\overline{G}$. We define the ***HORS graph*** $\mathsf{Gr}(G)$ to be the Λ_G*-labelled DDG determined by* G

$$\mathsf{Gr}(G) \quad = \quad \langle\, V, \quad E \subseteq V \times V, \quad \lambda_G : V \longrightarrow \Lambda_G, \quad v_0 \in V \,\rangle$$

thatq is obtained by the following procedure:

1. First we define the ranked alphabet $\Lambda_G^+ = \Lambda_G \cup \mathcal{N}_{\overline{G}}$ where each symbol in $\mathcal{N}_{\overline{G}}$ (i.e. a non-terminal of $\overline{G}$) is given arity 0.
2. For each $\overline{G}$-rule (say) $F \;\rightarrow\; \boldsymbol{\lambda}\varphi_1 \cdots \varphi_n.e$, the corresponding Λ_G^+-labelled DDG
$$\mathcal{D}_F \quad = \quad \langle\, V_F, \quad E^F \subseteq V_F \times V_F, \quad l_F : V_F \longrightarrow \Lambda_G^+, \quad rt_F \,\rangle$$
given by the Λ_G^+-labelled tree that is determined by the right-hand side of the rule, namely, $\boldsymbol{\lambda}\varphi_1 \cdots \varphi_n.e$. Note that we have $l_F(rt_F) = \boldsymbol{\lambda}\varphi_1 \cdots \varphi_n$ with reference to the rule F given above.
3. Set the digraph $\mathcal{D}$ to be the disjoint union of the underlying digraph of $\mathcal{D}_F$, as F ranges over $\mathcal{N}_{\overline{G}}$. We then define the underlying digraph of $\mathsf{Gr}(G)$ to be $\mathcal{D}$ quotiented by the equivalence classes $\mathcal{E}_F$, one for each F in $\mathcal{N}_{\overline{G}}$; where we define
$$\mathcal{E}_F \quad = \quad \left(\bigcup_{H \in \mathcal{N}_{\overline{G}}} l_H^{-1}(\{\, F \,\}) \right) \;\cup\; \{\, rt_F \,\}$$
I.e. in $\mathsf{Gr}(G)$ all vertices in $\mathcal{E}_F$ are identified, for each $F \in \mathcal{N}_{\overline{G}}$. Henceforth, as a vertex of $\mathsf{Gr}(G)$, we shall refer to the equivalence class $\mathcal{E}_F$ by the representative rt_F.

   The edge-labels of $\mathsf{Gr}(G)$ are inherited from the edge-labels of the component DDGs $\mathcal{D}_F$ (we define $E_i(rt_F) = E_i^F(rt_F)$ for each $F \in \mathcal{N}_{\overline{G}}$). The vertex-labels are defined by
$$\lambda_G(v) \quad = \quad \begin{cases} l_F(rt_F) & \text{if } v = \mathcal{E}_F \text{ for some } F \in \mathcal{N}_{\overline{G}} \\ l_H(v) & \text{otherwise, suppose } v \text{ is a vertex in } V_H \end{cases}$$
   The root v_0 of $\mathsf{Gr}(G)$ is rt_S, where S is the start symbol of $\overline{G}$.

In the following, we shall only concern ourselves with the connected component of $\mathsf{Gr}(G)$ that contains the root node (and assume that $\mathsf{Gr}(G)$ *is* that connected component). It is easy to see that unfolding $\mathsf{Gr}(G)$ gives the computation tree $\lambda(G)$.

Example 3.2 We revisit the recursion scheme of Example 3.1 and consider the graph determined by it. First the long transform:

$$G \;:\; \begin{cases} S & = & G\,a \\ G\,z & = & F\,(g\,z) \\ F\,\varphi & = & \varphi\,(\varphi\,(F\,h)) \end{cases} \quad \mapsto \quad \overline{G} \;:\; \begin{cases} S & = & \boldsymbol{\lambda}.@\,G\,(\boldsymbol{\lambda}.a) \\ G & = & \boldsymbol{\lambda}z.@\,F\,(\boldsymbol{\lambda}y.g\,(\boldsymbol{\lambda}.z)\,(\boldsymbol{\lambda}.y)) \\ F & = & \boldsymbol{\lambda}\varphi.\varphi\,(\boldsymbol{\lambda}.\varphi\,(\boldsymbol{\lambda}.@\,F\,(\boldsymbol{\lambda}x.h\,(\boldsymbol{\lambda}.x)))) \end{cases}$$

The graph $\mathsf{Gr}(G)$ is then given in Figure 1.

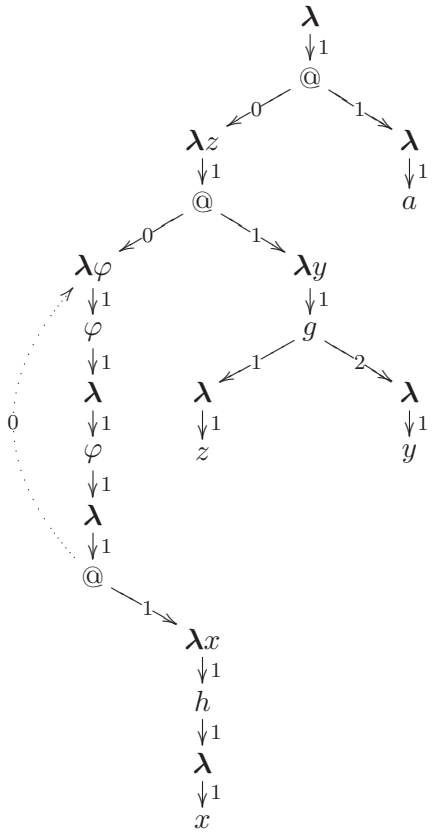


Figure 1: The graph determined by an order-2 recursion scheme.

Notations and features of HORS graphs

Fix a HORS graph $\mathsf{Gr}(G) = \langle\, V, E, \lambda_G, v_0 \,\rangle$. We shall call a node of $\mathsf{Gr}(G)$ ***prime*** just if it is the 0-child[10] of a @-labelled node. By construction, a prime node is labelled by a lambda. We define the ***depth*** of a node to be the length of the shortest path from the root to the node (so that the root has depth 0). Let u be a node. We define $\mathsf{pred}(u) = \{\, u' \in V : (u', u) \in E \,\}$ i.e. the set of *predecessors* of u. For every node u labelled by a variable φ_i (say), its ***binder***, written $binder(u)$, is the node that is labelled $\boldsymbol{\lambda}\overline{\varphi}$, where $\overline{\varphi}$ is a list of variables that contains φ_i. (Since bound variables are renamed to prevent any clash in the construction of $\overline{G}$, every variable node in $\mathsf{Gr}(G)$ has a unique binder.) We say that u is the *i-parameter* of $binder(u)$ just if φ_i is the ith-item of the list $\overline{\varphi}$. The ***span*** of the variable node u is defined to be the depth of $binder(u)$ minus the depth of u.

We note the following features of HORS graphs:

(i) Except the root and possibly some prime nodes, every node u has a unique predecessor.

(ii) For every non-root node u, there is some j such that for *every* predecessor v of u, u is the j-child of v; hence (and in this case) we can say that u is a ***j-child***. Indeed a node is a 0-child if and only if it is prime.

(iii) For every node u, there is a unique shortest path from $binder(u)$ to u, and this path does not contain any prime node.

For convenience, and whenever it is safe to do so, we shall confuse a node u with its Λ_G-label $\lambda_G(u)$.

4 From CPDA to recursion schemes

In this section we prove that for every $n \geq 0$, and every n-CPDA $\mathcal{A}$, there is an order-n recursion scheme $G_{\mathcal{A}}$ such that $\mathcal{A}$ and $G_{\mathcal{A}}$ define the same Σ-labelled tree. We begin by introducing a method to represent higher-order stacks and configurations by applicative terms constructed from non-terminals of the same order; the correctness of the representation is then established in Theorem 3. The Theorem is quite general: it is independent of the transition relation of the automaton, nor does it matter whether the automaton is for defining a tree or a graph. In case the automaton $\mathcal{A}$ is for generating a Σ-labelled tree t (say), we show in Section 4.3 how an order-n deterministic recursion scheme $G_{\mathcal{A}}$ can be constructed that defines the same tree i.e. $[\![\, G_{\mathcal{A}} \,]\!] = t$. The correctness of the transform $G_{\mathcal{A}}$ (Theorem 5) is then obtained as a corollary of Theorem 3.

Our construction simplifies the translation (order-2) in [16] and generalizes it to all finite orders. For convenience let the state-set of $\mathcal{A}$ be $[m]$ where $m \geq 1$. Let 0 be the base type. Inductively, for $n \geq 0$, we define the type

$$n+1 \quad = \quad n^m \to n$$

where $n^m = \underbrace{n \times \cdots \times n}_{m \text{ times}}$. Thus $n+1 = n^m \to (n-1)^m \to \cdots \to 0^m \to 0$.

4.1 Term representation of stacks and configurations

Fix an order-n CPDA $\mathcal{A}$. We shall first introduce a general scheme for representing stacks and configurations of $\mathcal{A}$ by applicative terms generated from the non-terminals from $\mathcal{N}_{\mathcal{A}}$. The key result in this section is a correctness theorem (Theorem 3) for the representation scheme.

[10] The leftmost child of a @-labelled node is the latter's 0-child (i.e. the child is at the end of a 0-labelled edge); the leftmost child of any other node is a 1-child.

Recall that every non-$\perp$ symbol in an n-stack has a link to some stack (of order necessarily less than n) that lies below it in the stack. If the stack pointed to is of order $j-1$ where $1 \leq j \leq n$, the link is said to be a *j-link*. A 1-link from a symbol *always* points to the symbol *immediately* below it; no stack operation will alter it. Henceforth we shall use the formal definition of higher-order stacks, according to which a symbol with link takes the form $a^{(j,k)}$. For technical convenience, we require that $j = k = 1$ in case $a = \perp$. E.g. the initial configuration is $(q_0, \perp_n)$ with $top_1 \perp_n = \perp^{(1,1)}$.

For each stack symbol a, each $1 \leq e \leq n$ and each state $1 \leq p \leq m$, we introduce a non-terminal

$$\mathcal{F}_p^{a,e} \ : \ (n-e)^m \to (n-1)^m \to \cdots \to 0^m \to 0$$

(Note that the type of $\mathcal{F}_p^{a,e}$ is non-homogeneous in the sense of Knapik *et al.* [15].) In addition, for each $0 \leq i \leq n-1$, we introduce a non-terminal $\Omega_i : i$; and we fix a start symbol $S : 0$. The set $\mathcal{N}_\mathcal{A}$ of *non-terminals* is defined as follows:

$$\mathcal{N}_\mathcal{A} \ = \ \{\, \mathcal{F}_p^{a,e} : a \in \Gamma, 1 \leq p \leq m, 1 \leq e \leq n \,\} \ \cup \ \{\, \Omega_i : 0 \leq i \leq n-1 \,\} \ \cup \ \{\, S : 0 \,\}$$

where $S : 0$ is the distinguished start symbol. Let $\mathcal{T}^j(\mathcal{N}_\mathcal{A})$ be the set of *applicative terms* (or simply *terms*) of type j generated from the elements of $\mathcal{N}_\mathcal{A}$. In the following we shall use the following shorthand. Let $P(i)$ be a term with an occurrence of i; we write $\langle P(i) \,|\, i \rangle$ to mean the m-tuple $\langle\, P(1), \cdots, P(m) \,\rangle$. E.g. $\langle \mathcal{F}_i^{a,e} \,|\, i \rangle$ means $\langle\, \mathcal{F}_1^{a,e}, \cdots, \mathcal{F}_m^{a,e} \,\rangle : ((n-e)^m \to n)^m$.

A term $M : n-j$ where $0 \leq j \leq n$ is said to be ***head normal*** if its head symbol is a non-terminal of the form $\mathcal{F}_p^{a,e}$ i.e. M has the shape $\mathcal{F}_p^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_{n-j}}$, for some a, e and p and for some vectors of terms $\overline{L}, \overline{M_{n-1}}, \cdots, \overline{M_{n-j}}$ of the appropriate types; we shall call $\mathcal{F}_p^{a,e}$ the ***head non-terminal*** of M. Let $0 \leq j \leq n$, $1 \leq p \leq m$ and let s be a j-stack, a pair of the form (p, s) is called a ***j-configuration*** (thus a configuration is an n-configuration). The idea is that we use head-normal terms of type $n-j$ – which has the general shape $\mathcal{F}_p^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_{n-j}} \ : \ n-j$ – to represent j-configurations; equivalently we use m-tuples of the form

$$\langle \mathcal{F}_i^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_{n-j}} \,|\, i \rangle \ : \ (n-j)^m$$

to represent j-stacks.

Suppose the head-normal ground-type term

$$\mathcal{F}_p^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_0} \ : \ 0$$

represents the configuration (p, s). Then the 0-configuration $(p, top_1\, s)$ — where the top_1-element of s, say, a with a link to the e-stack represented by the m-tuple $\overline{L} : (n-e)^m$ — is represented by $\mathcal{F}_p^{a,e}\, \overline{L} : n$. Further for each $1 \leq j \leq n$ and $1 \leq p \leq m$, we have:

- The $(j-1)$-configuration $(p, top_j\, s)$ is represented by $\mathcal{F}_p^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_{n-(j-1)}} : n-(j-1)$.
- The configuration $(p, pop_j\, s)$ is represented by $M_{n-j,p}\, \overline{M_{n-j-1}} \cdots \overline{M_0} : 0$.
- The configuration $(p, collapse\, s)$ is represented by $L_p\, \overline{M_{n-e-1}} \cdots \overline{M_0} : 0$.

Take a head-normal ground-type term $\mathcal{F}_p^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_0} : 0$. For each $1 \leq j \leq n$, we shall call the m-tuple $\overline{M_{n-j}} \ : \ (n-j)^m$ its $(n-j)$-***factor***; by abuse of language, we shall call $\overline{L} \ : \ (n-e)^m$ its n-*factor*. Let $1 \leq j \leq n$; we say that ground-type terms M and N are $(n-j)$-***similar***, written $M \sim_{n-j} N$, just if they have the same head non-terminal, and for each $0 \leq k \leq n$, provided $k \neq j$, M and N have the same $(n-k)$-factors. It follows from the definition that if M and N are $(n-j)$-similar and have the same $(n-j)$-factor, then they are (syntactically) identical terms.

Labelled rewrite rules and the labelled transition relation

Definition 4.1 (i) We first consider labelled rewrite rules of the general form

$$\mathcal{F}_p^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0} \quad \stackrel{(q,\theta)}{\longrightarrow} \quad \Xi_{(q,\theta)}$$

where for each $0 \leq j \leq n-1$, we have $\overline{\Psi_j} = \Psi_{j1}, \cdots, \Psi_{jm}$ is a vector of variables, with each $\Psi_{ji} : j$; similarly $\overline{\Phi} = \Phi_1, \cdots, \Phi_m$ is a vector of variables, with each $\Phi_i : n-e$. The shape of $\Xi_{(q,\theta)}$ depends on the pair (q,θ), as shown in Table 1 below, where $2 \leq j \leq n$ and $1 \leq e, k \leq n$ in the following: In the following,

Cases of (q,θ)	**Corresponding** $\Xi_{(q,\theta)}$
$(q, push_1^{b,k})$	$\mathcal{F}_q^{b,k}\,\overline{\Psi_{n-k}}\,\langle \mathcal{F}_i^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}} \mid i\rangle\,\overline{\Psi_{n-2}}\cdots\overline{\Psi_0}$
$(q, push_j)$	$\mathcal{F}_q^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_{n-(j-1)}}\langle \mathcal{F}_i^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_{n-j}} \mid i\rangle\overline{\Psi_{n-(j+1)}}\cdots\overline{\Psi_0}$
(q, pop_k)	$\Psi_{n-k,q}\,\overline{\Psi_{n-k-1}}\cdots\overline{\Psi_0}$
$(q, coll.)$	$\Phi_q\,\overline{\Psi_{n-e-1}}\cdots\overline{\Psi_0}$

Table 1: Definition of $\Xi_{(q,\theta)}$

whenever we use θ, instead of (p,θ), to label a rewrite rule

$$\mathcal{F}_p^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0} \quad \stackrel{\theta}{\longrightarrow} \quad \Xi_{(p,\theta)}$$

it is understood that the state p is preserved.

(ii) The labelled rewrite rules induce a family of *outermost* labelled one-step transition relations $\stackrel{(q,\theta)}{\longrightarrow} \subseteq \mathcal{T}^0(\mathcal{N}_\mathcal{A}) \times \mathcal{T}^0(\mathcal{N}_\mathcal{A})$, indexed by the label (q,θ), where q ranges over states and θ ranges over stack operations. Informally we define $M \stackrel{(q,\theta)}{\longrightarrow} M'$ just if M' is obtained from M by replacing the *head* (equivalently, outermost) non-terminal F by the right-hand side of the corresponding rewrite rule in which all formal parameters are in turn replaced by their respective actual parameters. Formally $\stackrel{(q,\theta)}{\longrightarrow}$ is defined by the following rule schemes: for $a \in \Gamma$, $1 \leq p, q \leq m$, $1 \leq e \leq n$, $\theta \in Op_n$, and for each rewrite rule $\mathcal{F}_p^{a,e}\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0} \stackrel{(q,\theta)}{\longrightarrow} \Xi_{(q,\theta)}$, we have the rule scheme

$$\mathcal{F}_p^{a,e}\overline{L}\,\overline{M_{n-1}}\cdots\overline{M_0} \quad \stackrel{(q,\theta)}{\longrightarrow} \quad \Xi_{(q,\theta)}[\overline{L}/\overline{\Phi}, \overline{M_{n-1}}/\overline{\Psi_{n-1}}\cdots, \overline{M_0}/\overline{\Psi_0}]$$

where $\overline{L}, \overline{M_{n-1}}, \cdots, \overline{M_0}$ range over vectors of terms respecting the type of $\mathcal{F}_p^{a,e}$.

Note that each binary relation $\stackrel{(q,\theta)}{\longrightarrow}$ is a partial function. Let $\alpha = \theta_1 \,;\cdots;\, \theta_l$ be a (composite) sequence of stack operations. We write $\stackrel{\alpha}{\longrightarrow} \subseteq \mathcal{T}^0(\mathcal{N}_\mathcal{A}) \times \mathcal{T}^0(\mathcal{N}_\mathcal{A})$ to be the sequential composition of the partial function $\stackrel{\theta_1}{\longrightarrow}, \cdots, \stackrel{\theta_l}{\longrightarrow}$ (in this order). As $\stackrel{\alpha}{\longrightarrow}$ is a partial function, whenever there is an M' such that $M \stackrel{\alpha}{\longrightarrow} M'$, we shall often use the postfix notation $M\stackrel{\alpha}{\longrightarrow}$ to denote (the necessarily unique) M'.

It is straightforward to check that for any head-normal ground-type term M, we have

$$(M\stackrel{push_j}{\longrightarrow})\stackrel{pop_k}{\longrightarrow} \;=\; \begin{cases} M\stackrel{pop_k}{\longrightarrow} & \text{if } j < k \\ M & \text{if } j = k \end{cases}$$

Lemma 1 *Suppose, for a fixed $2 \leq j \leq n$, M and N are $(n-j)$-similar ground-type terms.*

(i) If $j_1 > j$ then $M \xrightarrow{pop_{j_1}}$ and $N \xrightarrow{pop_{j_1}}$ are identical terms; if $j_1 < j$ then $M \xrightarrow{pop_{j_1}}$ and $N \xrightarrow{pop_{j_1}}$ are $(n-j)$-similar.

(ii) Let $\mathcal{F}_p^{a,e}$ be the head non-terminal of M and N with $e \geq 2$. If $e > j$ then $M \xrightarrow{coll.}$ and $N \xrightarrow{coll.}$ are identical terms; and if $e < j$ then $M \xrightarrow{coll.}$ and $N \xrightarrow{coll.}$ are $(n-j)$-similar.

Proof Straightforward consequences of the definitions of the transition relation. □

For example take a head normal ground-type term M with $2 \leq j \leq n$; we have M and $M \xrightarrow{push_j}$ are $(n-j)$-similar.

4.2 Correctness of the representation

The position of a given stack symbol in an n-stack s can be described by a sequence of pop operations that are needed to "collapse" the stack up to the point where that position becomes the top_1-element. For example, the position of b in the 2-stack (the top of stack is at the right-hand end) $[\,[\,\bot\, a\, a\,]\,[\,\bot\, a\, b\, a\,]\,[\,\bot\, a\, a\,]\,[\,\bot\, a\,]\,]$ is $pop_2^2 \,;\, pop_1$. In general such sequences are not unique, though they can be normalized to one in which the respective orders of the pop operations form a non-increasing sequence. (Equivalently A sequence is normalized if it is inequivalent to any sequence of shorter length.) We shall call a normalized sequence for a given stack s an *s-probe*. Informally we say that a ground-type term M *represents* a configuration (p, s) if for every s-probe α if the top_1-element of $\alpha\, s$ is $a^{(j,k)}$, then the head non-terminal of $M \xrightarrow{\alpha}$ is $\mathcal{F}_p^{a,j}$; further $(M \xrightarrow{\alpha}) \xrightarrow{pop_j^k} = (M \xrightarrow{\alpha}) \xrightarrow{coll.}$, and it represents the configuration $(p, collapse(\alpha\, s))$.

Definition 4.2 Let s be an n-stack. A sequence α of stack operations of the shape

$$pop_{j_1}^{k_1} \,;\, \cdots \,;\, pop_{j_l}^{k_l}$$

such that $l \geq 0$, and $j_1 > \cdots > j_l \geq 1$, and each $k_i \geq 1$, is said to be an *s-**probe*** just in case $\alpha\, s$ is defined.

(i) We say that a ground-type term M ***represents*** the configuration (p, s) if for every s-probe α, we have M *α-matches* (p, s).

(ii) We say that M *α-**matches*** the configuration (p, s) just if for some j and k, we have $top_1(\alpha\, s) = a^{(j,k)}$ and there exist m-tuples $\overline{L_{n-j}}, \overline{N_{n-1}}, \cdots, \overline{N_0}$ of the required types such that

$$M \;\xrightarrow{\alpha}\; \mathcal{F}_p^{a,j}\, \overline{L_{n-j}}\, \overline{N_{n-1}} \cdots \overline{N_0};$$

further if $a \neq \bot$ then

$$(M \xrightarrow{\alpha}) \xrightarrow{pop_j^k} \;=\; (M \xrightarrow{\alpha}) \xrightarrow{coll.}$$

and $(M \xrightarrow{\alpha}) \xrightarrow{pop_j^k}$ *represents* the configuration $(p, collapse(\alpha\, s))$.

Henceforth whenever we assert that a term α-matches a configuration (p, s), it has the force that α is an s-probe.

Note that $\mathcal{F}_p^{\bot,1}\, \overline{\Omega_{n-1}}\, \overline{\Omega_{n-1}} \cdots \overline{\Omega_{n-j}} \;:\; n-j$ represents the j-configuration $(p, \bot_{n-j})$. We have $top_1 \bot_n = \bot^{(1,1)}$, and id is the only $\bot_n$-probe.

Lemma 2 *(i) If M represents (p, s) then for every s-probe α, we have $(M\xrightarrow{\alpha})$ represents $(p, \alpha\, s)$.*

(ii) Let α be an s-probe. If $(M\xrightarrow{\alpha})$ represents $(p, \alpha\, s)$ then M α-matches (p, s).

Proof Straightforward consequences of the definitions. □

Having pinned down the notion of a term representing a configuration, we now show that the representation is preserved by the stack operations. This confirms that our notion of representation is the right one.

