Collapsible Pushdown Automata and Recursion Schemes*

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

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 topmost symbol of s. Our first result is that CPDA are equi-expressive with *recursion schemes* as generators of (possibly infinite) 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 an abstract syntax graph of the scheme, and hence paths in the tree generated by the scheme. Our equi-expressivity result is the first automata-theoretic characterization of higher-order recursion schemes. Thus CPDA are also a characterization of the *simply-typed lambda calculus with recursion* (generated from uninterpreted 1st-order symbols) and of (pure) *innocent strategies*.

An important consequence of the equi-expressivity result is that it allows us to reduce decision problems on trees generated by recursion schemes to equivalent problems on CPDA and *vice versa*. Thus we show, as a consequence of a recent result by Ong (modal mu-calculus model-checking of trees generated by recursion schemes is n-EXPTIME complete), that the problem of solving parity games over the configuration graphs of order-n CPDA is n-EXPTIME complete, subsuming several well-known results about the solvability of games over higher-order pushdown graphs by (respectively) Walukiewicz, Cachat, and Knapik *et al*. Another contribution of our work is a self-contained proof of the same solvability result by generalizing *standard* techniques in the field. By appealing to our equi-expressivity result, we obtain a new proof of Ong's result.

In contrast to higher-order pushdown graphs, we show that the monadic second-order theories of the configuration graphs of CPDA are *undecidable*. It follows that – as generators of graphs – CPDA are strictly more expressive than higher-order pushdown automata.

1 Introduction

Higher-order pushdown automata (PDA) were first introduced by Maslov [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 Maslov's hierarchy was given by Damm and Goerdt [9, 10]: 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. Maslov'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 [2], though little is known about languages at higher orders (see e.g. [12]).

Higher-order PDA as a generating device for (possibly infinite) labelled ranked trees was first studied by Knapik, Niwiński and Urzyczyn [16]. As in the case of word languages, an infinite hierarchy of trees can be 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 [20], algebraic [8] and hyperalgebraic trees [15]. 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 of that work is the equi-expressivity of the two methods as tree generators. A question of fundamental importance in higher-type recursion is to find a class of automata that characterizes the expressivity of higher-order

*We direct readers to the (downloadable) long version [13] of this paper in which all proofs are presented.

†Matthew.Hague@comlab.ox.ac.uk Oxford University Computing Laboratory (OUCL)

‡Andrzej.Murawski@comlab.ox.ac.uk OUCL

§Luke.Ong@comlab.ox.ac.uk OUCL

¶Olivier.Serre@liafa.jussieu.fr LIAFA (CNRS and Université Paris Diderot – Paris 7)

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, which seem somewhat 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 [17], 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* [1], 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 (Theorems 4.3 and 5.1) of this paper is that for every $n \geq 0$, order-n recursion schemes and order-n CPDA are equi-expressive as generators of ranked trees.

Our equi-expressivity result has a number of important consequences. It allows us to reduce 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 over ranked trees (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 [19] 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 Problem, which subsumes many known results [22, 5, 17]. Moreover our approach yields techniques that are significantly different from standard methods for solving model-checking problems on infinite graphs generated by finite machines.

This transfer of techniques goes both ways. Indeed another contribution 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 [19].

In contrast to higher-order pushdown graphs (which do have decidable MSO theories [6]), we show that the MSO theories of collapsible pushdown graphs are undecidable. Hence collapsible pushdown graphs are, to our knowledge, the first example of a general[2] and natural class of finitely-presentable graphs that have undecidable MSO theories while enjoying decidable modal mu-calculus theories.

2 Collapsible pushdown automata (CPDA)

Fix a stack alphabet Γ and a distinguished *bottom-of-stack symbol* $\bot \in \Gamma$. An ***order-0 stack*** is just a stack symbol. An ***order-***$(n+1)$ ***stack*** s is a non-null sequence (written $[\, s_1 \cdots s_l \,]$) of order-n stacks such that every non-$\bot$ Γ-symbol a that occurs in s has a link to a stack (of order k where $k \leq n$) situated below it in s; we call the link a $(k+1)$-*link*. The order of a stack s is written $ord(s)$; and we shall abbreviate order-n stack to n-stack. As usual, the bottom-of-stack[3] symbol $\bot$ cannot be popped from or pushed onto a stack. We define $\bot_k$, the ***empty*** k***-stack***, as: $\bot_0 = \bot$ and $\bot_{k+1} = [\, \bot_k \,]$. When displaying n-stacks in examples, we shall omit the bottom-of-stack symbols and 1-links (i.e. links to stack symbols) to avoid clutter (writing e.g. $[\,[\,]\,[\, a\, b \,]\,]$ instead of $[\,[\,\bot\,]\,[\,\bot \overset{\frown}{a}\, b \,]\,]$.