Theorem 3 (Correctness) *Let M be a ground-type term, (p, s) be a configuration, and θ be a stack operation. If M represents (p, s) and $M\xrightarrow{\theta}$ is defined, then $M\xrightarrow{\theta}$ represents the configuration $(p, \theta\, s)$.*

Proof Suppose $M = \mathcal{F}_p^{a,e}\,\overline{U}\,\overline{V_{n-1}}\cdots\overline{V_0}$ represents the configuration (p, s). We aim to prove that $M\xrightarrow{\theta}$ represents $(p, \theta s)$ for each of the four cases of the operation $\theta \in Op_n$.

Case 1: $\theta = push_1^{b,j_0}$ ***where*** b ***is a non-***$\perp$ ***symbol and*** $1 \le j_0 \le n$.

We have $M \xrightarrow{\theta} \mathcal{F}_p^{b,j_0}\,\overline{V_{n-j_0}}\,\langle\mathcal{F}_i^{a,e}\,\overline{U}\,\overline{V_{n-1}}\,|\,i\rangle\,\overline{V_{n-2}}\cdots\overline{V_0}$. Let $\alpha = pop_{j_1}^{k_1}\,;\cdots;\,pop_{j_l}^{k_l}$ be an $(\theta\, s)$-probe.

- Suppose $l = 0$ i.e. α is the identity operation id. We need to show that $(M\xrightarrow{\theta})$ id-matches $(p, \theta s)$. Now $top_1\,(\theta\, s) = b^{(j_0,1)}$; assuming $b \neq \perp$, we check that

$$(M\xrightarrow{\theta})\xrightarrow{pop_{j_0}} \;=\; (M\xrightarrow{\theta})\xrightarrow{coll.} \;=\; V_{n-j_0,p}\,\overline{V_{n-j_0-1}}\cdots\overline{V_0}.$$

  It remains to show that $(M\xrightarrow{\theta})\xrightarrow{pop_{j_0}}$ represents the configuration $(p, collapse(id\,(\theta\, s))) = (p, pop_{j_0}\,(\theta\, s))$, which is equivalent to

$$\begin{cases} M \text{ represents } (p, s) & \text{if } j_0 = 1 \\ (M\xrightarrow{pop_{j_0}}) \text{ represents } (p, pop_{j_0}\, s) & \text{if } j_0 > 1 \end{cases}$$

  Both cases follow from the assumption that M represents (p, s): the former because $(M\xrightarrow{push_1^b j_0})\xrightarrow{j_0} = M$, and the latter because of Lemma 2.

- Suppose $l \ge 1$ and $j_1 \ge 2$. We have $(M\xrightarrow{\theta})$ α-matches $(p, \theta s)$ iff M α-matches (p, s), which follows from the assumption.

- Suppose $j_1 = 1$. It follows that $l = 1$ and $\alpha = pop_1^{k_1}$ where $k_1 \ge 1$. Set $\alpha' = pop_1^{k_1-1}$ (we shall assume that $pop_1^0 = id$). Plainly $(M\xrightarrow{\theta})$ α-matches $(p, \theta s)$ iff M α'-matches (p, s), which is an immediate consequence of the assumption.

Case 2: $\theta = push_j$ ***where*** $2 \le j \le n$.

We have $M \xrightarrow{\theta} \mathcal{F}_p^{a,e}\,\overline{U}\,\overline{V_{n-1}}\cdots\langle\mathcal{F}_i^{a,e}\,\overline{U}\,\overline{V_{n-1}}\cdots\overline{V_{n-j}}\,|\,i\rangle\cdots\overline{V_0}$. Let $\alpha = pop_{j_1}^{k_1}\,;\cdots;\,pop_{j_l}^{k_l}$ be an $(\theta\, s)$-probe. There are two subcases:

a. The probe α reaches into $pop_j\,(\theta\, s) = S$; i.e. $l \ge 1$ and $j_1 \ge j$.

b. The probe α is confined to $top_j\,(\theta\, s)$; i.e. either $l \ge 1$ and $j_1 < j$, or $l = 0$.

Case 2a: Since

$$(M \xrightarrow{\theta})\xrightarrow{pop_{j_1}} \;=\; \begin{cases} M\xrightarrow{pop_{j_1}} & \text{if } j < j_1 \\ M & \text{if } j = j_1 \end{cases}$$

and correspondingly

$$(p, pop_{j_1}\,(\theta\, s)) \;=\; \begin{cases} (p, pop_{j_1}\, s) & \text{if } j < j_1 \\ (p, s) & \text{if } j = j_1 \end{cases}$$

we have $(M\xrightarrow{\theta})$ α-matches $(p, \theta\, s)$ as required.

Case 2b: Let $\mathbf{b}$ be the set of $(\theta\, s)$-probes of this case. We define a partial order $<$ over $\mathbf{b}$-probes by $\alpha_1 < \alpha_2$ just if there is a non-empty sequence of pop-operations such that $\beta(\alpha_2(\theta\, s)) = \alpha_1(\theta\, s)$; in other words $\alpha_1(\theta\, s)$ is a prefix of $\alpha_2(\theta\, s)$. We shall show, by an induction argument, that $(M\xrightarrow{\theta})\xrightarrow{\alpha}$ represents $(p, \alpha\,(\theta\, s))$ for every $\mathbf{b}$-probe α; note that this implies $(M\xrightarrow{\theta})$ α-matches $(p, \theta\, s)$ as required. The base case is trivial. Suppose for all $\mathbf{b}$-probes $\alpha < \alpha_0$, we have $(M\xrightarrow{\theta})\xrightarrow{\alpha}$ represents $(p, \alpha\,(\theta\, s))$. We aim to show that $(M\xrightarrow{\theta})\xrightarrow{\alpha_0}$ represents $(p, \alpha_0\,(\theta\, s))$. Take any $\alpha_0\,(\theta\, s)$-probe β; we want to show that $((M\xrightarrow{\theta})\xrightarrow{\alpha_0})$ β-matches $(p, \alpha_0(\theta\, s))$. Now α_0 ; β is equivalent to a $(\theta\, s)$-probe of either case a or case b. The former case has already been dealt with. In the latter case, either α_0 ; $\beta < \alpha_0$ or $\beta = id$. If α_0 ; $\beta < \alpha_0$, then by the induction hypothesis $((M\xrightarrow{\theta})\xrightarrow{\alpha_0})\xrightarrow{\beta}$ represents $(p, \beta(\alpha_0(\theta\, s)))$, which implies that $(M\xrightarrow{\theta})\xrightarrow{\alpha_0}$ β-matches $(p, \alpha_0(\theta\, s))$ as desired. It remains to prove $((M\xrightarrow{\theta})\xrightarrow{\alpha_0})$ id-matches $(p, \alpha_0(\theta\, s))$. Suppose the top_1-element of $\alpha_0(\theta\, s)$ is $b^{(e', f')}$. Since $top_j\,(\theta\, s) = top_j\, S$, the top_1-element of $\alpha_0\, S$ is $b^{(e', g)}$ where

$$g \;=\; \begin{cases} f' & \text{if } e' \neq j \\ f' - 1 & \text{if } e' = j \end{cases}$$

(Note that in case $e' = j$, by definition of $push_j$ we have $f' \geq 2$.) Since M represents (p, s) by assumption, it follows that the head non-terminal of $M\xrightarrow{\alpha_0}$ is $\mathcal{F}_p^{b,e'}$. Now $M\xrightarrow{\theta}$ and M are $(n-j)$-similar. Since α_0 is a $\mathbf{b}$-probe, by applying Lemma 1 repeatedly, we have $(M\xrightarrow{\theta})\xrightarrow{\alpha_0}$ and $M\xrightarrow{\alpha_0}$ are $(n-j)$-similar. Hence the head non-terminal of $(M\xrightarrow{\theta})\xrightarrow{\alpha_0}$ is also $\mathcal{F}_p^{b,e'}$. It remains to prove that, assuming $b \neq \perp$, we have

$$((M\xrightarrow{\theta})\xrightarrow{\alpha_0})\xrightarrow{pop_{e'}^{f'}} \quad = \quad ((M\xrightarrow{\theta})\xrightarrow{\alpha_0})\xrightarrow{coll.} \tag{1}$$

and

$$((M\xrightarrow{\theta})\xrightarrow{\alpha_0})\xrightarrow{pop_{e'}^{f'}} \text{ represents } (p, collapse(\alpha_0(\theta\, s))). \tag{2}$$

But since the top_1-element of $\alpha_0(\theta\, s)$ is $b^{(e', f')}$, we have

$$(p, pop_{e'}^{f'}(\alpha_0(\theta\, s))) \quad = \quad (p, collapse(\alpha_0(\theta\, s)));$$

further, by the induction hypothesis (in case $e' < j$) or the assumption of M representing (p, s) (in case $e' \geq j$) as appropriate, we have $((M\xrightarrow{\theta})\xrightarrow{\alpha_0})\xrightarrow{pop_{e'}^{f'}}$ represents $(p, pop_{e'}^{f'}(\alpha_0(\theta\, s)))$; hence (2) holds. It now remains to prove (1). We consider the three cases of e' in turn:

i. *Case of* $e' < j$: Since M and $M\xrightarrow{\theta}$ are $(n-j)$-similar, by applying Lemma 1 repeatedly, we have

$$\begin{array}{lll} ((M\xrightarrow{\theta})\xrightarrow{\alpha_0})\xrightarrow{pop_{e'}^{f'}} & \sim_{n-j} & (M\xrightarrow{\alpha_0})\xrightarrow{pop_{e'}^{f'}} \\ ((M\xrightarrow{\theta})\xrightarrow{\alpha_0})\xrightarrow{coll.} & \sim_{n-j} & (M\xrightarrow{\alpha_0})\xrightarrow{coll.} \end{array}$$

Since $M \xrightarrow{\alpha_0}$ represents $(p, \alpha_0\, s)$ and $\alpha_0\, S$ has top_1-element $b^{(e',f')}$, we have $(M \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} = (M \xrightarrow{\alpha_0}) \xrightarrow{coll.}$. Thus

$$((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} \quad \sim_{n-j} \quad ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{coll.}.$$

But observe that $((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}}$ and $((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{coll.}$ have the same $(n-j)$-factor. Hence

$$((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} \;=\; ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{coll.}$$

as required.

ii. *Case of* $e' = j$: By definition of $push_j$, we have $top_1\,(\alpha_0\, s) = b^{(e',f'-1)}$ and $f' \geq 2$. Since M represents (p, s) by assumption, we have M α_0-matches (p, s). It follows that

$$(M \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'-1}} \quad = \quad (M \xrightarrow{\alpha_0}) \xrightarrow{coll.}. \tag{3}$$

Now, as a consequence of Lemma 1, for some vectors $\overline{P}, \overline{N_{n-1}}, \cdots, \overline{N_{n-j+1}}$ of terms of the appropriate types, we have

$$\begin{cases} (M \xrightarrow{\theta}) & \xrightarrow{\alpha_0} & \mathcal{F}_p^{b,e'}\, \overline{P}\, \overline{N_{n-1}} \cdots \overline{N_{n-j+1}}\, \langle \mathcal{F}_i^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_{n-j}} \mid i \rangle\, \overline{V_{n-j-1}} \cdots \overline{V_0} \\ M & \xrightarrow{\alpha_0} & \mathcal{F}_p^{b,e'}\, \overline{P}\, \overline{N_{n-1}} \cdots \overline{N_{n-j+1}}\, \overline{V_{n-j}}\, \overline{V_{n-j-1}} \cdots \overline{V_0} \end{cases} \tag{4}$$

and hence we have

$$(M \xrightarrow{\alpha_0}) \xrightarrow{coll.} \quad = \quad ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{coll.}. \tag{5}$$

In view of (3) and (5), in order to prove (1), it suffices to prove

$$(M \xrightarrow{\alpha_0}) \xrightarrow{pop_j^{f'-1}} \quad = \quad ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_j^{f'}}. \tag{6}$$

But it follows from (4) that

$$((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_j} \quad = \quad \mathcal{F}_p^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_{n-j}}\, \overline{V_{n-j-1}} \cdots \overline{V_0}$$

and so, since $f' \geq 2$, we have $((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} = ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{coll.}$ as required.

iii. *Case of* $e' > j$: By applying Lemma 1 repeatedly, we have

$$\begin{aligned} ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} &= (M \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} \\ ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{coll.} &= (M \xrightarrow{\alpha_0}) \xrightarrow{coll.} \end{aligned}$$

Since $(M \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} = (M \xrightarrow{\alpha_0}) \xrightarrow{coll.}$ we have $((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{pop_{e'}^{f'}} = ((M \xrightarrow{\theta}) \xrightarrow{\alpha_0}) \xrightarrow{coll.}$ as required.

Case 3: $\theta = pop_k$ ***where*** $1 \leq k \leq n$.

This is an immediate consequence of Lemma 2.

Case 4: $\theta = collapse$.

Suppose $top_1\, S = a^{(e,k)}$ and $a \neq \perp$ so that $\theta\, s = pop_e^k\, s$; we have $M \xrightarrow{\theta} U_p\, \overline{V_{n-e-1}} \cdots \overline{V_0}$. By assumption, we have M id-matches (p, s). It follows that $(M \xrightarrow{id}) \xrightarrow{pop_e^k} U_p\, \overline{V_{n-e-1}} \cdots \overline{V_0}$; further $(M \xrightarrow{id}) \xrightarrow{pop_e^k}$, which equals $M \xrightarrow{\theta}$, represents the configuration $(p, collapse(ids)) = (p, \theta\, s)$, as required. □

4.3 The recursion scheme $G_{\mathcal{A}}$ determined by a CPDA $\mathcal{A}$

Definition 4.3 Fix a tree-generating order-n CPDA $\mathcal{A} = \langle \Sigma, \Gamma, Q, \delta, q_0 \rangle$ with $Q = [m]$ for some $m \geq 1$, and $q_0 = 1$. The order-n ***recursion scheme determined by*** $\mathcal{A}$, written $G_{\mathcal{A}}$, is defined by the following rewrite rules. There are two kinds of rewrite rules, corresponding to the labels ***I*** and ***P***:

I. For each $(p, a, q, \theta) \in \delta$ and $1 \leq e \leq n$, we have the rule:

$$\mathcal{F}_p^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0} \quad \overset{(q,\theta)}{\longrightarrow} \quad \Xi_{(q,\theta)}$$

where $\Xi_{(q,\theta)}$ is as given in Table 1.

P. For each $(p, a, (f; q_1, \cdots, q_{ar(f)})) \in \delta$ and $1 \leq e \leq n$, we have the rule:

$$\mathcal{F}_p^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0} \quad \overset{(f,\bar{q})}{\longrightarrow} \quad f\,(\mathcal{F}_{q_1}^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0})\,\cdots\,(\mathcal{F}_{q_{ar(f)}}^{a,e}\,\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0}).$$

Finally there is the *start rule*: $S \longrightarrow \mathcal{F}_1^{\perp,1}\overline{\Omega_{n-1}}\,\overline{\Omega_{n-1}}\cdots\overline{\Omega_0}$. We write $\longrightarrow \subseteq \mathcal{T}^0(\Sigma \cup \mathcal{N}) \times \mathcal{T}^0(\Sigma \cup \mathcal{N})$ for the one-step reduction relation[11] between ground-type applicative terms (or *ground terms*, for short), defined to be the substitutive and contextual closure of the rewrite rules. We write $\longrightarrow^*$ as the reflexive, transitive closure of $\longrightarrow$.

A ground term R is called a *redex* if for some term R' we have $R \longrightarrow R'$ is a *substitutive* instance of a rewrite rule of label ℓ (say), and the redex is said to be ***P**-type* or ***I**-type* according to the type of ℓ; by abuse of notation, we shall write $R \overset{\ell}{\longrightarrow} R'$. A ground term is either *head non-terminal* (i.e. the head symbol is a non-terminal) or *head terminal* (i.e. of the shape $f\,N_1 \cdots N_{ar(f)}$). A head non-terminal ground term is either atomic (i.e. S or Ω_0) or it is *head normal* (i.e. the head symbol is of the form $\mathcal{F}_p^{a,e}$, in which case, the ground term is an ***I***-type or ***P***-type redex).

It follows from the definition that for every one-step reduction $M \longrightarrow M'$, there are a unique redex R with $R \overset{\ell}{\longrightarrow} R'$ and a unique *active context*[12] E such that $M = E[R]$ and $M' = E[R']$. To indicate the *active redex* (and its occurrence in the term), we shall sometimes write M as (E, R), and write the one-step reduction as $(E, R) \overset{\ell}{\longrightarrow} (E, R')$, which we shall call ***P**-type* or ***I**-type* according to the type of the label ℓ. We shall call expressions of the shape (E, R) an *active term*; in general the ground-type term R may be a redex or a head-terminal term.

Path reduction sequences

Path reduction sequences are finite or infinite sequences of one-step reductions starting from the start symbol S. In essence, they constitute a reduction strategy for computing paths in the tree generated by the recursion scheme $G_{\mathcal{A}}$. Given a path in $[\![\, G_{\mathcal{A}} \,]\!]$ as specified by a word in the branch language (e.g. $(f,2)\,(g,1)\,a$) the path reduction strategy (that computes it) begins rewriting from S in a sequence of *macro steps*; each macro step computes an element (i.e. a Σ-symbol and a direction) of the word. Take the word $(f,2)\,(g,1)\,a$. The path reduction sequence begins from S by rewriting the leftmost redex until it reaches a head terminal term $f\,N_1\,\cdots\,N_{ar(f)}$, upon which N_2 is selected, corresponding to the pair $(f,2)$; the next macro step then starts

[11] When defining $\longrightarrow$ and the tree generated by the recursion scheme $G_{\mathcal{A}}$, we simply ignore the labels ℓ that annotate the rewrite rules $\overset{\ell}{\longrightarrow}$.

[12] An *active context* is just an ground-type applicative term that contains a ground-typed hole, into which a term may be inserted.

from N_2 by rewriting the leftmost redex until it reaches a head terminal term $g\,P_1 \cdots P_{ar(g)}$, whereupon P_1 is selected, corresponding to $(g, 1)$; the (terminal) macro step then starts from P_1 by rewriting the leftmost redex until it reaches a, which is a terminal node of the tree.

A *path reduction sequence* is a finite or infinite sequence of *macro steps*. Formally a ***macro step*** is a finite or infinite sequence of one-step reductions (of active terms) organized into (up to) three stages, starting from a given active term (E, R_1); if it terminates, the macro step returns an active term (E', N) such that $(E, R_1) \longrightarrow^* (E', N)$. The three stages are as follows:

1. A finite (possibly empty) or infinite sequence of ***I**-type* one-step reductions

   $$(E, R_1) \stackrel{(q_1,\theta_1)}{\longrightarrow} (E, R_2) \stackrel{(q_2,\theta_2)}{\longrightarrow} (E, R_3) \stackrel{(q_3,\theta_3)}{\longrightarrow} (E, R_4) \cdots$$

   for some active context E. The sequence terminates (at (E, R_{r+1})) iff for some $r \geq 0$, we have R_{r+1} is not an ***I***-type redex (i.e. R_{r+1} is a ***P***-type redex). If the ***I***-type reduction sequence does not terminate, the macro step is said to be *partial*.

2. A ***P**-type* one-step reduction $(E, R_{r+1}) \stackrel{(f;\overline{q})}{\longrightarrow} (E, f\,N_1 \cdots N_{ar(f)})$ where $\overline{q} = q_1 \cdots q_{ar(f)}$.

3. A ***O**-type* one-step reduction: The *head-terminal* active term $(E, f\,N_1 \cdots N_{ar(f)})$ is transformed to (E', N_i) for some $1 \leq i \leq ar(f)$, where

   $$E' \;=\; E[f\,N_1 \cdots N_{i-1}\,[\text{-}]\,N_{i+1} \cdots N_{ar(f)}]$$

   if $ar(f) \geq 1$ (in which case, by abuse of notation, we write $(E, f\,N_1 \cdots N_{ar(f)}) \stackrel{(f,i)}{\longrightarrow} (E', N_i)$, even though $E[f\,N_1 \cdots N_{ar(f)}]$ and $E'[N_i]$ are identical terms); otherwise (i.e. f is nullary), there is no transformation, and the macro step in question is the terminal step of the path reduction sequence.

Thus a typical macro step is a sequence of one-step reductions that may have the following shape:

$$(E, R_1) \stackrel{(q_1,\theta_1)}{\longrightarrow} \cdots \stackrel{(q_r,\theta_r)}{\longrightarrow} (E, R_{r+1}) \stackrel{(f;\overline{q})}{\longrightarrow} (E, f\,N_1 \cdots N_{ar(f)}) \stackrel{(f,i)}{\longrightarrow} (E', N_i)$$

with $E' = E[f\,N_1 \cdots N_{i-1}\,[\text{-}]\,N_{i+1} \cdots N_{ar(f)}]$ and $r \geq 0$. There are two other possibilities, namely, $ar(f) = 0$ (in which case the macro step in question is the final step), or the macro step may consist of an infinite ***I***-type reduction (in which case it is a partial step).

Proposition 4 *The path reduction sequences compute all maximal traces of the Σ-labelled tree generated by $G_{\mathcal{A}}$.*

Proof Take a macro step starting from a given decomposed term (E, R); stages 1 and 2 are completely determined; and stage 3 is specified by an ***O***-type label (f, i), in case $ar(f) \geq 1$, with $1 \leq i \leq ar(f)$. Thus a (maximal) path reduction sequence starting from S can be specified by

- an infinite sequence of ***O***-type labels, or
- a finite sequence of ***O***-type labels (ending in an partial macro step), or
- a finite sequence of ***O***-type labels ending in a terminal macro step (specified by a nullary Σ-symbol).

□

We are now in a position to state the second major result of the section.

Theorem 5 (Equi-Expressivity 1) *Let $\mathcal{A}$ be a tree-generating CPDA, and let $G_{\mathcal{A}}$ be the recursion scheme determined by $\mathcal{A}$. Then the CPDA and the recursion scheme generate the same Σ-labelled tree.*

Proof Because of Proposition 4, it suffices to prove

> *Claim*: For any (finite or infinite) computation path of $\mathcal{A}$ of the form $\gamma_0 \overset{\ell_0}{>} \gamma_1 \overset{\ell_1}{>} \gamma_2 \overset{\ell_2}{>} \cdots$, there is a unique path reduction sequence $S \longrightarrow (E_0, R_0) \xrightarrow{\ell_0} (E_1, R_1) \xrightarrow{\ell_1} (E_2, R_2) \xrightarrow{\ell_2} \cdots$ with $E_0 = [\text{-}]$ such that for every $i \geq 0$, if R_i is head-normal, then R_i represents γ_i. The converse also holds.

To see why the claim is true, suppose $R_i = \mathcal{F}_p^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_0}$ and R_i represents $\gamma_i = (p, s)$. We consider the various types of the label ℓ_i in turn. First, thanks to Theorem 3, we have $(p, s) \overset{(q,\theta)}{>} \gamma_{i+1}$ iff $R_i \xrightarrow{(q,\theta)} R_{i+1}$; and if either side of the preceding bi-implication holds, then R_{i+1} represents γ_{i+1}, and we have $E_i = E_{i+1}$. Secondly, it follows from Definition 4.3 that $\gamma_i \overset{(f;\overline{q})}{>} (f; q_1, \cdots, q_{ar(f)}; s) = \gamma_{i+1}$ iff

$$\underbrace{\mathcal{F}_p^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_0}}_{R_i} \quad \xrightarrow{(f;\overline{q})} \quad \underbrace{f\, (\mathcal{F}_{q_1}^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_0}) \cdots (\mathcal{F}_{q_{ar(f)}}^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_0})}_{R_{i+1}};$$

further if either side of the preceding bi-implication holds, and if $ar(f) \geq 1$, then for some $1 \leq j \leq ar(f)$, we have $(f; q_1, \cdots, q_{ar(f)}; s) \overset{(f;j)}{>} (q_j, s) = \gamma_{i+2}$ and $R_{i+2} = \mathcal{F}_{q_j}^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_0}$ represents γ_{i+2}, with $E_{i+2} = E_{i+1}[f\, (\mathcal{F}_{q_1}^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_0}) \cdots [\text{-}] \cdots (\mathcal{F}_{q_{ar(f)}}^{a,e}\, \overline{U}\, \overline{V_{n-1}} \cdots \overline{V_0})]$ as required; and this concludes the proof. □

5 From recursion schemes to CPDA

The previous section demonstrates that higher-order recursion schemes are at least as expressive as CPDAs. In this section we shall sketch a proof of the converse. Hence, CPDAs and recursion schemes are in fact equi-expressive. A number of related results can be found in the literature, but an exact correspondence with general recursion schemes has never been proved before. Notably, in order to establish a correspondence between recursion schemes and HOPDAs, Damm and Goerdt (for word languages [8, 9]) as well as Knapik, Niwiński and Urzyczyn (for labelled trees [15]), have had to impose constraints on the shape of the former (called *derived types* and *safety* respectively) and their translation techniques relied on the restrictions in a crucial way.

Our translation from recursion schemes to CPDA is novel: we transform an arbitrary order-n recursion scheme G to an order-n *collapsible pushdown automaton* $\mathcal{A}_G$ that computes the *traversals* over the computation tree $\lambda(G)$ (in the sense of Ong [20, 21]). The game-semantic interpretation of G is an *innocent strategy* (in the sense of Hyland and Ong [13]), which coincides with the *value tree* $[\![\, G \,]\!]$ of G, so that paths in the value tree are plays of the strategy. Traversals over the computation tree are just (appropriate representations of) *uncoverings* [13] of paths in the value tree.

5.1 $\mathsf{CPDA}(G)$ - the CPDA determined by a recursion scheme G

Fix an order-n recursion scheme G and the HORS graph

$$\mathsf{Gr}(G) \quad = \quad \langle\, V,\ E \subseteq V \times V,\ \lambda_G : V \longrightarrow \Lambda_G,\ v_0 \in V \,\rangle$$

determined by it. Note that G is not assumed to be homogeneously typed, and hence, not necessarily safe. We shall construct an order-n collapsible pushdown automaton, written $\mathsf{CPDA}(G)$, that computes *traversals* over the computation tree $\lambda(G)$. Thus it is an automaton that computes the innocent strategy $[\![G]\!]$ given by the value tree of G. (Traversals and computation trees are introduced in Ong's preprint [20, 21].)

Remark 5.1 For convenience, in the definition of the transform $\mathsf{CPDA}(G)$, we shall write $push_1^{a,1}$ as $push_1^a$, effectively ignoring the 1-link (to the preceding stack symbol). This is harmless since 1-links are guaranteed not to feature in any of collapse operations in of the transform $\mathsf{CPDA}(G)$.

Definition 5.2 The transform $\mathsf{CPDA}(G)$ is an n-CPDA that has the set V of nodes as the stack alphabet. The initial configuration is the n-stack $[\,\cdots[\,\bot\, v_0\,]\cdots\,]$ i.e. $push_1^{v_0}\,\bot_n$, where v_0 is the root of $\mathsf{Gr}(G)$. Let u range over the stack symbols of $\mathsf{CPDA}(G)$. For ease of explanation, we define the transition map δ as a function that takes a node $u \in V$ to a sequence of stack operations, by a case analysis of the label (from Λ_G) of u. The definition is presented in Figure 2. In the Figure (and henceforth), we shall write $push_1^a$ for $push_1^{a,1}$ for simplicity.

If u's label is not a variable, the action is just a $push_1^v$, where v is an appropriate child of the node u. Precisely:

(A) If the label is an @ then $\delta(u) = push_1^{E_0(u)}$.

(S) If the label is a Σ-symbol f then $\delta(u) = push_1^{E_i(u)}$, where $1 \le i \le ar(f)$ is the direction requested by the Environment, or Opponent.

Note that if f is nullary, the automaton terminates (since Opponent has no move to make).

(L) If the label is a lambda then $\delta(u) = push_1^{E_1(u)}$.

Suppose u is labelled by a variable which is the i-parameter of the lambda node $binder(u)$; and suppose $binder(u)$ is a j-child. Let p be the span of the variable node u.

($\mathbf{V}_1$) If the variable has order $l \ge 1$, then

$$\delta(u) \quad = \quad \begin{cases} push_{n-l+1}\,;\, pop_1^{p+1}\,;\, push_1^{E_i(top_1),n-l+1} & \text{if } j = 0 \\ push_{n-l+1}\,;\, pop_1^{p}\,;\, collapse\,;\, push_1^{E_i(top_1),n-l+1} & \text{otherwise} \end{cases}$$

where pop_1^p means the operation pop_1 iterated p times, and $push_1^{E_i(top_1),k}$ is defined to be the operation $s \mapsto push_1^{E_i(top_1\, s),k}\, s$.

($\mathbf{V}_0$) Otherwise (i.e. the variable has order 0)

$$\delta(u) \;=\; \begin{cases} pop_1^{p+1}\,;\, push_1^{E_i(top_1)} & \text{if } j = 0 \\ pop_1^{p}\,;\, collapse\,;\, push_1^{E_i(top_1)} & \text{otherwise.} \end{cases}$$

Figure 2: Definition of the transform $\mathsf{CPDA}(G)$.

Let s and s' range over configurations (i.e. n-stack contents) of $\mathsf{CPDA}(G)$. We define a binary relation $\to$ over configurations: we say that $s \to s'$ just if $s' = \delta(top_1\, s)(s)$. We write $\to^*$ for the reflexive, transitive

closure of $\rightarrow$.

Remark 5.3 (i) The definition of $\mathsf{CPDA}(G)$ as presented is not formally an instance of a CPDA: the transition function maps stack symbols to composites of stack actions, based on a case analysis. However it can be transformed to a proper CPDA, and in general the equivalent CPDA has more than one control states.

(ii) The transformation is radically different from the compilation method of Knapik *et al.* [15, 16]. To date, it is not known whether the approach in [16] is extendable to non-homogeneously typed recursion schemes of order 2. More generally, it is not known whether the method is extendable to arbitrary recursion schemes of all finite orders.

Question. Are order-n CPDA equi-expressive (for generating trees) with order-$(n+1)$ PDA?

How does $\mathsf{CPDA}(G)$ work?

The transform $\mathsf{CPDA}(G)$ computes traversals over the computation tree $\lambda(G)$. Take a computation sequence of traversals[13]:

$$t_1 \ > \ t_2 \ > \ t_3 \ > \cdots$$

starting from the singleton traversal t_1 consisting of the root of the graph $\mathsf{Gr}(G)$ (so that for each i, the traversal t_{i+1} extends t_i by one node). Let

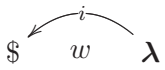
$$s_1 \ \rightarrow \ s_2 \ \rightarrow \ s_3 \ \rightarrow \ \cdots$$

be the corresponding computation sequence of $\mathsf{CPDA}(G)$ starting from the initial configuration s_1.

The two computation sequences are closely related in a lock-step fashion. First we shall see that for each $i \geq 1$, the top 1-stack of s_i (regarded as a sequence of nodes) is the P-view of t_i i.e.

$$top_2\, s_i \quad = \quad \ulcorner t_i \urcorner.$$

Secondly we construct a kind of approximant of t_i, written $\widehat{t_i}$, which is obtained from t_i by removing all segments w sandwiched between matching pairs of the shape



where \$ is either an order-1 variable or an @-symbol, and $i \geq 1$. Note that by definition of traversal, the segment w necessarily has the shape

where x is an order-0 variable symbol and $\overline{\varphi}$ is a list of variables in which x occurs. We then transform each n-stack s_i to a sequence of nodes $\underline{s_i}$, which will be shown to coincide with $\widehat{t_i}$.

[13] Here traversals are justified sequences of nodes of the graph $\mathsf{Gr}(G)$, as opposed to nodes of the computation tree $\lambda(G)$ which is the unfolding of $\mathsf{Gr}(G)$. The latter is the notion defined in the preprint [21], but the difference is of no significance for the purpose we have in mind.

How to construct the sequence $\underline{s_i}$ from an n-stack s_i?

We follow a simple recipe.

1. We "flatten" the n-stack s_i so that it has the form of a well-bracketed sequence such as the following (top of stack is the right-hand end)

$$[[[\cdots]\cdots[\cdots]][[\cdots]]\cdots[[\cdots][\cdots]]]$$

2. The target of any pointer to a stack is deemed to be the rightmost symbol representing the stack, i.e. it is always an occurrence of].

3. The required subsequence – which we shall write as $\underline{s_i}$ – is obtained by a right-to-left scan of the well-bracketed sequence above according to the following rules.

   - When an occurrence of] is encountered, we simply continue the scan without recording].
   - We record any stack symbols that are being scanned.
   - Whenever we encounter the source of a link, the scan jumps to its target (an occurrence of]) without recording any nodes sandwiched in-between. The source of the link is always recorded.
   - The scan ends when the first [is hit.

   Note that the last condition is necessary to ensure that $\underline{s}$ is suitably defined for any prefix of a reachable stack. This will be important in the proof of Proposition 7.

Here is a more formal definition.

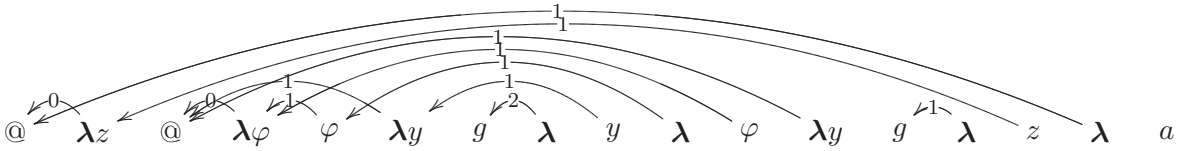
Definition 5.4 Let s be an n-stack. The sequence $\underline{s}$ of stack symbols is defined as follows.

$$\underline{s} = \begin{cases} \varepsilon & top_2 s = [\,], \\ \underline{(pop_1 s)}\, u & top_1 s = u \text{ and } u \text{ does not have a link}, \\ \underline{(collapse\, s)}\, u & top_1 s = u \text{ and } u \text{ has a link}. \end{cases}$$

Examples

We illustrate the workings of $\mathsf{CPDA}(G)$ by examples and point out the correspondence between runs of the automaton and the traversals it computes.

Example 5.5 Take the following traversal over the computation tree of G in Example 3.2:



We give a run of the corresponding 2-CPDA that computes the above traversal in Figure 3.

To save space, we only present the interesting configurations in which the top_1-element of the stack is a variable node. In the picture, the top of a stack is at the right-hand end, and links are represented by dotted arrows. Set t to be the prefix of the above traversal that ends in the node labelled by z. We have

$$\widehat{t} = \; @ \quad \lambda z \quad @ \quad \lambda\varphi \quad \varphi \quad \lambda \quad \varphi \quad \lambda y \quad g \quad \lambda \quad z$$

which coincides with the 2-stack $\underline{s}$ (see Figure 3) by following the recipe.

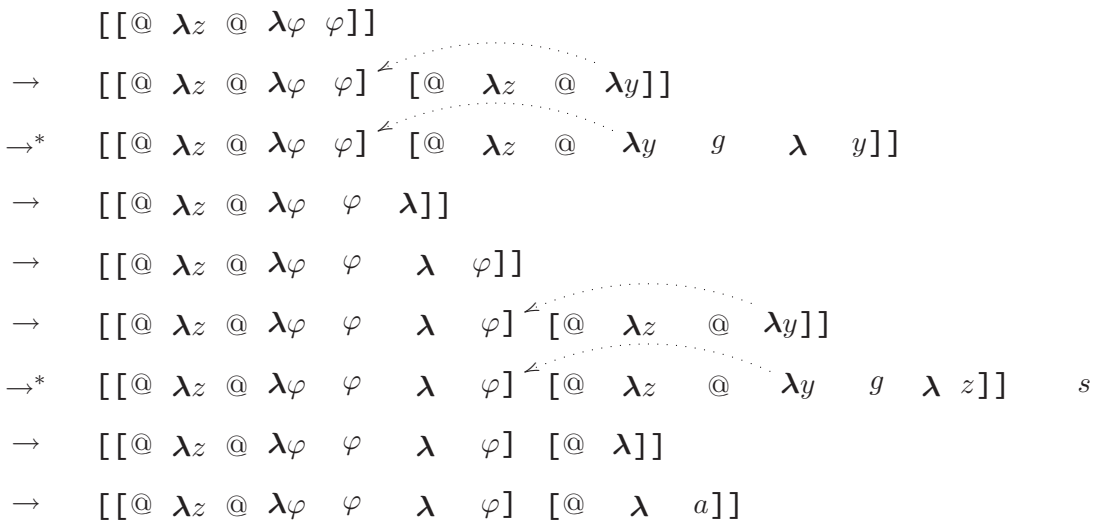


Figure 3: A run of a 2-CPDA

Example 5.6 Consider the order-3 HORS graph in Figure 5. For ease of reference, we give nodes numeric names, which are indicated (within square-brackets) as superscripts. Take the traversal $\mathbf{t}$ in Figure 4.

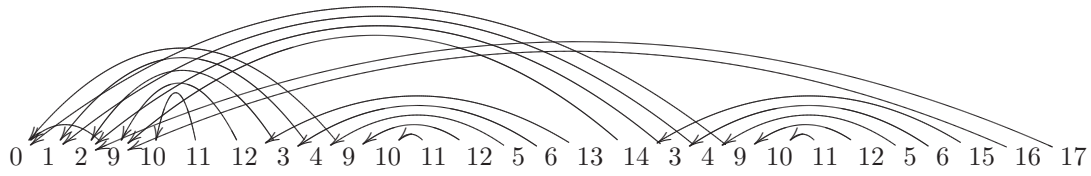


Figure 4: An order-3 traversal

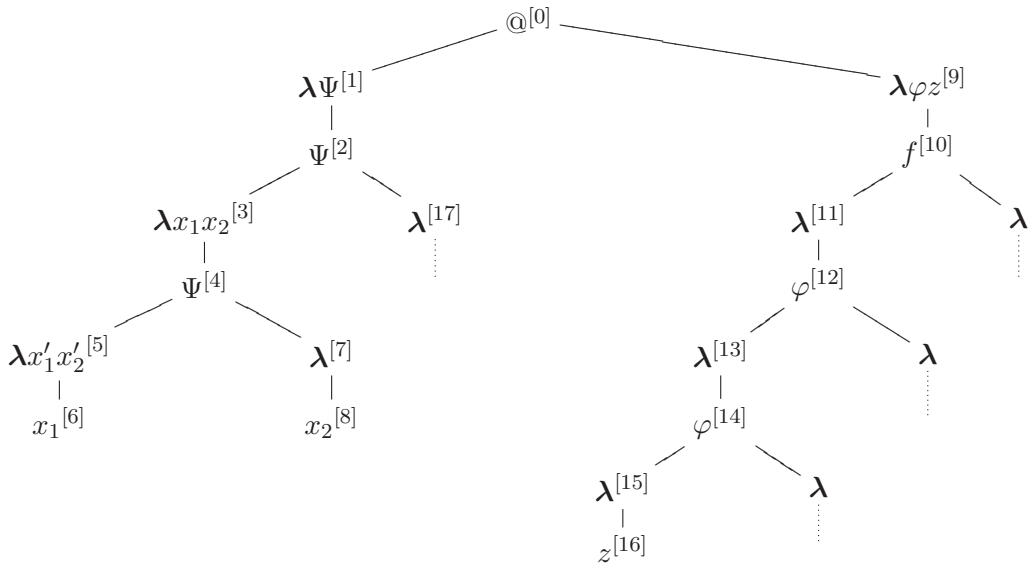


Figure 5: An example of an order-3 HORS graph.

We present a run of the 3-CPDA that computes the traversal $\mathbf{t}$ in Figure 6 followed by Figure 7 and Figure 8 (for ease of reading, we represent nodes by their labels).

To see the correspondence with the traversal $\mathbf{t}$, note that configurations s_2 and s_3 in Figures 7 and 8 respectively have the same top_1-element which is node 6 (labelled by x_1). They correspond respectively to the two prefixes of $\mathbf{t}$ that end in node 6.

The traversal t corresponding to s_3 is the prefix of $\mathbf{t}$ that ends in the later occurrence of 6; we have

$$\widehat{t} \quad = \quad @\ \boldsymbol{\lambda}\Psi\ \ \Psi\ \ \boldsymbol{\lambda}\varphi z\ \ f\ \ \boldsymbol{\lambda}\ \ \varphi\ \ \boldsymbol{\lambda}\ \ \varphi\ \ \boldsymbol{\lambda}x_1x_2\ \ \Psi\ \ \boldsymbol{\lambda}\varphi z\ \ f\ \ \boldsymbol{\lambda}\ \ \varphi\ \ \boldsymbol{\lambda}x_1'x_2'\ \ x_1$$

The reader might wish to check that $\widehat{t} = \underline{s_3}$. (Note that the justification pointers are uniquely reconstructible from the underlying sequence of nodes and their respective labels.)

5.2 Correctness of the transform

In this section we first set out (in Proposition 7), given any order-n recursion scheme G, the way in which the reachable configurations of the transform $\mathsf{CPDA}(G)$

$$s_1 \rightarrow s_2 \rightarrow \ \cdots\ \rightarrow s_n \rightarrow \cdots$$

can be said to compute traverals over the computation tree $\lambda(G)$ (i.e. the Λ_G-labelled tree obtained from $\mathsf{Gr}(G)$ by unfolding)

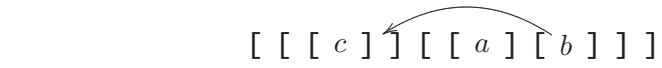
$$t_1 > t_2 > \ \cdots\ > t_n > \cdots$$

in a lock-step fashion. It then follows (from the Correspondence Theorem in [21]) that $\mathsf{CPDA}(G)$ computes (all paths in) the Σ-labelled tree $[\![\, G \,]\!]$ generated by G.

Lemma 6 *Let s be a reachable configuration of an n-CPDA, and let u be an occurrence of a stack symbol in s. Suppose u has a link to an i-stack σ. Then for every $i < j \leq n$, the i-stack σ is contained in the same j-stack in which u occurs.*

Proof Since the property is preserved by every n-CPDA instruction, the Lemma follows by induction on the number of instructions it takes to obtain s from the initial configuration. □

For example, the following 3-stack is not reachable: b has a link to the 1-stack $[\ c\ ]$, but the source (i.e. b) and the target (i.e. $[\ c\ ]$) of the link are *not* in the same 2-stack



Definition 5.7 Let G be an order-n recursion scheme, let s be a reachable configuration of $\mathsf{CPDA}(G)$, and let t be a traversal over $\lambda(G)$. We shall say that s ***computes*** t if and only if the following conditions hold.

(a) $top_2(s) = \ulcorner t \urcorner$.

(b) $\underline{s} = \widehat{t}$.

(c) Suppose $top_2(s) = [v_1, \cdots, v_n]$. Let $v_1', \cdots, v_n'$ be the respective occurrences of $v_1, \cdots, v_n$ in t that contribute to $\ulcorner t \urcorner$. Then $pop_1^{n-i}(s)$ computes $t_{\leq v_i'}$ for any $1 \leq i < n$.

(d) Using the same notation as in (c), suppose v_i has a link to an l-stack σ. Let s_σ be the prefix of s such that σ is its top l-stack, i.e. $s_\sigma = collapse(pop_1^{n-i}\, s)$. Then s_σ computes $t_{<v_i'}$.

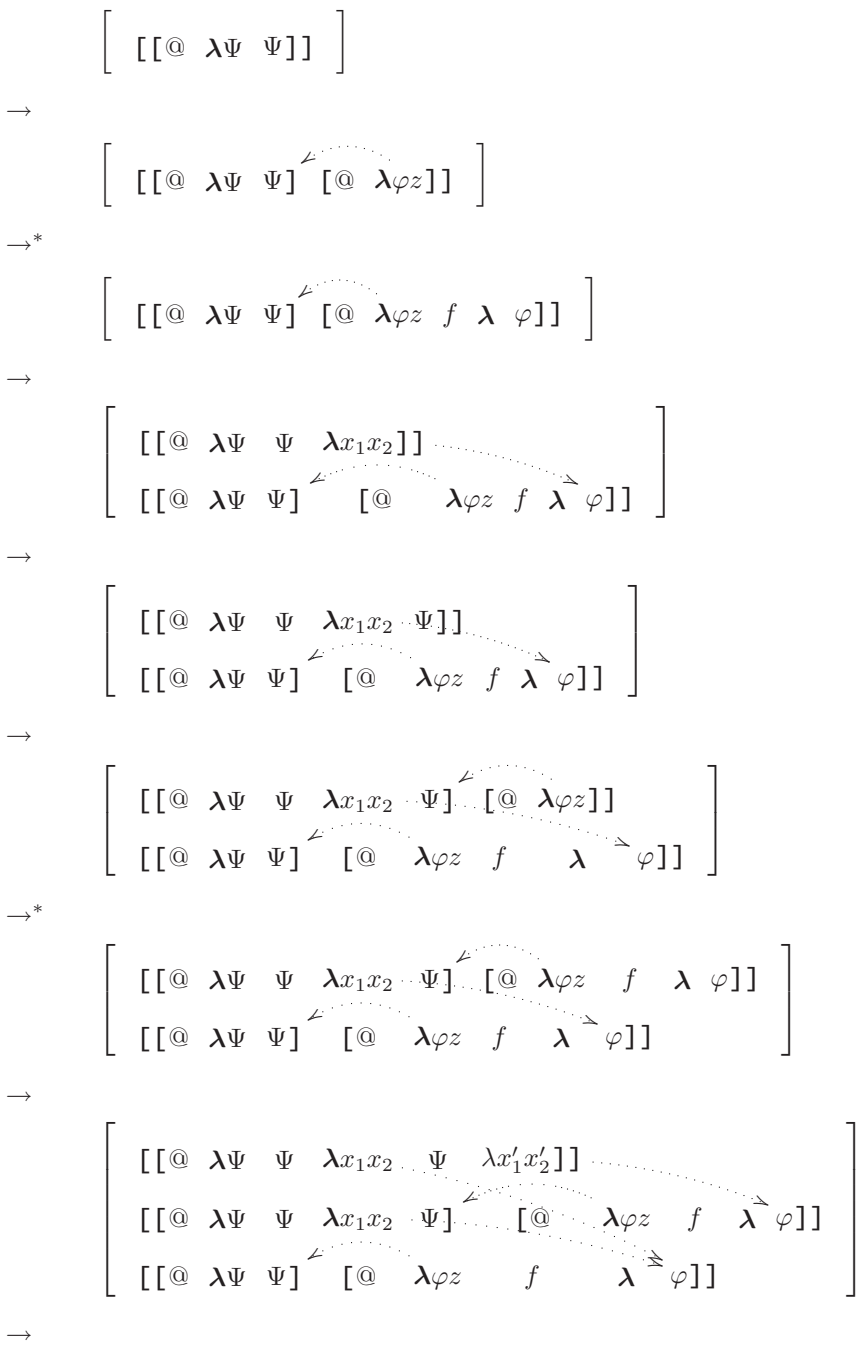


Figure 6: A run of a 3-CPDA (Part 1 of 3).