The set Op_n of order-n *stack operations* consists of the following four types of operations:

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

First we introduce the auxiliary operations: top_i, which takes a stack s and returns the top $(i-1)$-stack of s; and $push_1^a$, which takes a stack s and pushes the symbol a onto the top of the top 1-stack of s. Precisely let $s = [\, s_1 \cdots s_{l+1} \,]$ be a stack with $1 \leq i \leq ord(s)$, we define

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

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

[2] *Pace* nested trees [3], which are a highly constrained class of acyclic graphs with "jump edges", as reflected in the specialized vocabulary of their logical representation.

[3] Thus we require an *order-1 stack* to be a non-null sequence $[\, a_1 \cdots a_l \,]$ of Γ-symbols such that for all $1 \leq i \leq l$, $a_i = \bot$ iff $i = 1$.

and define $push_1^a \underbrace{[\, s_1 \cdots s_{l+1} \,]}_{s}$ by

$$\begin{cases} [\, s_1 \cdots s_l \; push_1^a \; s_{l+1} \,] & \text{if } ord(s) > 1 \\ [\, s_1 \cdots s_{l+1} \; a \,] & \text{if } ord(s) = 1 \end{cases} \quad (1)$$

We can now explain the four operations in turn. For $i \geq 1$ the *order-i pop* operation, pop_i, takes a stack and returns it with its top $(i-1)$-stack removed. Let $1 \leq i \leq ord(s)$ we define $pop_i \underbrace{[\, s_1 \cdots s_{l+1} \,]}_{s}$ by

$$\begin{cases} [\, s_1 \cdots s_l \,] & \text{if } i = ord(s) \text{ and } l \geq 1 \\ [\, s_1 \cdots s_l \; pop_i s_{l+1} \,] & \text{if } i < ord(s) \end{cases} \quad (2)$$

We say that a stack s_0 is a ***prefix*** of a stack s (of the same order), written $s_0 \leq s$, just if s_0 can be obtained from s by a sequence of (possibly higher-order) pop operations.

Take an n-stack s and let $i \geq 2$. To construct $push_1^{a,i}\, s$ we first attach a link from a fresh copy of a to the $(i-1)$-stack that is immediately below the top $(i-1)$-stack of s, and then push the symbol-with-link onto the top 1-stack of s. As for $collapse$, suppose the top_1-symbol of s has a link to (a particular copy of) the k-stack u somewhere in s. Then $collapse\ s$ causes s to "collapse" to the prefix s_0 of s such that $top_{k+1}\, s_0$ is that copy of u. Finally, for $j \geq 2$, the *order-i push* operation, $push_j$, simply takes a stack s and duplicates the top $(j-1)$-stack of s, preserving its link structure.

Example 2.1 Take the 3-stack $s = [[[\, a]]\ [[]\, [\, a]]]$. We have

$$\begin{aligned} push_1^{b,2}\, s &= [[[\, a]]\ [[]\, [\, a\, b]]] \\ collapse\,(push_1^{b,2}\, s) &= [[[a]]\ [[]]] \\ \underbrace{push_1^{c,3}(push_1^{b,2}\, s)}_{\theta} &= [[[\, a]]\ [[]\, [\, a\, b\, c]]]. \end{aligned}$$

Then $push_2\, \theta$ and $push_3 \theta$ are respectively

$$[[[\, a]]\ [[]\, [\, a\, b\, c]\, [\, a\, b\, c]]] \text{ and}$$

$$[[[\, a]]\ [[]\, [\, a\, b\, c]]\ [[]\, [\, a\, b\, c]]].$$

We have $collapse\,(push_2\, \theta) = collapse\,(push_3\, \theta) = collapse\, \theta = [[[\, a]]]$.

One way to define these stack operations formally is to work with an appropriate numeric representation of the links. Knapik *et al.* [17] have shown how this can be done in the order-2 case. Here we introduce a different encoding of links that works for all orders. The idea is simple: take an n-stack s and suppose there is a link from (a particular occurrence of) a symbol a in s to some $(j-1)$-stack. Let s_0 be the unique prefix of s whose top_1-symbol is that occurrence of a. Then there is a unique k such that $collapse\ s_0 = pop_j^k\ s_0$ where pop_j^k means $\underbrace{pop_j\, ;\cdots;\, pop_j}_{k}$. We shall represent the occurrence of a with its link as $a^{(j,k)}$ in s. Formally, a *symbol-with-link* of an n-stack is written $a^{(j,k)}$, where $a \in \Gamma$, $1 \leq j \leq n$ and $k \geq 1$, such that[4] if $j = 1$ then $k = 1$. Even though there is no link from $\perp$, for technical convenience, we assume if $a = \perp$ then $j = k = 1$.