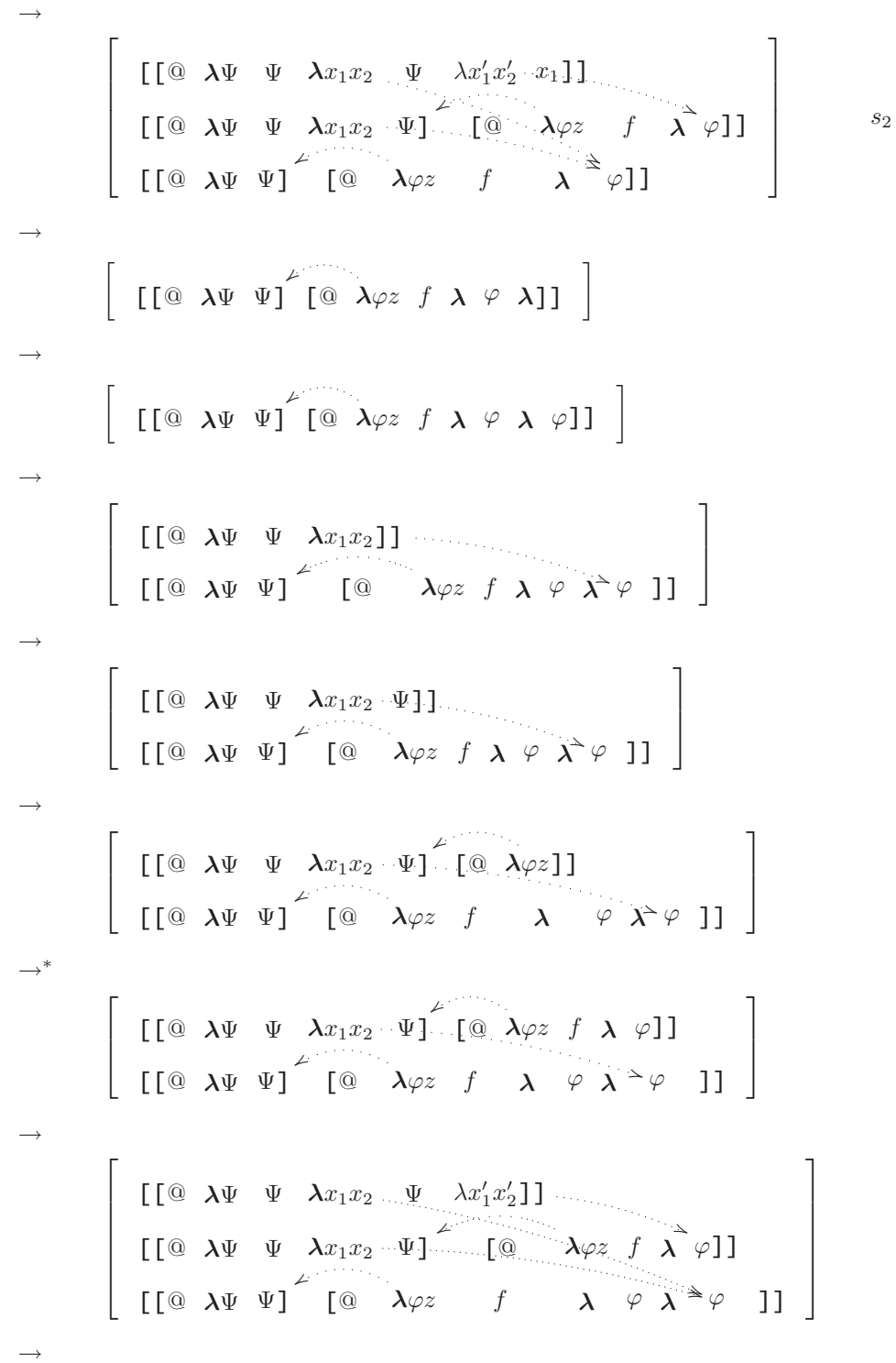


Figure 7: A run of a 3-CPDA (Part 2 of 3).

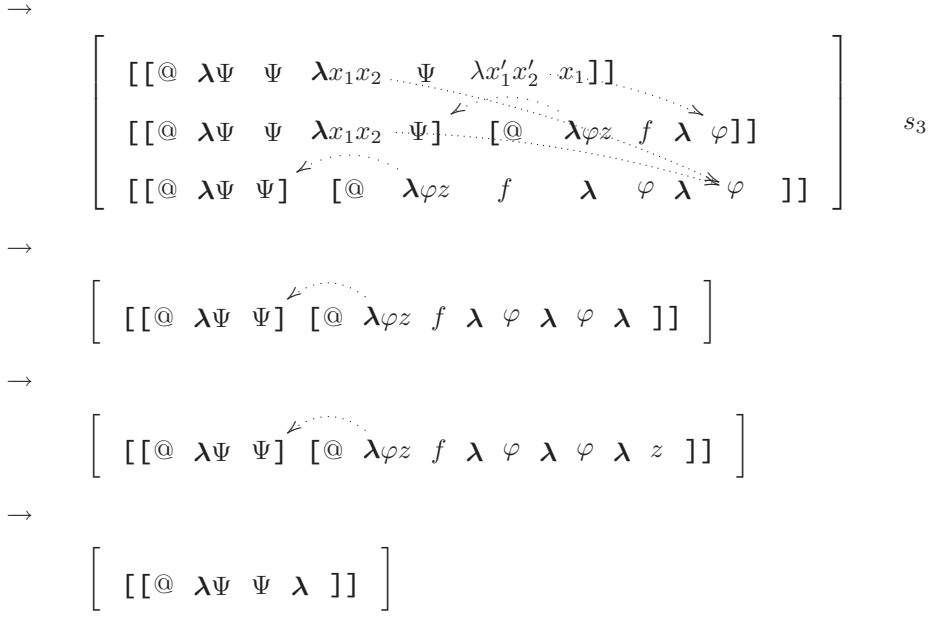


Figure 8: A run of a 3-CPDA (Part 3 of 3).

Note that the definition is not circular, since $t_{\leq v'_i}$ $(1 \leq i < n)$ and $t_{<v'_i}$ $(1 \leq i \leq n)$ are strictly shorter than t. In what follows we shall blur the distinction between v_i and its occurrence v'_i, as it will be clear from the context which occurrence is meant.

Proposition 7 *Let G be an order-n recursion scheme and let s be a reachable configuration of $\mathsf{CPDA}(G)$.*

(i) Let u be a node of $\lambda(G)$. Then u has a link in s if and only if it is a j-child ($j > 0$) labelled by a lambda of type A which has order $l \geq 1$. Further, if u has a link, it points to an $(n-l)$-stack.

(ii) There exists a traversal t over $\lambda(G)$ such that s computes t.

Proof We prove the Proposition by induction on the number of $\rightarrow$-steps s is from the initial configuration. Clearly, the above assertions are valid when s is the initial configuration.

For the inductive cases, suppose $s \rightarrow s'$. Assuming that (i) holds for s, (ii) holds for s and t, we shall prove that (i) holds for s' and (ii) holds for s' and t', where t' is a suitable one-node extension of t. We shall do so by a case analysis of the label of $top_1(s) = u$.

First suppose u's label is not a variable. Then $s' = push_1^v s$, for an appropriate node v. In particular, no new link is created. For (i), observe that, because u's label is not a variable, it follows from the structure of $\lambda(G)$ that if v was a j-child ($j > 0$) labelled by a lambda of type A, then u would have to be a Σ-symbol and, consequently, A would have order 0. Thus, (i) still holds, because no new links have been created.

For (ii), let $t' = tv$, where v has a pointer to a suitable node (there is only one way in which a pointer from v can be inserted so as to make t' into a traversal). Then we have $t > t'$. We shall show that s' computes t'.

For (a) we need to check that $top_2 s' = \ulcorner t' \urcorner$. We have $top_2 s' = (top_2 s)v$ and, in all three cases corresponding to the rules $(\mathbf{A})$, $(\mathbf{S})$, $(\mathbf{L})$, $\ulcorner t' \urcorner = \ulcorner t \urcorner v$ holds. Thus, by induction hypothesis, we get $top_2 s' = \ulcorner t' \urcorner$. For (b) we note that $\underline{s'} = \underline{s}v$ and $\widehat{t'} = \widehat{t}v$. So, by induction hypothesis, $\underline{s'} = \widehat{t'}$. (c) follows immediately from the induction hypothesis and, because no new links have been created, so does (d).

Next suppose u's label is an order-l variable, which is the i-parameter of $binder(u)$ (note that then we have $i \geq 1$) and suppose $binder(u)$ is a j-child. Then $s' = \delta(u)s$ where $\delta(u)$ is given in Definition 5.2. There are four cases; in the following we shall use the notations from the Definition.

1. Case $l \geq 1$ ***and*** $j = 0$**.** Suppose u's label is the order-l variable φ_i.

By the induction hypothesis of (ii), u must be the last node of t. It then follows from the definition of a traversal that t has the following shape:

$$\begin{array}{lcl} t & = & \cdots \overset{0}{\underset{@}{u_0}} \quad \overset{i}{\underset{\boldsymbol{\lambda}\overline{\varphi}}{u_1}} \quad \cdots \quad \underset{\varphi_i}{u} \end{array}$$

(in the figure, the label of a node is the symbol just below it). Since the P-view of a traversal is a path in the HORS graph, $\ulcorner t \urcorner$ has the shape $\cdots u_0 \underbrace{u_1 \cdots u}_{\theta}$ and the segment θ has length $p+1$, where p is the span of the variable node u.

Consider the operation $\delta(u) = push_{n-l+1}; pop_1^{p+1}; push_1^{E_i(top_1),n-l+1}$. By the induction hypothesis of (ii), the top 1-stack of s - call it σ - is the P-view of t. Since the top 1-stack of $push_{n-l+1}s$ is a copy of σ, applying pop_1^{p+1} to $push_{n-l+1}s$ returns a stack that has the @-labelled node u_0 as the top_1-element. The node that is $push_1$ed onto the top of the stack at this point is the i-child of u_0, which we call v. Further, it has a link to the top $(n-l)$-stack of the prefix s of s'.

It follows from the structure of $\lambda(G)$ that v must be labelled by $\boldsymbol{\lambda}\overline{\psi}$ (say) of the same type as the label φ_i of u, i.e. its type is also of order $l \geq 1$. Thus, since $i \geq 1$, (i) follows as required.

For (ii) set $t' = tv$, where v has a pointer (labelled by i) to the occurrence of u_0 indicated in the figure above. Then t' is a traversal and we have $t > t'$. We shall show that s' computes t'.

(a) We need to show $top_2 s' = \ulcorner t' \urcorner$. By definition of s', we have $top_2 s' = (top_2 s)_{\leq u_0} v$, i.e. $top_2 s'$ is the prefix of the 1-stack $top_2 s$ - regarded as a sequence - up to and including the occurrence of u_0 described above, extended by v. By induction hypothesis (a) we have $top_2 s = \ulcorner t \urcorner$. Thus

$$top_2 s' = \ulcorner t \urcorner_{\leq u_0} v = \ulcorner t_{\leq u_0} \urcorner v = \ulcorner t' \urcorner$$

as required (the second equation holds because u_0 appears in $\ulcorner t \urcorner$).

(b) We have $\underline{s'} = \underline{s}v$ and $\widehat{t'} = \widehat{t}v$. Since $\underline{s} = \widehat{t}$ by induction hypothesis (b), we have $\underline{s'} = \widehat{t'}$.

(c) Because the top 1-stack of s' is (a copy of) a prefix of $top_2 s$ extended with v, we can simply appeal to the induction hypothesis (c).

(d) For the same reason as above, (d) holds for all links in $top_2 s'$ except (possibly) the single new link. Let σ' be the $(n-l)$-stack pointed at from v. Then we have $s'_{\sigma'} = s$. Because $t = t'_{<v}$ and s computes t, (d) also holds for the new link.

2. Case $l \geq 1$ ***and*** $j > 0$. Suppose the label of u is the order-l variable φ_i, which is the ith item of the list $\overline{\varphi}$.

By induction hypothesis (ii) and definition of a traversal, t has the following shape:

$$\begin{array}{lcl} t & = & \cdots \underset{\Psi}{u_0} \quad \overset{i}{\underset{\boldsymbol{\lambda}\overline{\varphi}}{u_1}} \quad \cdots \quad \underset{\varphi_i}{u} \end{array}$$

Further, the variable ψ has the same type as $\boldsymbol{\lambda}\overline{\varphi}$, which (say) is of order l'. It follows that $l' > l$ and, consequently, $l' \geq 1$. By induction hypotheses (i) and (ii), s has the following shape

$$s \quad = \quad [\cdots\cdots\cdots [\cdots\sigma\cdots \underbrace{\underbrace{[\cdots u_1 \cdots u]}_{\text{top 1-stack of } s}\ ]\cdots]}_{\text{top } (n-l)\text{-stack of } s}$$

wherein u_1 has a link to some $(n-l')$-stack σ. Since $l' > l$, the $(n-l')$-stack σ is embedded in the top $(n-l)$-stack of s (Lemma 6), as indicated by the figure above. Note that, by induction hypothesis (iid), s_σ computes $t_{<u_1}$. In particular the top_1-element of σ must be u_0.

Now, to see the structure of s', consider the operation $\delta(u)$. Let $w = top_{n-l+1}s$. The operation $push_{n-l+1}s$ pushes a copy of w on top of s. The rest of $\delta(u)$, namely,

$$pop_1^p; collapse; push_1^{E_i(top_1),n-l+1},$$

affects only the top (duplicate) copy of w. Applying pop_1^p to $push_{n-l+1}s$ returns a stack that has u_1 as the top_1-element; the $collapse$-operation then reduces it to a stack that has a copy σ' (say) of σ as its top $(n-l')$-stack, i.e. its top_1-element is u_0. The node that is $push_1$ed onto the top of the stack at the end of the $\delta(u)$-operation (to yield s') is the i-child of u_0, which we shall call v. Observe that the structure of $\lambda(G)$ implies that v must then be labelled by $\boldsymbol{\lambda}\overline{\chi}$ (say) whose type is the same as that of φ_i, i.e. its order is l. Since v is linked to the $(n-l)$-stack w, (i) is satisfied.

For (ii) let $t' = tv$, where v has an i-pointer to the distinguished occurrence of u_0. t' is then a traversal such that $t > t'$. We need to show that s' computes t'.

(a) Observe that $\ulcorner t'\urcorner = \ulcorner t_{<u_0}\urcorner u_0 v = \ulcorner t_{<u_1}\urcorner v$. Since s_σ computes $t_{<u_1}$, so does $s'_{\sigma'}$. Hence, $top_2 s' = \ulcorner t_{<u_1}\urcorner v = \ulcorner t'\urcorner$.

(b) Observe that $\underline{s'} = \underline{s}v$ and $\widehat{t'} = \widehat{t}v$. Thus, by induction hypothesis, $\underline{s'} = \widehat{t'}$.

(c) Since $s'_{\sigma'}$ computes $t_{<u_1}$, (c) holds.

(d) We only need to verify (d) for the new link (all other links satisfy (d) because $s'_{\sigma'}$ computes $t_{<u_1}$). Recall that v points at the stack w. Since $s'_w = s$ and $t'_{<v} = t$, (d) holds because s computes t.

3. Case $l = 0$ ***and*** $j = 0$**.** Suppose u's label is the order-0 variable x.

By induction hypothesis (ii) and the definition of a traversal, t must have the following shape:

$$t \quad = \quad \cdots \overset{\overset{0}{\swarrow}\ \overset{i}{\swarrow}}{u_0\ \ u_1}\ \cdots\ u$$
$$\qquad\qquad\quad @\ \ \boldsymbol{\lambda}\overline{\varphi}\qquad x$$

As in **1.**, $\ulcorner t\urcorner$ has the shape $\cdots u_0 \underbrace{u_1 \cdots u}_{\theta}$ and the segment θ has length $p+1$, where p is the span of the variable node u.

Consider the operation $\delta(u) = pop_1^{p+1}; push_1^{E_i(top_1)}$. Applying pop_1^{p+1} to s returns a stack that has the @-labelled node u_0 as the top_1-element. The node that is $push_1$ed onto the top of the stack at this point is the i-child of u_0, which we call v. It follows from the structure of $\lambda(G)$ that v must be labelled by $\boldsymbol{\lambda}$, i.e. its type has order 0. Thus, since v has no link, (i) follows as required.

For (ii) set $t' = tv$, where v has a pointer (labelled by i) to the occurrence of u_0 indicated above. Then t' is a traversal and we have $t > t'$. We shall show that s' computes t'.

(a) We need to show $top_2 s' = \ulcorner t' \urcorner$. By definition of s', we have $top_2 s' = (top_2 s)_{\leq u_0} v$. By induction hypothesis (ii), we have $top_2 s = \ulcorner t \urcorner$. Thus

$$top_2 s' = \ulcorner t \urcorner_{\leq u_0} v = \ulcorner t_{\leq u_0} \urcorner v = \ulcorner t' \urcorner$$

as required (the second equation holds because u_0 appears in $\ulcorner t \urcorner$).

(b) We have $\underline{s'} = \overline{(pop_1^{p+1} s)\, v}$ and $\widehat{t'} = \widehat{t_{\leq u_0}} v$. By induction hypothesis (iic), $pop_1^{p+1} s$ computes $t_{\leq u_0}$, in particular $\underline{pop_1^{p+1} s} = \widehat{t_{\leq u_0}}$. Thus, (b) holds.

(c) We simply appeal to the induction hypothesis (iic).

(d) Note that no new links have been created in this case, so it suffices to appeal to the induction hypothesis (iid).

4. Case $l = 0$ ***and*** $j > 0$**.** Suppose the label of u is the order-0 variable x, which is the ith item of the list $\overline{\varphi}$.

By induction hypothesis (ii) and definition of a traversal, t has the following shape:

$$\begin{array}{lcl} t & = & \cdots \; u_0 \;\; \overset{i}{\overbrace{u_1}} \;\; \cdots \;\; u \\ & & \quad\; \psi \;\; \boldsymbol{\lambda}\overline{\varphi} \qquad\; x \end{array}$$

Further, the variable Ψ has the same type as $\boldsymbol{\lambda}\overline{\varphi}$, which (say) is of order l'. It follows that $l' > l$. By induction hypotheses (i) and (ii), s has the following shape

$$s \quad = \quad [\, \cdots\cdots \sigma \cdots \underbrace{[\, \cdots u_1 \cdots u \,]}_{\text{top 1-stack}} \cdots \,]$$

wherein u_1 has a link to some $(n - l')$-stack σ. Note that, by induction hypothesis (iid), s_σ computes $t_{<u_1}$. In particular the top_1-element of σ must be u_0.

Now, to understand what s' looks like, consider the operation $\delta(u) = pop_1^p; collapse; push_1^{E_i(top_1)}$. Applying pop_1^p to s returns a stack that has u_1 as the top_1-element; the $collapse$-operation then reduces it to a stack that has σ as its top $(n - l')$-stack, i.e. its top_1-element is u_0. The node that is then $push_1$ed onto the top of the stack at the end of the $\delta(u)$-operation (to yield s') is the i-child of u_0, which we shall call v. Observe that the structure of $\lambda(G)$ implies that v must then be labelled by $\boldsymbol{\lambda}$. Since v does not have a link, (i) is satisfied.

For (ii) let $t' = tv$, where v has an i-pointer to the distinguished occurrence of u_0. t' is then a traversal such that $t > t'$. We need to show that s' computes t'.

(a) Observe that $\ulcorner t' \urcorner = \ulcorner t_{<u_1} \urcorner v$. Since s_σ computes $t_{<u_1}$, we have $top_2 s' = \ulcorner t_{<u_1} \urcorner v = \ulcorner t' \urcorner$.

(b) Observe that $\underline{s'} = \underline{s_\sigma} v$ and $\widehat{t'} = \widehat{t_{<u_1}} v$. Again, since s_σ computes $t_{<u_1}$, we have $\underline{s_\sigma} = \widehat{t_{<u_1}}$ and (b) follows.

(c) Because s_σ computes $t_{<u_1}$, (c) holds.

(d) Again, it suffices to appeal to the fact that s_σ computes $t_{<u_1}$, because no new links have been created.

□

Note that in the above proof t' was constructed from s' in a lock-step fashion. Moreover, observe that, when the last node of a traversal is not a Σ-symbol, traversals (as well as the corresponding runs of $\mathsf{CPDA}(G)$) can be extended in a unique way. Similarly, when the last node of a traversal is a Σ-symbol f, both traversals and the corresponding runs of $\mathsf{CPDA}(G)$ can be extended in $ar(f)$ matching ways. Consequently, we have:

Corollary 8 *Suppose s computes t. Then s and t are "bisimilar" with regard to $\rightarrow$ and $>$ respectively:*

(i) If $s \rightarrow s'$ then there exists t' such that $t > t'$ and s' computes t'.

(ii) If $t > t'$ then there exists s' such that $s \rightarrow s'$ and s' computes t'.

Theorem 9 (Equi-Expressivity 2) *For every order-n recursion scheme G, $\mathsf{CPDA}(G)$ computes all paths in the value tree $[\![G]\!]$ generated by G.*

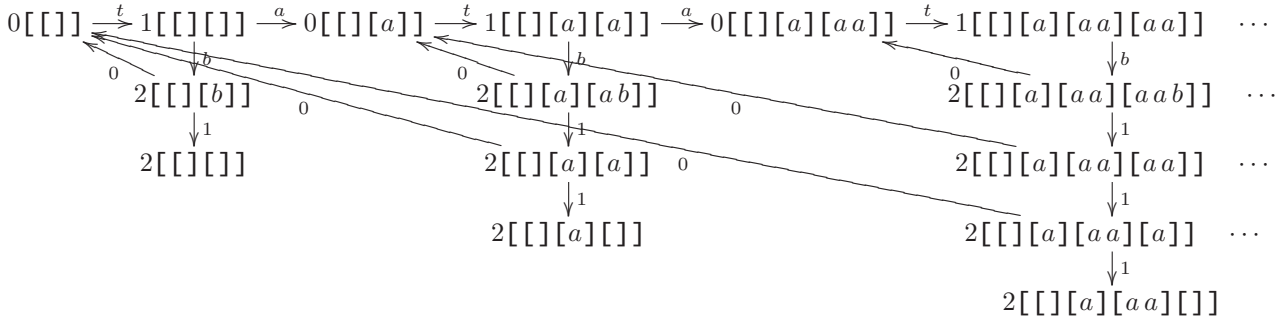
6 Games over collapsible pushdown graphs

We are interested in solving parity games over collapsible pushdown graphs i.e. we want to know whether one can decide, for any position in such a game, if Éloïse has a winning strategy from it, and if so, determine its complexity. An ***order-n collapsible pushdown system***[14] (n-CPDS) is given by a quadruple $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$ where Γ is the stack alphabet, Q is a finite state-set, $\Delta \subseteq Q \times \Gamma \times Q \times Op_n$ is the transition relation, and q_0 is the initial state. *Configurations* of an n-CPDS are pairs of the form (q, s) where $q \in Q$ and s is an n-stack over Γ. We define a one-step labelled transition relation of the CPDS $\mathcal{A}$, written $\overset{\ell}{>}$ where $\ell \in Q \times Op_n$, which is a family of binary relations over configurations, as follows: $(q, s) \overset{(q',\theta)}{>} (q', s')$ just if we have $(q, top_1\, s, q', \theta) \in \Delta$ and $s' = \theta(s)$. The initial configuration is $(q_0, \bot_n)$. We can now define the ***configuration graph*** of $\mathcal{A}$: vertices are just the (reachable) configurations, and the edge relation is the relation $\overset{\ell}{>}$ restricted to the reachable configurations.

Example 6.1 Take the 2-CPDS[15] with state-set $\{ 0, 1, 2 \}$, stack alphabet $\{ a, b, \bot \}$ and transition relation given by

$$(0, -, 1, t),\ (1, -, 0, a),\ (1, -, 2, b),\ (2, \dagger, 2, 1),\ (2, \dagger, 0, 0)$$

where $-$ means any symbol, $\dagger$ means any non-$\bot$ symbol, and $t, a, b, 0$ and 1 are shorthand for the stack operations $push_2$, $push_1^{a,2}$, $push_1^{b,2}$, $collapse$ and pop_1 respectively. We present its configuration graph (with edges labelled by stack operations only) as follows:



[14] We use collapsible pushdown *system* (as opposed to *automaton*) whenever the device is used to generate a graph.

[15] This is inspired by an example in [6].

Let $G = \langle V, E \rangle$ denote the configuration graph of $\mathcal{A}$, let $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ be a partition of Q and let $\Omega : Q \longrightarrow C \subset \mathbb{N}$ be a colouring function. Altogether they define a partition $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ of V whereby a vertex belongs to $V_{\mathbf{E}}$ iff its control state belongs to $Q_{\mathbf{E}}$, and a colouring function $\Omega : V \longrightarrow C$ where a vertex is assigned the colour of its control state. The structure $\mathcal{G} = \langle G, V_{\mathbf{E}}, V_{\mathbf{A}} \rangle$ is an ***n*-CPDS game graph** and the pair $\mathbb{G} = \langle \mathcal{G}, \Omega \rangle$ is an ***n*-CPDS parity game**. A *play* in $\mathbb{G}$ from the initial vertex $v_0 = (q_0, \bot_n)$ is defined as follow: the player that controls v_0 (Éloïse if $v_0 \in V_{\mathbf{E}}$ or Abelard otherwise) moves a token from v_0 to some neighbour v_1 (we assume here that G has no dead-end), then the player that controls the token moves it to a neighbour v_2 of v_1 and so on. A play is therefore an infinite path $v_0 v_1 \cdots$ and is won by Éloïse iff $\liminf \langle \Omega(v_i) : i \geq 0 \rangle$ is even. Finally, v_0 is winning for a player if he has a winning strategy from it. See [25, 28, 27] for more details.

In this section we consider the following problem:

($\mathbf{P}_1$) *Given an n-CPDS parity game decide if Éloïse has a winning strategy from the initial configuration.*

From the well-known techniques of [11], it follows that (i) Problem ($\mathbf{P}_1$) is polynomially equivalent to Problems ($\mathbf{P}_2$) and ($\mathbf{P}_3$) in the following; and (ii) Problem ($\mathbf{P}_1$) is equivalent to Problem ($\mathbf{P}_4$) – the reduction from ($\mathbf{P}_1$) to ($\mathbf{P}_4$) is polynomial, but non-elementary one in the other direction :

($\mathbf{P}_2$) Given an n-CPDS graph G, and a mu-calculus formula φ, does φ hold at the initial configuration of G?

($\mathbf{P}_3$) Given an alternating parity tree automaton and n-CPDS graph G, does it accept the unravelling of G?

($\mathbf{P}_4$) Given an MSO formula φ and an n-CPDS graph G, does φ holds at the root of the unravelling of G?

An useful fact is that the unravelling of an n-CPDS graph is actually generated by an n-CPDA (one mainly has to note that putting labels on the edges makes the n-CPDS graph *deterministic* and hence its unravelling as desired).

Lemma 10 *Let $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$ be some n-CPDS and let G be its configuration graph. Then let t be the tree obtained by unravelling G and by labeling every node by control state and top_1 stack element of the corresponding configuration in G. Then t is generated by an n-CPDA of polynomial size in the one of $\mathcal{A}$.*

Proof Consider the following n-CPDA $\mathcal{A}' = \langle \Sigma, \Gamma, Q', \delta, q_0 \rangle$ where we set:

- $Trans = \{(q, \theta) \mid \exists p \in Q,\ a \in \Gamma \text{ s.t. } (p, a, q, \theta) \in \Delta\}$ is the set of all transitions that can be applied in $\mathcal{A}$.
- $Q' = Q \cup Trans$
- $\Sigma = Q \times \Gamma$ is the set of shapes (here are do not care of the link) and the arity of $(q, a) \in \Sigma$ is $|\{(q', \theta) \mid (q, a, q', \theta) \in \Delta\}|$.
- For every $q \in Q$, and every $a \in \Gamma$, $\delta(q, a) = ((q, a); (q_1, \theta_1), \cdots, (q_k, \theta_k))$ where $\{(q_1, \theta_1), \cdots, (q_k, \theta_k)\} = \{(q', \theta) \mid (q, a, q', \theta) \in \Delta\}$.
- For every $(q, \theta) \in Trans$, and every $a \in \Gamma$, $\delta((q, \theta), a) = (q, \theta)$.

Then one easily checks that $\mathcal{A}'$ generates t. □

An important consequence of the Equi-Expressivity Theorem is the following.

Theorem 11 *Let t be a tree generated by an order-n recursion scheme. Consider the following problems:*

($\mathbf{P}'_2$) *Given t and a modal mu-calculus formula φ, does φ hold at the root of t?*

($\mathbf{P}'_3$) *Given t and an alternating parity tree automaton, does the automaton accept t?*

($\mathbf{P}'_4$) *Given t and an MSO formula φ, does φ hold at the root of t?*

Then problem $(\mathbf{P}_i)$ is polynomially equivalent to problem $(\mathbf{P}'_i)$ for every $i = 2, 3, 4$.

Since the modal Mu-calculus model checking problem for trees generated by (higher-order) recursion schemes is decidable [20], we obtain the following as an immediate consequence.

Theorem 12 *Problems $(\mathbf{P}_1)$ – $(\mathbf{P}_4)$ are decidable with complexity n-EXPTIME complete.*

Hence the Equi-Expressivity Theorem is a powerful tool to transfer decidability properties from recursion schemes to CPDS. Another remarkable consequence is that it gives totally new techniques for model-checking or solving games played on infinite structure generated by automata. In particular it leads to new proofs/optimal algorithms for the special cases that were considered previously [26, 3, 16]. Conversely, as the Equi-Expressivity Theorem works in both directions, one can note that a solution of problem $(\mathbf{P}_1)$ would give a new proof of the decidability of problems $(\mathbf{P}_1)$–$(\mathbf{P}_1)$, and would give a new approach to problems on recursion schemes.

Actually, the techniques of [26, 16] can be generalized to solve n-CPDS parity games without referring to [20]. Further it gives effective winning strategies for the winning player (which was not the case in [16] where the special case $n = 2$ was considered).

Theorem 13 *Solving an n-CPDS parity game is n-EXPTIME complete and it can be achieved without appealing to a translation to recursion schemes and to the decidability result in [20]. Moreover, one can build an n-CPDA with output that realizes a winning strategy for the winning player.*

The next section is devoted to the proof of Theorem 13

6.1 Solving games over collapsible pushdown games: a direct proof

We an order-n collapsible pushdown system $\mathcal{P} = \langle \Gamma, Q, \Delta \rangle$, a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q and a colouring function $\Omega : Q \to C \subset \mathbb{N}$. We denote by $G = (V, E)$ the transition graph of $\mathcal{P}$, by $\mathcal{G} = \langle G, V_{\mathbf{E}}, V_{\mathbf{A}} \rangle$ the game graph associated with the previous partition of Q, and by $\mathbb{G} = \langle \mathcal{G}, \Omega \rangle$ the parity game on $\mathcal{P}$ induced by the previous colouring function.

In this section, we give a proof of Theorem 13, *i.e.* we explain how to solve the parity game $\mathbb{G}$ defined from $\mathcal{P}$. Here, we only focus on deciding the winner from a configuration of the form $(p_{in}, \bot_n)$, that is a configuration with an empty stack. Then it is easy to solve the game from any configuration. Indeed if one wants to decide for some configuration $(p, \sigma) \in V$, it suffices to construct a new n-CPDS that mimics $\mathcal{P}$ excepts that from its initial configuration it starts by reaching configuration (p, σ) and only when this has happens the simulation of $\mathcal{P}$ starts. Therefore (p, σ) is winning for the same player in $\mathbb{G}$ and in the (naturally defined) new n-CPDS game. Note that this reduction is linear in the size of both $\mathcal{P}$ and in the length of (p, σ).

The section is organized as follows. We start by giving a general overview of the proof. Then we prove a technical result (Lemma 14) which allows to restrict our attention to a specific class of n-CPDS (collapse rank-aware CPDS). For this class, we provide a reduction of an n-CPDS game to an $(n-1)$-CPDS games, which allows to conclude using an inductive argument.

Overview of the proof and preliminaries

We follow the general approach of [26] and [16]. In [26], in order to solve a pushdown parity game, one builds a simulation game that is played on a finite game graph and is equipped with a parity condition. Then one proves that a configuration in the pushdown game is winning for Éloïse iff a corresponding configuration is winning for her in the simulation game. Moreover from a winning strategy in the simulation game, one can built a winning strategy in the pushdown game. The increase of memory results in the use of a stack. In [16] the authors consider parity games on 2-CPDS[16] and mostly adapt the construction of [26]. To handle the collapse actions, they need to make the CPDS *rank aware*. Assume that the play is in some configuration v where collapse may be applied. Applying collapse leads to some configuration v', and the stack content in v' has already appeared before v in the play. The *collapse ancestor* of v is the latest configuration in the play with this stack content. A CPDS is rank aware (to be defined formally) if the top stack symbol contains the *collapse rank* which is the minimal rank of a state occurring between the collapse ancestor of the current configuration and the current configuration. Now solving a 2-CPDS game when the 2-CPDS is rank aware, can be done by *adapting* the reduction in [26] (now the simulation game is no longer played on a finite graph but can still be solved). Let us mention an important point here. In [16], the authors proves the following: if one player win the 2-CPDS game then he wins the simulation game. Hence this proves the equivalence of both games but do not provide a way to build a strategy in the 2-CPDS game.

Here, we extend the result of [16] to n-CPDS parity game for any n. Moreover, from our proof, we can effectively build winning strategies in the n-CPDS game. The proof works by induction on n. Note that the special case where $n = 1$ is the one of parity pushdown game studied in [26].

What follows do not rely on a specific representation of the links and therefore we do not make it more precise.

We start with some definitions adapted from the one introduced in [16].

Definition 6.2 Consider a partial play Λ in $\mathbb{G}$ ending in a configuration (q, s) such that $top_1(s)$ has an n-link. Hence there is in Λ at least one configuration of the form $(q', collapse(s))$ for some $q' \in Q$. Then the closest to (q, s) is called the *collapse ancestor* of (q, s). The *collapse rank* of (q, s) is the minimal colour of a state occurring in Λ between the panic ancestor of (q, s) and (q, s). Note that these notions are not defined if $top_1(s)$ has an k-link for some $k < n$: indeed it may happen that no configuration of the form $(q', collapse(s))$ was visited in Λ, and therefore the collapse ancestor notion can not be adapted.

Definition 6.3 An n-CPDS equipped with a colouring function is *collapse rank-aware* iff there exists a function $ColRk : \Gamma \to \mathbb{N}$ such that, when defined, the collapse rank of every configuration (q, s) is equal to $ColRk(top_1(s))$. In other words, the collapse rank is stored in the top_1-element of the stack.

We also introduce a notion of ancestor.

Definition 6.4 Consider a play $\Lambda = v_1 v_2 \cdots$ in $\mathbb{G}$ and consider that we attach to every k-stack (for any $1 \leq k \leq (n-1)$) an identifier (which is an integer) as follows: if a $push_k$ operation is applied at configuration

[16] They use the terminology *panic automata* instead of collapsible pushdown automata.

v_m in Λ, the top $(k-1)$-stack of configuration v_{m+1} is assigned identifier m, and all identifiers for the j-stacks for $j < (k-1)$ in this new top $(k-1)$-stack are copied from the former top $(k-1)$-stack. Now, for $2 \leq k \leq n$ the *k-ancestor* of some configuration v in Λ is the configuration v_m where m is the identifier of v's top $(k-1)$-stack, and the *level rank for k* is the minimal colour of a state occurring in Λ between the $(k-1)$-ancestor of v and v.

Example 6.5 In the following (order-3) example, we assume that the colour of v_i is i, and the identifier j of a stack $[\cdots]$ is denoted by $[\cdots]_j$. Starting from $v_1 = [[[]_0]_0]$, consider the following sequence of stack actions:

$$\begin{array}{rll}
 & [[[]_0]_0] & = v_1 \\
\xrightarrow{push_2} & [[[]_0[]_1]_0] & = v_2 \\
\xrightarrow{push_1^{a,2}} & [[[]_0[a^{(2,1)}]_1]_0] & = v_3 \\
\xrightarrow{push_3} & [[[]_0[a^{(2,1)}]_1]_0[[]_0[a^{(2,1)}]_1]_3] & = v_4 \\
\xrightarrow{push_2} & [[[]_0[a^{(2,1)}]_1]_0[[]_0[a^{(2,1)}]_1[a^{(2,1)}]_4]_3] & = v_5
\end{array}$$

Then the 2-ancestor of v_5 is v_4 and hence the level rank for 2 of v_5 is 4. The 3-ancestor of v_5 is v_3 and the level rank for 3 of v_5 is therefore 3. The 2 ancestor of v_4 is v_1 and therefore the level rank for 2 of v_4 is 1.

We show, as in [16, Lemma 6.3], that we can restrict our attention to CPDS games where the underlying CPDS is collapse rank-aware.

Lemma 14 *For any n-CPDS $\mathcal{P}$ and any parity game $\mathbb{G}$ on it, one can construct a collapse rank-aware n-CPDS $\mathcal{P}'$ and an associated parity game $\mathbb{G}'$ such that Éloïse has a winning strategy in $\mathbb{G}$ from some configuration $(p_{in}, \bot_n)$ iff she has a winning strategy in $\mathbb{G}'$ from the same configuration.*

Proof

The proof is an non-trivial adaptation of the one of [16, Lemma 6.3] to the general setting of n-CPDS (instead of 2-CPDS).

Fix an n-CPDS $\mathcal{P} = \langle \Gamma, Q, \Delta \rangle$, a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q and a colouring function $\Omega : Q \to C \subset \mathbb{N}$. Denote by $\mathbb{G}$ the induced parity game. We define a collapse rank-aware (to be proven) n-CPDS $\mathcal{P}' = \langle \Gamma', Q', \Delta' \rangle$ such that $Q \subset Q'$ and $\Gamma' = \Gamma \times C \times C^{\{2,\ldots,n\}} \times C^{\{2,\ldots,n\}}$. A configuration (q, s) of $\mathcal{P}'$ with its $top_1(s) = (a, m_c, \tau, \tau_{old})$ having a $(k+1)$-link will satisfy the following.

- m_c is the minimal colour seen since the k-ancestor of the k-stack pointed to by the $(k+1)$-link. In particular, when the link is an n-link, equivalently when the collapse rank is defined, m_c will be the collapse rank: indeed the $(n-1)$-stack pointed by the n-link is such that its n-ancestor is exactly the collapse ancestor of the current configuration and hence m_c is the smallest colour seen since the collapse ancestor. By abuse of notation, we designate in the sequel m_c as the *collapse rank*.
- τ is the *level rank*, that is, for any $i = 2, \cdots, n$, $\tau(i)$ is the level rank for i.
- τ_{old} is such that $\tau_{old}(k) = h$ where $h = \tau'(k)$ with $top_1(pop_k(s)) = (a', m'_c, \tau', \tau'_{old})$ if exists and otherwise may be any h. In other word $\tau_{old}(k)$ gives the level rank for k for the $(k-1)$-stack just above the current one. Note that τ_{old} is introduced here only for technical reasons and is actually easy to maintain (one has to care only when pushing).

The transition relation of $\mathcal{P}'$ mimics the one of $\mathcal{P}$ and updates the ranks as explained below.

In order to save space and to make the construction more understandable, we do not describe formally Δ' but explain how $\mathcal{P}'$ is supposed to behave. It should be clear that Δ' can be formally described to fit this informal description (and that some extra control states are needed). Note also that the following description contains also the inductive proof of its validity, namely that m_c, τ and τ_{old} are as stated above.

Assume $\mathcal{P}'$ is in some configuration (q, s) with $top_1(s) = (a, m_c, \tau, \tau_{old})$. The following behaviours are those allowed in such a configuration.

1. For every $(q, a, q', pop_k) \in \Delta$ with $1 \leq k \leq n$, let $pop_k(s) = s'$ and let $top_1(s') = (a', m'_c, \tau', \tau'_{old})$. Then $\mathcal{P}'$ can go to the configuration (q', s'') where s'' is obtained from s' by replacing $top_1(s')$ by $(a', \min(m'_c, \tau(k), \Omega(q')), \tau'', \tau'_{old})$, with

$$\tau''(i) = \begin{cases} \min(\tau'(i), \tau(k), \Omega(q')) & \text{if } i \leq k \\ \min(\tau(i), \Omega(q')) & \text{if } i > k \end{cases}$$

   Indeed the play since this move was ending by a sequence $(q_1, s_1)(q_2, s_2) \cdots (q, s)$ (here the dots are for intermediate configurations that have no importance for the current argument) where (q_2, s_2) is the k-ancestor of (q, s) and was reached from (q_1, s_1) by applying a $push_k$ move, and $top_k(s_1) = top_k(s')$, which in particular means that (q_1, s_1) and (q', s') have the same i-ancestors for $i \leq k$ and that $top_1(s_1)$ and $top_1(s')$ are equal and point to two respective $(h-1)$-stacks (for some h) that both have the same h-ancestor. Hence, the collapse rank in (q', s') being the smallest colour since the h-ancestor of the pointed $(h-1)$-stack it is equal to the minimum of the collapse rank in (q_1, s_1) (namely m'_c) with $\Omega(q')$ and with the minimal colour visited in $(q_2, s_2) \cdots (q, s)$, which is (as (q_2, s_2) is the k-ancestor of (q, s)) equal to $\tau(k)$. Hence the collapse rank is $\min(m'_c, \tau(k), \Omega(q'))$.
   Now the level rank in (q', s') for some $i \leq k$ being the smallest colour seen since the i-ancestor of (q', s'), and this i-ancestor being the same as in (q_1, s_1), the level rank is the minimum of the level rank for i in (q_1, s_1) (namely $\tau'(i)$) with $\Omega(q')$ and with the minimal colour visited in $(q_2, s_2) \cdots (q, s)$ (namely $\tau(k)$). Hence it is equal to $\min(\tau'(i), \tau(k), \Omega(q'))$.
   The i-ancestor of (q', s') for some $i > k$ equals the i-ancestor of (q, s), and hence the level rank for i has to be simply updated from the one in (q, s) (namely $\tau(i)$) by taking the minimum with $\Omega(q')$.
   Finally the τ'_{old} information needs not to be updated.

2. For every $(q, a, q', push_j) \in \Delta$ with $2 \leq j \leq n$, let $push_j(s) = s'$ and then $top_1(s') = (a, m_c, \tau, \tau_{old})$. Then $\mathcal{P}'$ can go to the configuration (q', s'') where s'' is obtained from s' when replacing $top_1(s')$ by $(a, \min(m_c, \Omega(q')), \tau', \tau'_{old})$ with $\tau'(i) = \min(\tau(i), \Omega(q'))$ if $i \neq j$ and $\tau'(j) = \Omega(q')$, and $\tau'_{old}(i) = \tau_{old}(i)$ if $i \neq j$ and $\tau'_{old}(j) = \tau(j)$.

   Indeed, in the new configuration, the h-ancestor of the $(h-1)$-stack pointed by $top_1(s')$ (for some h) is the same as the h-ancestor of the $(h-1)$-stack pointed by $top_1(s)$ and hence the collapse rank simply needs to be updated by taking the minimum of the former one (namely m_c) with the current colour (namely $\Omega(q')$).
   Now, the i-ancestors for any $i \neq j$ in (q, s) and (q', s') are the same and therefore the level rank for i simply needs to be updated by taking the minimum of the former one (namely $\tau(i)$) with the current colour (namely $\Omega(q')$).
   The j-ancestor of (q', s') is (q, s), and hence the level rank for j in (q', s') equals $\Omega(q')$.
   Finally, one has $top_1(pop_i(s)) = top_1(pop_i(s'))$ for every $i \neq j$ and therefore $\tau'_{old}(i) = \tau_{old}(i)$ gives the correct value, and as $top_1(pop_j(s')) = (a, m_c, \tau, \tau_{old})$, one must set $\tau'_{old}(j) = \tau(j)$.

3. For every $(q, a, q', push^b_{1,k}) \in \Delta$ with $1 \leq k \leq n$, and $b \in (\Gamma \setminus \{\bot\})$, then $\mathcal{P}'$ apply $push^{(b, m'_c, \tau', \tau'_{old})}_{1,k}$

where $m'_c = \min(\tau_{old}(k), \tau(k), \Omega(q'))$, $\tau'(i) = \min(\tau(i), \Omega(q'))$ for every i and $\tau'_{old}(i) = \tau_{old}(i)$ for every i.

In order to define the correct value for m'_c one needs to consider the minimal colour since the k ancestor of the $(k-1)$-stack pointed by the new k-link. The play since the last move was ending by a sequence of the form $(q_1, s_1) \cdots (q_2, s_2)(q_3, s_3) \cdots (q, s)$ (here the dots are for intermediate configurations that have no importance for the current argument) where (q_1, s_1) is the k-ancestor of the $(k-1)$ stack pointed by the new link, (q_3, s_3) is the k-ancestor of (q, s) and therefore the move from (q_2, s_2) is a $push_k$ one. Moreover it is easily checked that $top_k(s_2) = top_k(pop_k(s))$ and therefore $top_1(s_2) = top_1(pop_k(s))$. Hence, $\tau_{old}(k)$ is the level rank for k in (q_2, s_2). Now note that the smallest colour visited since (q_3, s_3) is by definition the level rank for k in (q, s) (namely $\tau(k)$) and therefore the correct value for m'_c is the minimum of the minimum between (q_1, s_1) and (q_2, s_2) with the minimum between (q_3, s_3) and (q, s) and $\Omega(q')$, that is $m'_c = \min(\tau_{old}(k), \tau(k), \Omega(q'))$. Note that here $\tau_{old}(k)$ makes sense as if $top_1(pop_k(s))$ is not defined it would means that $pop_k(s)$ would be empty and then $push^b_{1,k}$ is not allowed (there is no $(k-1)$-stack to target).

Then the i-ancestors for all i in (q, s) and (q', s') are the same and therefore the level rank for i simply needs to be updated by taking the minimum of the former one (namely $\tau(i)$) with the current colour (namely $\Omega(q')$).

Finally, one has $top_1(pop_i(s)) = top_1(pop_i(s'))$ for every i and therefore $\tau'_{old}(i) = \tau_{old}(i)$ gives the correct value.

4. For every $(q, a, q', collapse) \in \Delta$, let $collapse(s) = s'$ and let $top_1(s') = (a', m'_c, \tau', \tau_{old})$. Then $\mathcal{P}'$ can go to the configuration (q', s'') where s'' is obtained from s' by replacing $top_1(s')$ by $(a', \min(m'_c, m_c, \Omega(q')), \tau'', \tau'_{old})$ with $\tau''(i) = \min(\tau'(i), m_c, \Omega(q'))$ for every i.

   Indeed the play since the last move (namely $collapse$) was ending by a sequence of the form $(q_1, s_1) \cdots (q_2, s_2)(q_3, s_3) \cdots (q, s)$ (here the dots are for intermediate configurations that have no importance for the current argument) where (q_1, s_1) is the k-ancestor of the $(k-1)$-stack pointed by the k-link in the top symbol of (q, s), (q_2, s_2) is such that $top_k(s_2) = top_k(s')$ and (q_3, s_3) is reached from (q_2, s_2) by a $push_k$ move. Consider the i ancestor for (q', s'): it is the same than the one in (q_2, s_2) and therefore the minimum colour seen since the i-ancestor is the minimum colour of the one seen since the i-ancestor in (q_2, s_2) (namely $\tau'(i)$) of the minimum colour visited since (q_3, s_3) and of $\Omega(q')$. As m_c is the minimum colour since (q_1, s_1) and as the i-ancestor of (q', s') must appear before (q_1, s_1) the previous minimum is the same as $\min(\tau'(i), m_c, \Omega(q'))$.
   Using exactly the same arguments, one deduces that the collapse rank in (q', s') equals $\min(m'_c, m_c, \Omega(q'))$.
   Finally the τ'_{old} information needs not to be updated.

From the previous description (and the included inductive proof) we conclude that $\mathcal{P}'$ is collapse rank-aware.

Now, in order to conclude the proof of Lemma 14, one considers the parity game $\mathbb{G}''$ on $\mathcal{P}''$ defined using the same partition as the from Q (the control state of $Q' \setminus Q$ inducing configurations with exactly one successor can be controlled by any player), and extending Ω to Q' by assigning the maximal colour to states in $Q' \setminus Q$ (hence not modifying the winner). For this game, it should then be clear that we have the desired property. □

Remark 6.6 Note that building a collapse rank-aware n-CPDS from a non-aware one increases the stack alphabet by C^{2n+1} and the state set by C^n (recall that we need extra states, that where hidden in the previous description, mainly to store τ)

Main reduction

In order to solve CPDS parity games, we give a reduction that built from an n-CPDS parity game $\mathbb{G}$ an *equivalent* $(n-1)$-CPDS parity game $\widetilde{\mathbb{G}}$. By equivalent we mean the following.

1. A configuration $(p_{\text{in}}, \bot_n)$ is winning for Éloïse in $\mathbb{G}$ iff $(p'_{\text{in}}, \bot_{n-1})$ is winning for Éloïse in $\widetilde{\mathbb{G}}$, where p'_{in} depends on p_{in} (and can be easily constructed);
2. From a winning strategy for Éloïse from $(p_{\text{in}}, \bot_n)$ in $\mathbb{G}$, one can deduce a strategy for her in $\widetilde{\mathbb{G}}$ from $(p'_{\text{in}}, \bot_{n-1})$;
3. From a winning strategy for Éloïse from $(p'_{\text{in}}, \bot_{n-1})$ in $\widetilde{\mathbb{G}}$, one can **effectively** construct a strategy for her in $\mathbb{G}$ from $(p_{\text{in}}, \bot_n)$;

Note the last two points implies the first one, and our proof will actually establish these two points. Also note that we get a similar statement for Abelard, by considering the game defined by the colouring function Ω' where we set $\Omega'(v) = \Omega(v) + 1$ for every state vertex v in the game graph.

Also note that the last point is effective. Hence, applying inductively the reduction will give an effective construction of strategies for both players in an n-CPDS game. This was not the case in the proof for 2-CPDS game in [16].

The reduction we present below (*i.e.* the description of $\widetilde{\mathbb{G}}$) can be though as a generalization of those presented in [26] and in [16]. The proof generalizes the one in [26] and partially (for the easy implication) the one in [16].

From now on we fix a collapse rank-aware n-CPDS $\mathcal{P} = \langle \Gamma, Q, \Delta \rangle$, a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q and a colouring function $\Omega : Q \to C = \{0, \dots, d\}$. The mapping computing the collapse rank is denoted $ColRk$. By $\mathbb{G}$, we denote the implied parity game, and we fix an initial configuration $(p_{\text{in}}, \bot_n)$.

For a configuration $v = (q, s)$ of $\mathcal{P}$, we define its *stack height* $sh(v)$ to be length of s seen as an n-stack. More precisely, $sh(v)$ is such that $s = [s_1 \cdots s_{sh(v)}]$.

For an infinite play $\Lambda = v_0 v_1 \cdots$, let $Steps_\Lambda$ be the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. More formally, $Steps_\Lambda = \{i \in \mathbb{N} \mid \forall j \geq i \; sh(v_j) \geq sh(v_i)\}$. Note that $Steps_\Lambda$ is always infinite and hence induces a decomposition of the play Λ into finite pieces.

For all pair $(i, j) \in Steps_\Lambda$, with $i \neq j$ and such that there is no $k \in Steps_\Lambda$ such that $i < k < j$, we define $mcol(i, j) = \min\{\Omega(v_k) \mid i \leq k \leq j\}$ and

$$kind(i, j) = \begin{cases} S & \text{if } sh(v_j) = sh(v_i) + 1 \\ B & \text{if } sh(v_j) = sh(v_i) \end{cases}$$

In the factorization induced by $Steps_\Lambda$, a factor $v_i \cdots v_j$ will be called a *bump* if $kind(i, j) = B$, and will be called a *Stair* if $kind(i, j) = S$. A bump $v_i \cdots v_j$ where $j = i + 1$ is called a *trivial* bump.

For any play Λ with $Steps_\Lambda = \{n_0 < n_1 < \cdots\}$, one defines the sequences $(mcol_i^\Lambda)_{i \geq 0} \in C^{\mathbb{N}}$ where $mcol_i^\Lambda = mcol(n_i, n_{i+1})$ and $(kind_i^\Lambda))_{i \geq 0} \in \{B, S\}^{\mathbb{N}}$ where $kind_i^\Lambda = kind(n_i, n_{i+1})$.

The sequence $(mcol_i^\Lambda)_{i \geq 0}$ fully characterizes the parity condition.

Proposition 15 *For a play Λ the following equivalences hold: Λ is winning for Éloïse iff $\liminf((mcol_i^\Lambda)_{i \geq 0})$ is even.*

The main idea to solve $\mathbb{G}$ is to build an $(n-1)$-CPDS parity game $\widetilde{\mathbb{G}}$ played on a game graph $\widetilde{\mathcal{G}}$ defined by an exponentially larger $(n-1)$-CPDS with the same set of colours. This new game *simulates* the pushdown game, in the sense that the sequences of visited colours during a correct simulation play are exactly the sequences $(mcol_i^\Lambda)_{i\geq 0}$ for plays Λ in the original game. Moreover, a play in which a player does not correctly simulate the n-CPDS game is loosing for that player.

Before providing a description of the game graph $\widetilde{\mathcal{G}}$, let us consider the following informal description of this simulation game.

In this simulation game, the players only recall an $(n-1)$-stack with some extra information. Such an $(n-1)$-stack aims at simulate a configuration in $\mathbb{G}$ whose top $(n-1)$ is that stack.

The most interesting case is when, simulating a configuration v with a control state p and a top $(n-1)$-stack s, the player owning p wants to apply a transition $push_n$ and change the control state to q. For every strategy of Éloïse, there is a (possibly empty) set of possible finite continuations of the play that will end with returning to a configuration with the same stack content as the one in v. If this happens, it means that eventually a configuration with stack height equal to $sh(v)$ is reached. Hence, in the simulation game Éloïse, is required to declare a vector $\overrightarrow{R} = (R_0, \ldots, R_d)$ of $(d+1)$ subsets in 2^Q, where R_i is the set of all states the game can be in when coming back to a configuration with stack height $sh(v)$ along these plays where in addition the smallest visited colour while the stack height is greater than $sh(v)$ is i.

Then Abelard has two options. He can continue the game by simulating the $push_n$ transition and update the control state (we call this a *pursue* move). Otherwise, he can pick a set R_i and a state $r \in R_i$, and continue the simulation from that state r (we call this a *jump* move). If he does a pursue move, the players now remember the vector $\overrightarrow{R}$ claimed by Éloïse; if, later on, a pop_n transition is simulated, a special configuration is reached where Éloïse wins if and only if the resulting state is in R_κ where κ is the smallest colour seen since the stack height was greater than $sh(v)$ (this information is encoded in the control state, reset after each pursue move and updated after each jump move. If Abelard does a jump move to a state r in R_i, the currently stored value for κ is updated to $\min(\kappa, i, \Omega(r))$, which is the smallest colour seen since the current stack level was reached).

Now we discuss the other kinds of transitions, including $collapse$, that can be simulated.

- $push_k$, pop_k, or $push^a_{1,k}$ for some $k < n$ and some $a \in \Gamma$: the simulation is an exact one and hence s is modifies and the control state is updates.
- $collapse$ if the top_1 symbol has a k-link for some $k < n$: the simulation is an exact one and hence modifies s and updates the control state.
- $push^a_{1,n}$: in this case as we only recall an $(n-1)$-stack there is no meaning of putting an n-link. Hence instead we apply a rule $push^{(a,\overrightarrow{R})}_{1,1}$, where $\overrightarrow{R}$ is the last vector declared by Éloïse (it is stored in the control state and easily retrieved). The idea behind this is that, if one simulates a $collapse$ from this new stack content, then the play is returning to the previous stack height, and hence one has to check whether the vector $\overrightarrow{R}$, that Éloïse claimed to describe the possible behaviours on returning to the previous stack level, was correct with respect to this collapse (see below). We equip this symbol with a 1-link but actually we will never allow to apply collapse from it so one could consider that we do not have a link at all here (but we did not formally allow this in our definition of CPDS).
- $collapse$ if the top_1 symbol simulates one with an n-link, equivalently is of the form $(a, \overrightarrow{R})$: in this case we check whether the new control state r reached when applying $collapse$ belongs to $R_{ColRk(a)}$ (recall that the n-CPDS $\mathcal{P}$ is collapse rank aware and that $ColRk$ computes this rank in $\mathcal{P}$). If it is the case, Éloïse wins, otherwise Abelard does.

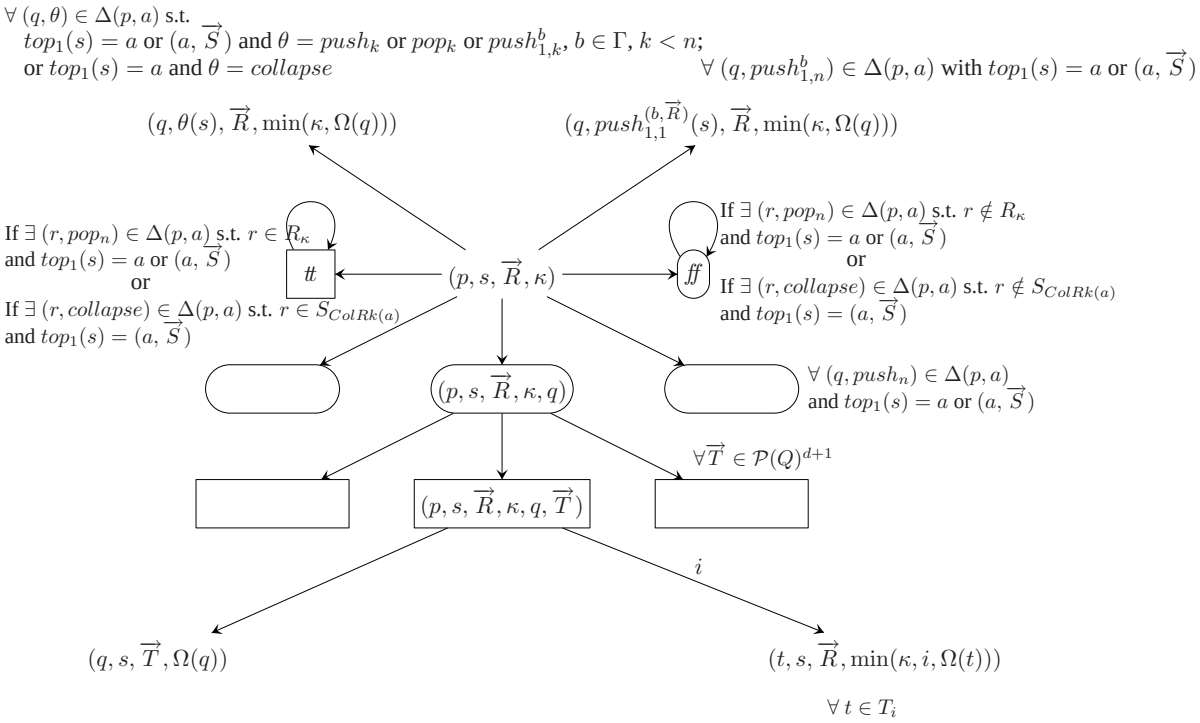


Figure 9: Local structure of $\widetilde{\mathcal{G}}$.

Therefore the main vertices of the simulation game graph are configurations of the form $(p, s, \overrightarrow{R}, \kappa)$ where s is an $(n-1)$-stack on the alphabet $\Gamma \cup (\Gamma \times \mathcal{P}(Q)^{d+1})$ and they are controlled by the player that controls p. Intermediate configurations are used to handle the previously described intermediate steps. The local structure is given in Figure 9 (circled vertices are those controlled by Éloïse). Two special vertices $\#\!\!\#$ and ff are used to simulate pop_n moves and $collapse$ moves from configurations with an n-link on their top_1 symbol. Here vertices $\#\!\!\#$ and ff are designed so that the player that controls it looses a game that reaches such a vertex: for this, $\#\!\!\#$ is assigned an even colour while ff is assigned an odd colour, and both vertices have a loop on them and no other outgoing edge.

The simulation game graph is equipped with a colouring function on the vertices and on the edges: vertices of the form $(p, s, \overrightarrow{R}, \kappa)$ have colour $\Omega(p)$, and an edge leaving from a vertex $(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{T})$ has a colour in $\{0, \ldots, d\}$ only if it simulates a bump (the colour is i iff the bump has colour i). It is easily seen that intermediate vertices can be introduced to have only colours on vertices. A precise description of the graph is given in the detailed proof of the following main result. The simulation parity game is denoted $\widetilde{\mathbb{G}}$.

Theorem 16 *A configuration $(p_{in}, \bot_n)$ is winning for Éloïse in $\mathbb{G}$ if and only if $(p_{in}, \bot_{n-1}, (\varnothing, \ldots, \varnothing), \Omega(p_{in}))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$. Moreover the game $\widetilde{\mathbb{G}}$ is an $(n-1)$-CPDS parity game.*

Proof of Theorem 16

We start with a precise description of $\widetilde{\mathcal{G}}$ and with some extra definitions. Then we give a full proof of Theorem 16

The game graph $\widetilde{\mathcal{G}}$ Let us first precisely describe the game graph $\widetilde{\mathcal{G}}$. As what follows is only a formal definition of the graph represented in Figure 9, one could skip this or refer only in case the figure is not clear enough.

- The main vertices of $\widetilde{\mathcal{G}}$ are those of the form $(p, s, \overrightarrow{R}, \kappa)$, where $p \in Q$, s is any $(n-1)$-stack (with links) on the stack alphabet $\Gamma \cup (\Gamma \times \mathcal{P}(Q)^{d+1})$, $\overrightarrow{R} = (R_0, \ldots, R_d) \in \mathcal{P}(Q)^{d+1}$ and $\kappa \in \{0, \ldots, d\}$. A vertex $(p, s, \overrightarrow{R}, \kappa)$ is reached when simulating a partial play Λ in $\mathbb{G}$ such that:
  - The last vertex in Λ has control state p and its top $(n-1)$-stack s' is such that $\pi(s')$=$\nu(s)$ where $\pi(s')$ denotes the stack obtained from s' by replacing every n-link by a 1-link and $\nu(s)$ is the stack obtained from s by replacing every symbol $(\gamma, \overrightarrow{S})$ by γ (and by preserving the link structure). This roughly means that s and s' are very slightly different representations of a same stack.
  - Éloïse claimed that she has a strategy to continue Λ in such a way that if s is eventually popped (by a pop_n action), the control state reached after popping α belongs to R_m, where m denotes the minimal colour visited since the current stack height (of the n-stack) was reached.
  - The colour κ is the smallest one since the current stack height was reached from a lower stack level.
  - Moreover, if $top_1(s)$ is of the form $(a, \overrightarrow{S})$ then the top_1 symbol of the last vertex in Λ has an n-linked and if $collapse$ is applied in the next move in Λ then Éloïse claimed that the control state of the configuration that is reached belongs to S_k where k is the collapse rank, *i.e.* $k = ColRk(a)$.

  A vertex $(p, s, \overrightarrow{R}, \kappa)$ is controlled by Éloïse if and only if $p \in Q_{\mathbf{E}}$.

- Vertices *ff* and *tt* are there to ensure the correctness of the vectors $\overrightarrow{R}$ encoded in the main vertices and of the vectors $\overrightarrow{S}$ encoded in the stack for symbols of the form $(a, \overrightarrow{S})$. Vertex *ff* is controlled by Abelard, whereas vertex of the form *tt* belongs to Éloïse. There is a loop on each of these vertices and it is the only edge from it. We assign an odd colour to *tt* and an even colour to *ff*: hence the player controlling such a vertex is loosing.

  There is a transition from some vertex $(p, s, \overrightarrow{R}, \kappa)$ to *ff*, if and only if one of the two cases happens:
  - $top_1(s) = a$ or $(a, \overrightarrow{S})$ and there is a transition rule $(r, pop_n) \in \Delta(p, a)$ such that $r \in R_\kappa$ (this means that $\overrightarrow{R}$ is correct with respect to this transition rule).
  - $top_1(s) = (a, \overrightarrow{S})$ and there is a transition rule $(r, collapse) \in \Delta(p, a)$ such that $r \in S_{ColRk(a)}$ (this means that $\overrightarrow{S}$ is correct with respect to this transition rule).

  Symmetrically, there is a transition from a vertex $(p, s, \overrightarrow{R}, \kappa)$ to vertex *tt* if and only one of the two cases happens:
  - $top_1(s) = a$ or $(a, \overrightarrow{S})$ and there is a transition rule $(r, pop_n) \in \Delta(p, \alpha)$ such that $r \notin R_\kappa$ (this means that $\overrightarrow{R}$ is not correct with respect to this transition rule).
  - $top_1(s) = (a, \overrightarrow{S})$ and there is a transition rule $(r, collapse) \in \Delta(p, a)$ such that $r \notin S_{ColRk(a)}$ (this means that $\overrightarrow{S}$ is not correct with respect to this transition rule).

- To simulate a transition rule that does not remove the topmost $(n-1)$-stack, one only has to update the control state, the component κ and apply the corresponding transformation on the stack. More precisely
  - there is a transition to $(q, \theta(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$ if $(q, \theta) \in \Delta(p, a)$, $top_1(s) = a$ or $(a, \overrightarrow{S})$ and $\theta = push_k$, pop_k or $push^b_{1,k}$ for some $b \in \Gamma$ and some $k < n$.

– there is a transition to $(q, collapse(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$ if $(q, collapse) \in \Delta(p, a)$ and $top_1(s) = a$ (this means that there is a k-link for some $k < n$ and hence the simulated collapse do not remove the top $(n-1)$-stack).

– there is a transition to $(q, push_{1,1}^{(b,\overrightarrow{R})}(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$ if $(q, push_{1,n}^{b}) \in \Delta(p, a)$ and $top_1(s) = a$ or $(a, \overrightarrow{S})$, for some $b \in \Gamma$. In this case, as we only have $(n-1)$-stacks in $\widetilde{\mathcal{G}}$, we replace the n-link by a (dumb) 1-link and annotate the symbol b by the last vector $\overrightarrow{R}$ claimed by Éloïse (if one wants to simulate collapse for b then the control state reached should be in a set described in $\overrightarrow{R}$ because the stack level reached is the one just behind the current one).

- To simulate a transition rule $(q, push_n) \in \Delta(p, a)$, the player that controls $(p, s, \overrightarrow{R}, \kappa)$ goes in $(p, s, \overrightarrow{R}, \kappa, q)$. This vertex is controlled by Éloïse who has to give a vector $\overrightarrow{T} = (T_0, \ldots, T_d) \in \mathcal{P}(Q)^{d+1}$ that describes the control states that can be reached if the play eventually comes back to the stack height just left by performing the $push_n$ action. To describe this vector, she goes to the corresponding vertex $(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{T})$.

  The vertex $(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{T})$ is controlled by Abelard who chooses either to simulate a bump or a stair. In the first case, he additionally has to pick the minimal colour in this bump. To simulate a bump with minimal colour i, he goes to a vertex $(t, s, \overrightarrow{R}, \min(\kappa, i, \Omega(t)))$, for some $t \in T_i$, through an edge coloured by i.

  To simulate a stair, Abelard goes to the vertex $(q, s, \overrightarrow{T}, \Omega(q))$.

  The last component (that stores the smallest colour seen since the currently simulated stack level was reached) has to be updated in each of these cases. After simulating a bump of minimal colour i, the minimal colour is $\min(\kappa, i, \Omega(s))$. After simulating a stair, the colour has to be initialized (since a new stack height is simulated). Its value, is therefore $\Omega(q)$, which is the unique colour since the (new) stack level was reached.

The only vertices that are coloured are those of the form $(p, s, \overrightarrow{R}, \kappa)$ and the colour of such a vertex is $\Omega(p)$. Some edges are also coloured. See Figure 9 for details.

Remark 6.7 In the definition of parity games we were requiring to have a total colouring function working only on vertices. For this, one can add extra intermediate states and introduce a new colour larger than d to fit the definition without changing the issue of that game.

Finally the following fact is easily checked.

Property 17 *The game graph $\widetilde{\mathcal{G}}$ is generated by a $(n-1)$-CPDS.*

Proof It is immediate. The only point to note is that the vertices *tt* and *ff* can be simulated using a control state where the CPDS is looping. □

Factorization of a play in $\widetilde{\mathbb{G}}$ Recall that in $\widetilde{\mathcal{G}}$ some edges are coloured. Hence, to represent a play, we have to encode this information on edge colouring. A play will be represented as a sequence of vertices together with colours in $\{0, \ldots, d\}$ that correspond to colours appearing on edges.

For any play in $\widetilde{\mathbb{G}}$, a *round* is a factor between two visits through vertices of the form $(p, s, \overrightarrow{R}, \kappa)$. We have the following possible forms for a round:

- The round is of the form $(p, s, \overrightarrow{R}, \kappa)(q, \theta(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$ and corresponds therefore to the simulation of a trivial bump.
- The round is of the form $(p, s, \overrightarrow{R}, \kappa)(p, s, \overrightarrow{R}, \kappa, q)(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{T})i(t, s, \overrightarrow{R}, \min(\kappa, i, \Omega(t)))$ and corresponds therefore to the simulation of a rule $push_n$ followed by a sequence of moves that ends by coming back to the former stack level. Moreover the minimal colour in this sequence of moves is i.
- The round is of the form $(p, s, \overrightarrow{R}, \kappa)(p, s, \overrightarrow{R}, \kappa, q)(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{T})i(q, s, \overrightarrow{T}, \Omega(q))$ and corresponds therefore to the simulation of a rule $push_n$ leading to a new stack height below which the play will never go. We designate it has a *stair*.

For any play $\lambda = v_0 v_1 v_2 \cdots$ in $\widetilde{\mathbb{G}}$, we consider the subset of indices corresponding to vertices of the form $(p, s, \overrightarrow{R}, \kappa)$. More precisely:

$$Rounds_\lambda = \{n \in \mathbb{N} \mid v_n = (p, s, \overrightarrow{R}, \kappa), \text{ for some } p, s, \overrightarrow{R}, \kappa\}$$

Therefore, the set $Rounds_\lambda$ induces a natural factorization of λ into rounds.

Definition 6.8 (Rounds factorization) For a (possibly partial) play $\lambda = v_0 v_1 v_2 \cdots$, we call *rounds factorization* of λ, the (possibly finite) sequence $(\lambda_i)_{i \geq 0}$ of rounds λ defined as follows. Let $Rounds_\lambda = \{n_0 < n_1 < n_2 < \cdots\}$, then for all $0 \leq i < |Rounds_\lambda|$, $\lambda_i = v_{n_i} \cdots v_{n_{i+1}}$.

Therefore, for every $i \geq 0$, the first vertex in λ_{i+1} equals the last one in λ_i. Moreover, $\lambda = \lambda_1 \odot \lambda_2 \odot \lambda_3 \odot \cdots$, where $\lambda_i \odot \lambda_{i+1}$ denotes the concatenation of λ_i with λ_{i+1} without its first vertex.

Finally, the *colour* of a round is the smallest colour in $\{0, \ldots, d\}$ appearing in the round.

In order to prove both implications of Theorem 16, we build from a winning strategy for Éloïse in one game a winning strategy for her in the other game. The main argument to prove that the new strategy is winning is to prove a correspondence between the factorizations of plays in both games.

Direct implication Assume that the configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}$, and let Φ be a corresponding strategy for her.

Using Φ, we define a strategy φ for Éloïse in $\widetilde{\mathbb{G}}$ from $(p_{in}, \bot_{n-1}, (\varnothing, \ldots, \varnothing), \Omega(p_{in}))$. Strategy Φ stores a partial play in $\mathbb{G}$, that is an element in V^* (where V denotes the set of vertices of $\mathcal{G}$). This memory will be denoted by Λ. At the beginning Λ is initialized to the vertex $(p_{in}, \bot_n)$. We first describe φ, and then we explain how Λ is updated. Both the strategy φ and the update of Λ, are described for a round.

Choice of the move. Assume that the play is in some vertex $(p, s, \overrightarrow{R}, \kappa)$ for $p \in Q_{\mathbf{E}}$. The move given by φ depends on $\Phi(\Lambda)$:

- If $\Phi(\Lambda) = (r, pop_n)$, then Éloïse goes to $\#$ (Proposition 18 will prove that this move is always possible).
- If $\Phi(\Lambda) = (r, collapse)$ and $top_1(s)$ is of the form $(a, \overrightarrow{S})$, then Éloïse goes to $\#$ (Proposition 18 will prove that this move is always possible).
- If $\Phi(\Lambda) = (q, \theta)$, for some $\theta = push_k$, pop_k or $push^b_{1,k}$ with $b \in \Gamma$ and $k < n$, or $\theta = collapse$ and $top_1(s) \in \Gamma$, then Éloïse goes to $(q, \theta(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$.
- If $\Phi(\Lambda) = (q, push^b_{1,n})$, then Éloïse goes to $(q, push^{(b, \overrightarrow{R})}_{1,1}(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$.

- If $\Phi(\Lambda) = (q, push_n)$, then Éloïse goes to $(p, s, \overrightarrow{R}, \kappa, q)$.

In this last case, or in the case where $p \in Q_{\mathbf{A}}$ and Abelard goes to $(p, s, \overrightarrow{R}, \kappa, q)$, we also have to explain how Éloïse behaves from $(p, s, \overrightarrow{R}, \kappa, q)$. She has to provide a vector $\overrightarrow{T} \in \mathcal{P}(Q)^{d+1}$ that describes which states can be reached if the play eventually comes back to the previous stack height, depending on the minimal colour visited in the meantime. In order to define $\overrightarrow{T}$, she considers the set of all possible continuation of $\Lambda \cdot (q, push_n(\sigma))$ (where (p, σ) denotes the last vertex of Λ) where she respects her strategy Φ. For each such play, she checks whether some configuration of the form (s, σ') with $sh(\sigma') = sh(\sigma)$ is visited after $\Lambda \cdot (q, push_n(\sigma))$, that is if the new top $(n-1)$-stack is eventually removed (note that it could be due either to a pop_n action either to a $collapse$). If it is the case, she considers the first such configuration (s, σ') and the smallest colour i visited between $(q, push_n(\sigma))$ (included) and (s, σ') (excluded). For every $i \in \{0, \dots d\}$, T_i is exactly the set of states $s \in Q$ such that the preceding case happens. Finally, we set $\overrightarrow{T} = (T_0, \dots, T_d)$ and Éloïse moves to $(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{T})$.

Update of Λ. The memory Λ is updated after each visit to a vertex of the form $(p, s, \overrightarrow{R}, \kappa)$. We have three cases depending on the kind of the round:

- The round is a trivial bump and therefore a transition (q, θ) where θ is of the form $push_k$, pop_k, $push^b_{1,k}$, $push^b_{1,n}$ or $collapse$ was simulated. Let (p, σ) be the last vertex in Λ, then the updated memory is $\Lambda \cdot (q, \theta(\sigma))$.
- The round is a bump. Therefore a bump of colour i (where i is the colour of the round) starting with some transition $(q, push_n)$ and ending in a state $s \in S_i$ was simulated. Let (p, σ) be the last vertex in Λ. Then the memory becomes Λ extended by $(q, push_n(\sigma))$ followed by a sequence of moves, where Éloïse respects Φ, that ends by some configuration (s, σ') with $sh(\sigma) = sh(\sigma')$ while having i as smallest colour. By definition of S_i such a sequence of moves always exists.
- The round is a stair and therefore we have simulated the transition $(q, push_n)$. If (p, σ) denotes the last vertex in Λ, then the updated memory is $\Lambda \cdot (q, push_n(\sigma))$.

Therefore, with any partial play λ in $\widetilde{\mathbb{G}}$ in which Éloïse respects her strategy φ, is associated a partial play Λ in $\mathbb{G}$. An immediate induction shows that Éloïse respects Φ in Λ. The same arguments works for an infinite play λ, and the corresponding play Λ is therefore infinite, starts from $(p_{in}, \bot_n)$ and Éloïse respects Φ in that play. Therefore she wins in Λ.

The following proposition is a direct consequence of how φ was defined.

Proposition 18 *Let λ be a partial play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot_{n-1}, (\varnothing, \dots, \varnothing), \Omega(p_{in}))$, ends in a vertex of the form $(p, s, \overrightarrow{R}, \kappa)$, and where Éloïse respects φ. Let Λ be the play associated to λ built by the strategy φ. Then the following holds:*

1. *Λ ends in a vertex of the form (p, σ) where $\pi(top_n(\sigma)) = \nu(s)$.*[17]
2. *κ is the smallest visited colour in Λ since the last configuration with a stack height strictly smaller than $sh(\sigma)$.*
3. *Assume that Λ is extended, that Éloïse keeps respecting Φ and that the next move after (p, σ) is to apply a transition (r, pop_n). Then $r \in R_\kappa$.*

[17] Recall that $\pi(\sigma')$ denotes the stack obtained from σ' by replacing every n-link by a 1-link and $\nu(s)$ is the stack obtained from s by replacing every symbol $(\gamma, \overrightarrow{S})$ by γ (and by preserving the link structure).

4. *Assume that Λ is extended, that Éloïse keeps respecting Φ, that the next move after (p, σ) is to apply a transition $(r, collapse)$ and that $top_1(\sigma)$ has an n-link. Then $top_1(s) = (a, \overrightarrow{S})$ and $r \in S_{ColRk(a)}$.*

Remark 6.9 Proposition 18 implies that the strategy φ is well defined when it provides a move to $\#$. Moreover, one can deduce that, if Éloïse respects φ, ff is never reached.

For infinite plays that do not reach $\#$, using the definitions of $\widetilde{\mathcal{G}}$ and φ, we easily deduce the following proposition.

Proposition 19 *Let λ be an infinite play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot_{n-1}, (\varnothing, \ldots, \varnothing), \Omega(p_{in}))$, and where Éloïse respects φ. We additionally suppose that λ never reaches the vertex $\#$. Let Λ be the associated play built by the strategy φ. Let $(\lambda_i)_{i\geq 0}$ be the round factorization of λ. Then, for every $i \geq 1$ the following hold:*

1. *λ_i is a bump if and only if $kind_i^\Lambda = B$.*
2. *λ_i has colour $mcol_i^\Lambda$.*

Proposition 19 implies that for any infinite play λ in $\widetilde{\mathbb{G}}$ starting from $(p_{in}, \bot_{n-1}, (\varnothing, \ldots, \varnothing), \Omega(p_{in}))$ where Éloïse respects φ, if λ never reaches $\#$ then the minimum colour infinitely often visited in λ is $\liminf((mcol_i^\Lambda)_{i\geq 0})$ for the corresponding play Λ in $\mathbb{G}$. Hence, using Proposition 15 we conclude that λ is winning if and only if Λ is winning. As Λ is winning for Éloïse, it follows that λ is also winning for her. Hence φ is a winning strategy for Éloïse, as any play where she respects it either reaches $\#$ (and therefore satisfies the parity condition) or satisfies the parity condition (by proposition 15 and 19).

Converse implication For the converse implication, one could adapt the proof of the direct implication to show that if Abelard has a winning strategy in $\mathbb{G}$ he also has one in $\widetilde{\mathbb{G}}$. The construction is the same as for Éloïse except that now Abelard has to decide whether to do a pursue move or a jump move. For this he computes his own vector $\overrightarrow{T}'$ (as Éloïse was doing in the previous construction) and checks for consistency with the one ($\overrightarrow{T}$) claimed by Éloïse: if they are consistent (*i.e.* there is some $t \in T_i \cap T'_i$ for some i) he does a jump move for this t, otherwise he does a pursue move. One can check that this strategy is winning for him in $\widetilde{\mathbb{G}}$.

Here we give a different proof, that builts a winning strategy in $\mathbb{G}$ from one in $\widetilde{\mathbb{G}}$. This is more involved than the proof sketched just above, but it has the major interest to provide an effective construction for strategies in CPDS parity games.

Assume now that Éloïse has a winning strategy φ in $\widetilde{\mathbb{G}}$ from $(p_{in}, \bot_{n-1}, (\varnothing, \ldots, \varnothing), \Omega(p_{\text{in}}))$. Using φ, we build a strategy Φ for Éloïse in $\mathbb{G}$ for plays starting from $(p_{in}, \bot_n)$.

Strategy Φ uses, as memory, a (1-) stack Π, to store the complete description of a play in $\widetilde{\mathbb{G}}$. Recall here that a play in $\widetilde{\mathbb{G}}$ is represented as a sequence of vertices together with colours in $\{0, \ldots d\}$.

Therefore the stack alphabet of Π is the set of vertices of $\widetilde{\mathcal{G}}$ together with $\{0, \ldots, d\}$. In the following, $top(\Pi)$ will denote the top stack symbol of Π while $StCont(\Pi)$ will be the word obtained by reading Π from bottom to top (without considering the bottom-of-stack symbol of Π). In any play where she respects Φ, $StCont(\Pi)$ will be a play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot_{n-1}, (\varnothing, \ldots, \varnothing), \Omega(p_{\text{in}}))$ and where Éloïse respects her winning strategy φ. Moreover, for any play Λ where Éloïse respects Φ, we will always have that $top(\Pi) = (p, s, \overrightarrow{R}, \kappa)$ if and only if the current configuration (p, σ) is such that $\pi(top_n(\sigma)) = \nu(s)$. Finally, if Éloïse keeps respecting Φ, and if the next move is to a configuration with a stack height smaller than $sh(\sigma)$, then its control state will be in R_κ if the configuration is reached by applying a pop_n action, and will be in $S_{ColRk(a)}$ where $(a, \overrightarrow{S}) = top_1(s)$. Initially, Π only contains $(p_{\text{in}}, \bot_{n-1}, (\varnothing, \ldots, \varnothing), \Omega(p_{\text{in}}))$.

In order to describe Φ, we assume that we are in some configuration (p, σ) and that $top(\Pi) = (p, s, \overrightarrow{R}, \kappa)$. We first describe how Éloïse plays if $p \in Q_{\mathbf{E}}$, and then we explain how the Π is updated.

- **Choice of the move.** Assume that $p \in Q_{\mathbf{E}}$ and that Éloïse has to play from some vertex (p, σ). For this, she considers the value of φ on $StCont(\Pi)$.

  If it is a move to $\not\pitchfork$, she plays an action (r, pop_n) for some state $r \in R_\kappa$ if exists or she play $(r, collapse)$ for some $r \in S_{ColRk(a)}$ where $top_1(s) = (a, \overrightarrow{S})$. Lemma 20 will prove that such a move always exists.

  If the move given by φ is to go to some vertex $(q, \theta(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$ with $\theta = push_k$, pop_k or $push^b_{1,k}$ for some $k < n$ and $b \in \Gamma$, she plays the transition (q, θ).

  If the move given by φ is to go to some vertex $(q, push^{b,\overrightarrow{R}}_{1,1}(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$, then she applies the transition $(q, push^b_{1,n})$.

  If the move given by φ is to go to some vertex $(p, s, \overrightarrow{R}, \kappa, q)$, then Éloïse applies the transition $(q, push_n)$.

- **Update of Π.** Assume that the last move, played by Éloïse or Abelard, was to go from (p, σ) to some configuration (r, σ') with $sh(\sigma') < sh(\sigma)$ (*i.e.* the move was either a pop_n or a $collapse$ involving an n-link).

  – If the move is a pop_n one, then Éloïse pops in Π until she finds some configuration of the form $(p', s', \overrightarrow{R'}, \kappa', p'', \overrightarrow{R''})$ that is not preceded by a colour in $\{0, \dots, d\}$. This configuration is therefore in the stair that simulates the step where the stack level $sh(\sigma)$ is reached. Eve updates Π by pushing κ in Π followed by $(r, s', \overrightarrow{R'}, \min(\kappa', \kappa, \Omega(r)))$.

  – If the move is a $collapse$ one, then let $d = sh(s) - sh(s')$ (d denotes the decrease of the stack height). Then Éloïse pops in Π until she finds some configuration of the form $(p', s', \overrightarrow{R'}, \kappa', p'', \overrightarrow{R''})$ that is not preceded by a colour in $\{0, \dots, d\}$. If $d = 1$ she stops otherwise she keeps popping until she finds another such configuration (and so on until she found d such configurations). Let $(p', s', \overrightarrow{R'}, \kappa', p'', \overrightarrow{R''})$ be the d-th such configuration (this configuration is therefore in the stair that simulates the step where the stack level $sh(\sigma')$ was left). Eve updates Π by pushing $ColRk(a)$, where $(a, \overrightarrow{S}) = top_1(s)$ in Π followed by $(r, s', \overrightarrow{R'}, \min(\kappa', ColRk(a), \Omega(r)))$. Note that we actually have $\overrightarrow{R''} = \overrightarrow{S}$.

  Assume that the last move, played by Éloïse or Abelard, was to go from (p, σ) to some configuration (q, σ') with $sh(\sigma) = sh(\sigma')$: hence the simulated move is (q, θ) for some $\theta = push_k$, pop_k, $push^b_{1,k}$ or $push^b_{1,n}$ for some $k < n$ and $b \in \Gamma$. Then Éloïse updates Π by pushing $(q, \theta(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$ if $\theta \neq push^b_{1,n}$ and $(q, push^{(b,\overrightarrow{R})}_{1,1}(s), \overrightarrow{R}, \min(\kappa, \Omega(q)))$ otherwise.

  Finally, assume that the last move, played by Éloïse or Abelard, was to go from (p, σ) to some configuration (q, σ') with $sh(\sigma') = sh(\sigma)+1$, let $(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{S}) = \varphi(StCont(\Pi) \cdot (p, s, \overrightarrow{R}, \kappa, q))$. Intuitively, $\overrightarrow{S}$ describes which states Éloïse can ensure to reach if a configuration with stack height $sh(s)$ is eventually reached (while not visiting a configuration of lower stack height). Éloïse updates Π by successively pushing $(p, s, \overrightarrow{R}, \kappa, q)$, $(p, s, \overrightarrow{R}, \kappa, q, \overrightarrow{S})$, and $(q, s, \overrightarrow{S}, \Omega(q))$.

The following lemma gives the meaning of the information stored in Π.

Lemma 20 *Let Λ be a partial play in $\mathbb{G}$, where Éloïse respects Φ, that starts from $(p_{in}, \bot_n)$ and that ends in a configuration (p, σ). We have the following facts:*

1. $top(\Pi) = (p, s, \overrightarrow{R}, \kappa)$ *with* $\overrightarrow{R} \in \mathcal{P}(Q)^{d+1}$, $0 \leq \theta \leq d$ *and* $\pi(top_{n-1}(\sigma)) = \nu(s)$.
2. $StCont(\Pi)$ *is a partial play in* $\widetilde{\mathbb{G}}$ *that starts from* $(p_{in}, \bot_n, (\varnothing, \ldots, \varnothing), \Omega(p_{in}))$, *that ends with* $(p, s, \overrightarrow{R}, \kappa)$ *and where Éloïse respects* φ.
3. κ *is the smallest colour visited since the stack height is greater or equal than* $sh(\sigma)$.
4. *If* Λ *is extended by some move* (r, pop_n), *then* $r \in R_\theta$.
5. *If* Λ *is extended by some move* $(r, collapse)$ *and if* $top_1(\sigma)$ *has an* n*-link, then* $top_1(s) = (a, \overrightarrow{S})$ *and the control state in the new configuration belongs to* $S_{ColRk(a)}$.

Proof The proof goes by induction on Λ. We first show that the fourth point is a consequence of the second and third points. Assume that the next move after (p, σ) is to apply an action (r, pop_n). The second point implies that $(p, s, \overrightarrow{R}, \kappa)$ is winning for Éloïse in $\widetilde{\mathbb{G}}$. If $p \in Q_{\mathbf{E}}$, by definition of Φ, there is some edge from that vertex to *tt*, which means that $r \in R_\theta$ and allows us to conclude. If $p \in Q_{\mathbf{A}}$, note that there is no edge from $(p, s, \overrightarrow{R}, \kappa)$ (winning position for Éloïse) to the loosing vertex *ff*. Hence we conclude the same way.

We now show that the fifth point is a consequence of the second and third points. Assume that the next move after (p, σ) is to apply an action $(r, collapse)$ and that it involves an n-link. The second point implies that $(p, s, \overrightarrow{R}, \kappa)$ is winning for Éloïse in $\widetilde{\mathbb{G}}$. If $p \in Q_{\mathbf{E}}$, by definition of Φ, there is some edge from that vertex to *tt*, which means that $r \in S_{ColRk(a)}$ and allows us to conclude. If $p \in Q_{\mathbf{A}}$, note that there is no edge from $(p, s, \overrightarrow{R}, \kappa)$ (winning position for Éloïse) to the loosing vertex *ff*. Hence we conclude the same way.

Let us now prove the other points. For this, assume that the result is proved for some play Λ, and let Λ' be some extension of Λ. We have several cases, depending on how Λ' extends Λ:

- Λ' is obtained by applying a rule of type $push_k$, pop_k or $push^b_{1,k'}$ for some $k < n$, $k' \leq n$ and $b \in \Gamma$. The result is trivial in that case.
- Λ' is obtained by applying a $collapse$ rule involving a k-link for some $k < n$. The result is also immediate in this case.
- Λ' is obtained by applying a pop_n rule. Let (p, σ) be the last configuration in Λ, and let $\overrightarrow{R}$ be the last vector in $top(\Pi)$ when being in configuration (p, σ). By induction hypothesis, it follows that $\Lambda' = \Lambda \cdot (r, \sigma')$ is such that $r \in R_\theta$. Considering how Π is updated, and using the fourth point, we easily deduce that the new strategy stack Π is as desired.
- Λ' is obtained by applying a $collapse$ rule involving an n-link. Let (p, σ) be the last configuration in Λ, and let $(a, \overrightarrow{S})$ be $top_1(\Pi)$ when being in configuration (p, σ). By induction hypothesis, it follows that $\Lambda' = \Lambda \cdot (s, \sigma')$ is such that $s \in S_{ColRk(a)}$. Considering how Π is updated, and using the fifth point, we easily deduce that the new strategy stack Π is as desired.

□

Now, the following result is an easy consequence of the previous lemma.

Lemma 21 *Let* Λ *be a partial play in* $\mathbb{G}$ *starting from* $(p_{in}, \bot_n)$ *and where Éloïse respects* Φ. *Let* $\lambda = StCont(\Pi)$, *where* Π *denotes the strategy stack in the last vertex of* Λ. *Let* $(\lambda_i)_{i=0,\ldots,k}$ *be the round factorization of* λ. *Then the following holds:*

- λ_i *is a bump if and only if* $kind_i^\Lambda$ *is a bump.*

- λ_i *has colour* $mcol_i^{\Lambda}$.

Both lemmas 20 and 21 are for partial plays. A version for infinite plays would allow to conclude. Let Λ be an infinite play in $\mathbb{G}$. We define an infinite version of λ by considering the limit of the stack contents $(StCont(\Pi_i))_{i\geq 0}$ where Π_i is the strategy stack after the i-th first moves in Λ. It is easily seen that such a limit exists, is infinite and corresponds to a play won by Éloïse in $\widetilde{\mathbb{G}}$. Moreover the results of Lemma 21 apply.

Let Λ be a play in $\mathbb{G}$ with initial vertex $(p_{\text{in}}, \bot_n)$, and where Éloïse respects Φ, and let λ be the associated infinite play in $\widetilde{\mathbb{G}}$. In particular λ is won by Éloïse. Thus, using Lemma 21 and Proposition 15, we conclude, as in the direct implication, that Λ is winning.

Complexity issues

The following Theorem gives the decidability and complexity bound of Theorem 13.

Theorem 22 *Solving an n-CPDS parity game is an n-EXPTIME complete problem.*

Proof The proof is by induction on n. Case $n = 1$ is the one of pushdown parity games [26]. Assume the result holds for $n-1$ and consider an n-CPDS parity game given $\mathbb{G}$ by an n-CPDS $\mathcal{P} = \langle \Gamma, Q, \Delta \rangle$ for some $n > 1$. In a first step one transforms $\mathbb{G}$ into an equivalent game generated by a collapse rank-aware n-CPDS. By remark 6.6 this increases both the states set and the stack alphabet by a factor in $|C|^{\mathcal{O}(n)}$ (C is the set of colors). Then one solves the game $\widetilde{\mathbb{G}}$ and concludes. Solving $\widetilde{\mathbb{G}}$ can be achieved in $(n-1)$-EXPTIME and the game $\widetilde{\mathbb{G}}$ is built from a CPDS with a states set of size $2^{\mathcal{O}(|Q|^2|C|^{\mathcal{O}(n)})}$ and a stack alphabet of size $\mathcal{O}(|\Gamma|2^{|Q|}C|^{\mathcal{O}(n)})$. Hence solving $\mathbb{G}$ can be achieved in n-EXPTIME.

Hardness follows from hardness for solving a parity game played on a (non-collapsible) higher order pushdown graph. A self content proof of this result was established by Thierry Cachat and Igor Walukiewicz, but was unfortunately not published.

Here we sketch another proof of this result that relies on the following result: checking emptiness of a nondeterministic higher-order pushdown automaton of order n is an $(n-1)$-EXPTIME complete problem [12]. Trivially this result is still true if we assume that the input alphabet is reduced to a single letter. The following result is also proved in [12]: checking emptiness of an alternating higher-order pushdown automaton of order n is an n-EXPTIME complete problem. Nevertheless note that this last result does not imply directly hardness for games on higher order pushdown graphs (because in general it is more *difficult* to check emptiness for an alternating device than to solve a reachability game on the corresponding class of graphs[18]: the problems are trivially equivalent only when considering infinite words on a single letter alphabet).

Now consider an order-$(n+1)$ nondeterministic higher order pushdown automaton $\mathcal{A}$ whose input alphabet is reduced to a single letter. The language accepted by $\mathcal{A}$ is non-empty if and only if there is a path from the initial configuration of $\mathcal{A}$ to a final configuration of $\mathcal{A}$ in the transition graph G of $\mathcal{A}$. Equivalently the language accepted by $\mathcal{A}$ is non-empty if and only if Éloïse wins the reachability game $\mathbb{G}$ over G where she controls all vertices (and where the play starts from the initial configuration of $\mathcal{A}$ and where final vertices are those corresponding to final configurations of $\mathcal{A}$). Now consider the reduction used to prove Theorem 13 and apply it to $\mathbb{G}$: it leads to an equivalent reachability game $\widetilde{\mathbb{G}}$ that is now played on the transition graph of an order-n higher order pushdown automaton. In the new game graph, the main vertices are of the form $(p, s, \overrightarrow{R}, \kappa)$: here $\overrightarrow{R}$ is actually a pair (R_0, R_1) (we consider a reachability condition) and κ is either 0 or 1. The important fact is

[18] As an example: solving a reachability game on a finite graph is in P while checking emptiness for an alternating automata on finite word (even if one considers a 1 letter alphabet) is PSPACE-complete

that R_0 and R_1 can be forced to be singletons: this follows from the fact that all vertices in $\mathbb{G}$ are controlled by Éloïse (and then from the proof's details). Therefore, one concludes that the size of the game graph associated with $\widetilde{\mathbb{G}}$ is polynomial in the size of $\mathcal{A}$. Hence, one has shown the following: checking emptiness for an order-$(n+1)$ nondeterministic higher order pushdown automaton whose input alphabet is reduced to a single letter can be polynomially reduced to solve a reachability game over the transition graph of an order-n higher order pushdown automaton. In conclusion, this last problem is n-EXPTIME hard (and actually n-EXPTIME-complete). □

Strategies

We finally focus on winning strategies. From the proof of the converse implication of Theorem 16 one can infer that it is possible to construct effectively strategies for both player in a parity CPDS game. Here, we give a more precise statement by providing a precise information on the memory needed in such a game. Recall that for regular games on finite graph, it is well know since Büchi Landweber seminal paper that a finite automaton with output suffices to represent winning strategies. For pushdown parity game, Walukiewicz has shown in [26] that pushdown automaton with output suffices to compute a winning strategy and this result was then extended (without proof) by Thierry Cachat [3] who showed that higher-order pushdown automata with output (of order n) allow to compute winning strategy for higher order pushdown game (of order n). Here we extend this last result to CPDS parity games, which terminates the proof of Theorem 13.

Theorem 23 *In an n-CPDS parity game one can build, for the player having a winning strategy from a given configuration with an empty stack, a (deterministic) n-CPDA with output* [19] *realizing a winning strategy. Moreover the stack used by the strategy automaton has exactly the same structure*[20] *has the one in the game.*

Proof The following proof is not totally formal. Indeed, we believe that a totally formal and detailed proof would be very hard to understand (and to write) and we expect the following sketch to be rather convincing.

First note that in the proof of Lemma 14, if one has an n-CPDA with output realizing a winning strategy in the rank-aware game one also has an n-CPDA with output realizing a winning strategy in the non-aware game.

Hence it suffices to consider the proof of the converse implication of Theorem 16 and infers that if the winning strategy φ in $\widetilde{\mathbb{G}}$ is realized by an $(n-1)$-CPDA with output, the strategy Φ in $\mathbb{G}$ can be realized by an n-CPDA with output.

Recall that strategy Φ was using a stack Π to store plays in $\widetilde{\mathbb{G}}$. Then it considers the value of φ on that stack content to decide which move to play. Hence, if strategy φ can be realized by an $(n-1)$-CPDA with output, one can instead of representing a play in Π represent the sequence of memory values used by φ: the resulting structure is an n-stack (stack of $(n-1)$-stack). Nevertheless there is still some work to do. Indeed the new strategy stack (that we also denote Π) has not the same shape as the one in the game: this is mainly because in the former Π we where storing all bumps which may cause the stack to be much larger than the one in the game $\mathbb{G}$. Nevertheless, we only need the information on the last vertex of a bump, and never need to retrieve information on an intermediate vertex in previous bumps in the current stack level; actually the only important information is the one concerning vertices corresponding to stack height such that no vertex later has a stack height smaller or *equal* to this one. Hence after performing a bump we can forgot about the information on the initial vertex of the bump and only recall the one on the last vertex of the bump. Therefore, the resulting strategy stack (again denoted Π) has the same shape (except for n-links) than the current stack in the game. Moreover

[19] An n-CPDA with output is defined as a CPDA except that the transition function also provide a symbol (in a specific alphabet – here one describing moves in the game) to be output for every transition

[20] Here we mean that if one replaces every symbol in the CPDA stack and in the strategy stack by a fixed new symbol then the two resulting stacks are the equal

the update after a pop_n move is now very easy, as one simply needs to perform a pop_n in Π (recall that we no longer keep information on intermediate nodes appearing on successive bumps on a same stack level). It remains to explain how the update can be handled for a $collapse$ involving an n-link. This is very simple: one only has to attach n-links in Π when simulating a $push^a_{1,n}$ action and follow this link if a collapse is applied later. Note that now the stack Π and the one in the game have the same shape. □

6.2 Extensions, consequences

Solving games with an ω-regular winning condition

Theorem 13 can easily be generalized to the case where on considers on ω-regular winning condition.

In order to define such a winning condition, we may assume that the collapsible pushdown graph comes with an edge labeling. More precisely we start with an order-n collapsible pushdown system with input

$$\mathcal{A} = \langle \Gamma, Q, A, \Delta \subseteq Q \times \Gamma \times A \times Q \times Op_n, q_0 \in Q \rangle$$

where Γ is the stack alphabet, Q is a finite set of control states, A is an input alphabet, Δ is the transition relation and q_0 is the initial state. We require Δ to respect the standard convention that $\perp$ cannot be pushed onto or popped from the stack. Configurations are defined as for the case without input. Now the one-step transition relation $\rightarrow$ is the union of the one-step transition relations $\rightarrow^a$ (where a ranges over A) defined by $(q, s) \rightarrow^a (q', s')$ iff for some $\theta \in Op_n$, we have $(q, top_1 s, a, q', \theta) \in \Delta$, and $s' = \theta(s)$. Then the configuration graph of $\mathcal{A}$ is defined as the graph whose vertices are the reachable configurations of $\mathcal{A}$ (defined as previously) and the edge relation is the relation $\rightarrow$ restricted to the reachable configurations. As for the unlabeled case, we partition the control states into E-states and A-states, and this extends naturally to a partition of the vertices of the CPD graph. Finally, the winning condition is given by an ω-regular language $\Omega \subseteq A^\omega$, and a play is winning for Éloïse iff its underlying A-labeling belongs to Ω.

In order to prove the decidability of games over collapsible pushdown graphs with an ω-regular winning condition we give a more general reduction result.

Proposition 24 *Let $\mathbb{G}$ be a game over some A-labeled graph $G = (V, E \subseteq V \times A \times V)$ with an ω-regular winning condition Ω. Then one can define a parity game $\mathbb{G}'$ over a graph $G' = (V \times S, E')$ for some set S containing a distinguished state s_0 such that the following holds: for any vertex $v \in V$, Éloïse wins from v in $\mathbb{G}$ if and only if she wins from (v, s_0) in $\mathbb{G}'$.*

Moreover, from a winning strategy from (v, s_0) in $\mathbb{G}'$, one can effectively build a corresponding winning strategy from v in $\mathbb{G}$ that only requires a finite amount of extra memory.

Proof The main idea of the construction is to consider a deterministic parity automaton accepting Ω and to plug it in G in order to compute on the fly the unique run of it on the labeling of the current play.

Let $\mathcal{S} = \langle S, A, \delta, s_0, \rho \rangle$ be a *deterministic* parity automaton accepting Ω: S is a finite set of control states, A is an input alphabet, $\delta : Q \times A \rightarrow Q$ is a transition function and $\rho : Q \rightarrow \mathbb{N}$ is a priority function. We define $G' = (V \times S, E')$ to be such that

$$E' = \{((v_1, s_1)(v_2, s_2)) \mid \exists a \in A \text{ s.t. } (v_1, a, v_2) \in E \text{ and } \delta(s_1, a) = s_2\}$$

Let $V_E \cup V_A$ be the partition of the vertices in $\mathbb{G}$. Then one considers the partition $V_E \times S \cup V_A \times S$ and defines $\mathbb{G}'$ to be the parity game defined on G' equipped with this partition together with the priority function assigning to any vertex (v, s) the priority $\rho(s)$.

Assume that Éloïse has a winning strategy in $\mathbb{G}'$ from some vertex (v, s_0). Let $\varphi' : (V \times S)^* \to (V \times S)$ be such a winning strategy [21]. From φ' we define a winning strategy φ for Éloïse in $\mathbb{G}$. First note that as G is edge-labeled, a play is now an infinite sequence of the form $(v_0, a_0, v_1)(v_1, a_1, v_2)(v_2, a_2, v_3)\cdots$ and hence a strategy is a mapping from $(V \times A \times V)^*$ into $(V \times A \times V)$. Let $\lambda = (v_0, a_0, v_1)(v_1, a_1, v_2)\cdots(v_{k-1}, a_{k-1}, v_k)$ be some partial play ending in a vertex v_k controlled by Éloïse. Let $s_0 s_1 \cdots s_k$ be the (unique) run of $\mathcal{S}$ on $a_0 a_1 \cdots a_{k-1}$. Then it easily follows that $\lambda' = (v_0, s_0)(v_1, s_1)\cdots(v_k, s_k)$ is a partial play in $\mathbb{G}'$ and hence $\varphi'(\lambda') = (v_{k+1}, s_{k+1})$ is well defined. As $((v_k, s_k), (v_{k+1}, s_{k+1})) \in E'$ then there is at least one $a_k \in A$ such that $(v_k, a_k, v_{k+1}) \in E$. Let us pick any such a_k and set $\varphi(\lambda) = (v_k, a_k, v_{k+1})$. The strategy φ is trivially well-defined, and we claim that it is wining. Indeed, consider an infinite play $\lambda = (v_0, a_0, v_1)(v_1, a_1, v_2)(v_2, a_2, v_3)\cdots$ in $\mathbb{G}$ where Éloïse respects φ. From how φ was defined, we deduce that if $s_0 s_1 s_2 \cdots$ is the unique run of $\mathcal{S}$ on $a_0 a_1 a_2 \cdots$, then $\lambda' = (v_0, s_0)(v_1, s_1)(v_2, s_2)\cdots$ is a play in $\mathbb{G}'$ where Éloïse respects her winning strategy φ' and hence is winning for her. Therefore, it follows that $s_0 s_1 s_2 \cdots$ is an accepting run of $\mathcal{S}$ on $a_0 a_1 a_2 \cdots$ and hence that $a_0 a_1 a_2 \cdots \in \Omega$, equivalently that λ is winning for Éloïse.

Conversely, using the same construction for Abelard, one built from a winning strategy for him in $\mathbb{G}'$ a winning strategy in $\mathbb{G}$.

One should note that the increase of memory from a winning strategy in $\mathbb{G}'$ to a winning strategy in $\mathbb{G}$ can be handled by a finite automaton, namely $\mathcal{S}$. □

Hence we have the following corollary of Theorem 13.

Corollary 25 *Given any n-CPDA game equipped with an ω-regular winning condition it is decidable whether Éloïse has a winning strategy from the initial configuration. Moreover one can construct an n-CPDS with output that realizes a winning strategy for the player that wins from the initial configuration.*

Proof Applying Proposition 24, one gets an *equivalent* parity game. It is then easy to check that if one starts with a collapsible pushdown graph, the resulting graph is still a collapsible one (of same order) as the transformation only operates on the control states. Then decidability follows from the one for parity collapsible pushdown games. □

ε-closure of CPD graphs

Theorem 13 can easily be generalized to the case where the game is played on the ε-closure of the configuration graph of an n-CPDS graph.

Hence One considers an order-n collapsible pushdown automaton with input and ε-transition $\mathcal{A}$, that is an order-n collapsible pushdown automaton with input alphabet A containing a special symbol denoted ε. As explain previously, it defines an A-labeled transition graph $G = (V, E \subseteq V \times A \times V)$. The ε-closure of G, denoted $G^{\varepsilon *}$, is the graph $(V^{\varepsilon *}, E^{\varepsilon *} \subseteq V^{\varepsilon *} \times (A \setminus \{\varepsilon\}) \times V^{\varepsilon *})$ where

- $V^{\varepsilon *} \subseteq V$ is the subset of vertices reachable from the initial configuration by a finite sequence of transitions such that the last one is not labeled by ε.
- $(v, a, v') \in E^{\varepsilon *}$ if and only if $v, v' \in V^{\varepsilon *}$ and there is a path in G from v to v' labeled by a (possibly empty) sequence of ε transition followed by a transition labeled by a.

[21] One could consider for φ a memoryless strategy, but we prefer to consider the general case here has we may later argue that one can build effective winning strategies for ω-regular collapsible pushdown games from winning strategies for parity collapsible pushdown games that we may not choose memoryless.

We consider a partition of the control states of $\mathcal{A}$ into E-states and A-states, and assign a priority to each control state. The game structure extends naturally to a partition of and a priority function on the vertices of $G^{\varepsilon *}$. We aim at deciding the winner in the resulting game from the initial configuration. For this we prove a more general reduction result.

Proposition 26 *Let $\mathbb{G}^{\varepsilon *}$ be a parity game on the ε-closure of some $(A \cup \{\varepsilon\})$-labeled graph $G = (V, E)$. Then one can define a parity game $\mathbb{G}'$ over an unlabeled graph $G' = (V' = V \cup V \times \{E, A\}, E')$ such that the following holds: for any vertex $v \in V$, Éloïse wins from v in $\mathbb{G}^{\varepsilon *}$ iff she wins from v in $\mathbb{G}'$.*

Moreover, from a winning strategy from v in $\mathbb{G}'$, one can effectively build a corresponding winning strategy from v in $\mathbb{G}^{\varepsilon *}$.

Proof By definition, a transition in $G^{\varepsilon *}$ can be naturally decomposed as a sequence of ε transitions in G ended by a non-ε transition in G. The graph G' aims at making this decomposition explicit: in G' when some player (let us say Éloïse) wants to mimic an a-transition from v to v' (for some a, v, v'), then she moves from v to (v_1, E) then to (v_2, E) and so on until reaching a vertex (v_k, E) from which she finally moves to v'. Such a sequence of moves is possible provided $(v, \varepsilon, v_1), (v_1, \varepsilon, v_2), \ldots, (v_{k-1}, \varepsilon, v_k)(v_k, a, v')$ is a valid sequence of moves in G (note here that we may have $k = 0$ in which case this sequence is replaced by (v, a, v') and the simulation is an exact one). Note that to make this work, one has to assign every vertex in $V \times E$ to Éloïse. A symmetrical simulation is defined for Abelard. □

Hence we have the following corollary of Theorem 13.

Corollary 27 *Given any parity game over the ε-closure of a collapsible pushdown graph, it is decidable whether Éloïse has a winning strategy from the initial configuration. Moreover one can construct an n-CPDS with output that realizes a winning strategy for the player that wins from the initial configuration.*

Proof Applying Proposition 26, one gets an *equivalent* parity game. It is then easy to check that if one starts with the ε-closure of a collapsible pushdown graph, the resulting graph is still a collapsible one (of same order) as the transformation only operates on the control states. Then decidability follows from the one for parity collapsible pushdown games. □

Using the same techniques, one can get a result that generalizes both Corollary 25 and Corollary 27.

Corollary 28 *Given game over the ε-closure of a collapsible pushdown graph equipped with an ω-regular winning condition, it is decidable whether Éloïse has a winning strategy from the initial configuration. Moreover one can construct an n-CPDS with output that realizes a winning strategy for the player that wins from the initial configuration.*

CPDS graphs *vs* Caucal graphs

The class of ε-closure of configuration graphs of CPDS admits decidable Mu-calculus theories, as parity games are decidable from Corollary 27. Moreover this class contains the class of Caucal graphs [5] as these graphs are exactly those obtained by taking the ε-closure of the transition graphs of (non-collapsible) higher-order pushdown graphs [4].

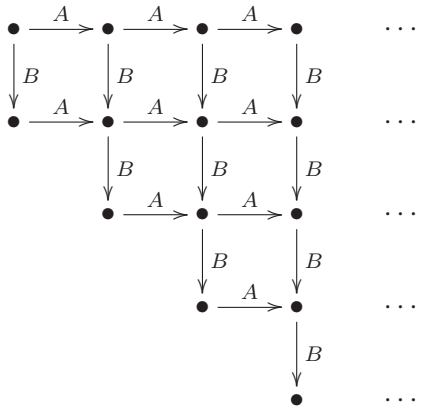
Now recall that Caucal graphs enjoy decidable MSO theories [5], and therefore one can consider the similar problem for (ε-closure) configuration graphs of CPDS. The next result proves that the MSO theories of those graphs are not decidable in general, and this implies that the inclusion of Caucal graphs inside the class of ε-closure of configuration is a strict one.

Theorem 29 (Undecidability) *MSO theories of configuration graphs of CPDS are not in general decidable. Hence the class of ε-closure of configuration of CPDS graphs strictly contains the Caucal graphs.*

Proof Consider the following MSO interpretation I of the configuration graph of the 2-CPDS in Example 6.1.

$$\begin{aligned} I_A(x,y) &= x \xrightarrow{C} y \ \wedge \ x \xrightarrow{R} y \\ I_B(x,y) &= x \xrightarrow{1} y \end{aligned}$$

with $C = \overline{1}^* \, \overline{b} \, a \, t \, b \, 1^*$ and $R = 0 \, t \, a \, \overline{0} \ \vee \ \overline{1} \, 0 \, t \, a \, \overline{0} \, 1$. We observe that the (image of the) interpretation is the following "infinite half-grid"



Note that for the A-edges, the constraint C requires that the target vertex should be in the next column to the right, while R specifies the correct row. Since the interpretation I preserves MSO decidability, and its image has an undecidable MSO theory (because the Halting Problem of Turing machines can be reduced to it), the MSO theory of the above configuration graph must be undecidable. □

7 Conclusions and further directions

In this paper, we introduce *collapsible pushdown automata* and prove that they are equi-expressive with (general) recursion schemes for generating trees. This is the first automata-theoretic characterization of higher-order recursions schemes. We argue that the equi-expressivity result is significant because it acts as a bridge, enabling inter-translation between model-checking problems about trees generated by recursion scheme and solvability of games on collapsible pushdown graphs. We show (Theorem 29) that order-n CPDS are strictly more expressive then order-n pushdown systems for generating graphs.

As for **further directions**:

1. The most pressing open problem is whether order-n CPDA are equi-expressive with order-n PDA *for generating trees*. The conjecture is that the former are strictly more expressive. Specifically it is conjectured that *Urzyczyn tree* [2] is definable by a 2-CPDA but not by any 2-PDA.

2. Is there a finite way to represent the set of winning positions of an n-CPDS parity game (equivalently to represent the set of vertices where a given modal mu-calculus formula holds)?

3. Is there an *à la* Caucal definition for the ε-closure of CPDS graphs? As trees generated by n-CPDA are exactly those obtained by unravelling an n-CPDS graph, is there a class of transformations $\mathcal{T}$ from trees to graphs such that every $(n+1)$-CPDS graph is obtained by applying a $\mathcal{T}$-transformation to some tree generated by an n-CPDA. Note that a $\mathcal{T}$-transformation may in general not preserve MSO decidability, but should preserve modal mu-calculus decidability of trees generated by n-CPDA.

4. The algorithm that transforms recursion schemes to CPDA (briefly sketched in Section 5) uses ideas in game semantics. It would be an interesting (and we believe challenging) problem to obtain a translation that uses only first principles.

References

[1] K. Aehlig, J. G. de Miranda, and C.-H. L. Ong. Safety is not a restriction at level 2 for string languages. Technical Report PRG-RR-04-023, Oxford University Computing Laboratory, 2004.

[2] K. Aehlig, J. G. de Miranda, and C.-H. L. Ong. Safety is not a restriction at level 2 for string languages. In *Proceedings of the 8th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'05)*, volume 3411 of *Lecture Notes in Computer Sciences*, pages 490–501. Springer-Verlag, 2005.

[3] T. Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *Proceedings of Automata, Languages and Programming, 30th International Colloquium (ICALP'03)*, volume 2719 of *Lecture Notes in Computer Sciences*, pages 556–569. Springer-Verlag, 2003.

[4] A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *Lecture Notes in Computer Sciences*, pages 112–123. Springer-Verlag, 2003.

[5] D. Caucal. On infinite terms having a decidable monadic theory. In *Proceedings of Mathematical Foundations of Computer Science 2002, 27th International Symposium (MFCS'02)*, volume 2420 of *Lecture Notes in Computer Sciences*, pages 165–176. Springer-Verlag, 2002.

[6] D. Caucal and S. Hassen. Higher-order recursive schemes. Private communication, 28 pages, July 2006.

[7] B. Courcelle. The monadic second-order logic of graphs IX: machines and their behaviours. *Theoretical Computer Science*, 151:125–162, 1995.

[8] W. Damm. The IO- and OI-hierarchy. *Theoretical Computer Science*, 20:95–207, 1982.

[9] W. Damm and A. Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Information and Control*, 71:1–32, 1986.

[10] J. de Miranda. *Structures generated by higher-order grammars and the safety constraint*. PhD thesis, University of Oxford, 2006. Forthcoming.

[11] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS'91)*, pages 368–377. IEEE Computer Society, 1991.

[12] J. Engelfriet. Interated stack automata and complexity classes. *Information and Computation*, pages 21–75, 1991.

[13] J. M. E. Hyland and C.-H. L. Ong. On Full Abstraction for PCF: I. Models, observables and the full abstraction problem, II. Dialogue games and innocent strategies, III. A fully abstract and universal game model. *Information and Computation*, 163:285–408, 2000.

[14] T. Knapik, D. Niwiński, and P. Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *Proceedings of Typed Lambda Calculi and Applications, 5th International Conference (TLCA'01)*, volume 2044 of *Lecture Notes in Computer Sciences*, pages 253–267. Springer-Verlag, 2001.

[15] T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'02)*, volume 2303 of *Lecture Notes in Computer Sciences*, pages 205–222. Springer-Verlag, 2002.

[16] T. Knapik, D. Niwiński, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *Proceedings of Automata, Languages and Programming, 32nd International Colloquium (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Sciences*, pages 1450–1461. Springer-Verlag, 2005.

[17] A. N. Maslov. The hierarchy of indexed languages of an arbitrary level. *Soviet mathematics Doklady*, 15:1170–1174, 1974.

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

[19] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1989.

[20] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 81–90. IEEE Computer Society, 2006.

[21] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. Preprint, 2006.

[22] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[23] C. Stirling. Higher-order matching and games. In *Proceedings of Computer Science Logic, 19th International Workshop (CSL'05)*, volume 3634 of *Lecture Notes in Computer Sciences*, pages 119–134. Springer-Verlag, 2005.

[24] C. Stirling. Decidability of higher-order matching. In *Proceedings of Automata, Languages and Programming, 33rd International Colloquium (ICALP'06)*, volume 4052 of *Lecture Notes in Computer Sciences*, pages 348–359. Springer-Verlag, 2006.

[25] W. Thomas. On the synthesis of strategies in infinite games. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *Lecture Notes in Computer Sciences*, pages 1–13. Springer-Verlag, 1995.

[26] I. Walukiewicz. Pushdown processes: games and model-checking. *Information and Computation*, 157:234–263, 2001.

[27] I. Walukiewicz. A landscape with games in the backgroung. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 356–366. Computer Society Press, 2004.

[28] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.