Example 2.2 To illustrate our numeric encoding of links, we revisit Example 2.1. Take the 3-stack $s = [[[\, a]]\ [[]\, [\, a]]]$ defined therein. Omitting the superscript $(1,1)$ to save writing, we have

$$\begin{aligned} push_1^{b,2}\, s &= [[[\, a]]\ [[]\, [\, a\, b^{(2,1)}]]] \\ \underbrace{push_1^{c,3}(push_1^{b,2}\, s)}_{\theta} &= [[[\, a]]\ [[]\, [\, a\, b^{(2,1)}\, c^{(3,1)}]]]. \end{aligned}$$

Then $push_2\, \theta$ and $push_3 \theta$ are respectively

$$[[[\, a]]\ [[]\, [\, a\, b^{(2,1)}\, c^{(3,1)}]\, [\, a\, b^{(2,2)}\, c^{(3,1)}]]] \text{ and}$$
$$[[[\, a]]\ [[]\, [\, a\, b^{(2,1)}\, c^{(3,1)}]]\ [[]\, [\, a\, b^{(2,1)}\, c^{(3,2)}]]].$$

Henceforth we shall adopt our numeric representation of symbols-with-links. We can now give the formal definitions of $collapse$, $push_1^{b,i}$ and $push_j$ (in terms of pop_j and $push_1^b$ as defined in (2) and (1) respectively). Let $1 \leq i \leq ord(s)$ and $2 \leq j \leq ord(s)$ we define

$$\begin{aligned} collapse\, s &= pop_e^f\, s \quad \text{where } top_1\, s = a^{(e,f)} \\ push_1^{b,i}\, s &= push_1^{b^{(i,1)}}\, s \end{aligned}$$

and define $push_j \underbrace{[\, s_1 \cdots s_{l+1} \,]}_{s}$ by

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

where $\Theta^{\langle j \rangle}$ is the operation of replacing every superscript (j, k_j) (for some k_j) occurring in the stack Θ by $(j, k_j + 1)$; note that in case $j = ord(s)$, the link structure of s_{l+1} is preserved by the copy (as represented by $s_{l+1}^{\langle j \rangle}$) that is pushed on top of s by $push_j$.

Definition 2.3 Fix an alphabet Σ. A ***word-language generating n-CPDA*** is a 5-tuple $\mathcal{A} = \langle\, Q, \Sigma, \Gamma, q_0 \in Q, \Delta \subseteq Q \times \Sigma \times Q \times Op_n \,\rangle$ where Γ is a stack alphabet, Q is a finite state-set, and q_0 is the initial state. *Configurations* of an n-CPDA are pairs of the form (q, s) where $q \in Q$ and

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

s is an n-stack over Γ; we call $(q_0, \bot_n)$ the *initial configuration*. The transition relation Δ induces a labelled transition relation over configurations: $(q,s) \overset{a}{>} (q', \theta(s))$ if $(q, a, q', \theta) \in \Delta$. We say that $a_1 \cdots a_l \in \Sigma^*$ is ***accepted*** by $\mathcal{A}$ just if we have $(q_0, \bot_n) \overset{a_1}{>} (q_1, s_1) \overset{a_2}{>} \cdots \overset{a_l}{>} (q', \bot_n)$ for some $q' \in Q$. Standardly the language ***recognized*** by $\mathcal{A}$ is the set of words it accepts. (An n-***PDA*** is just an n-CPDA in which $collapse$ is *not* a stack operation.)

Example 2.4 [1] We define the language U (for Urzyczyn) over the alphabet $\{\,\mathbf{(},\mathbf{)},*\,\}$ as follows. A U-***word*** is composed of 3 segments:

$$\underbrace{\mathbf{(}\cdots\mathbf{(}\cdots\mathbf{(}}_{A}\ \underbrace{\mathbf{(}\cdots\mathbf{)}\cdots\mathbf{(}\cdots\mathbf{)}}_{B}\ \underbrace{*\cdots*}_{C}$$

- Segment A is a prefix of a well-bracketed word that ends in **(**, and the opening **(** is not matched in the (whole) word.
- Segment B is a well-bracketed word.
- Segment C has length equal to the number of **(** in A, whether matched or unmatched.

Note that each U-word has a unique decomposition. E.g. $\mathbf{(()(\underline{)}(())}***$ and $\mathbf{((}^n\mathbf{)}^n\mathbf{(}\ *^n\ **$ are in U (their respective B-segments are underlined and empty). The language U is not context free (by applying the "$uvwxy$" Lemma to the preceding example) but recognizable by a *non-deterministic* 2-PDA. Surprisingly, U is recognizable by a *deterministic* 2-CPDA defined (informally) as follows:
- on reading **(** do $push_2$ **;** $push_1^{a,2}$
- on reading **)** do pop_1
- on reading the first $*$ do $collapse$, and on reading any subsequent $*$ do pop_2.

This illustrates the power of collapse. We conjecture that U is *not* recognizable by any *deterministic* 2-PDA (because of the need to guess the transition from segment A to B).

Definition 2.5 A ***tree-generating*** n***-CPDA*** is a 5-tuple $\langle\, \Sigma, \Gamma, Q, \delta, q_0 \,\rangle$ where Σ is a ranked alphabet (i.e. each Σ-symbol f has an *arity* $ar(f) \geq 0$) and $\delta : Q \times \Gamma \longrightarrow (Q \times Op_n + \{\,(f; q_1, \cdots, q_{ar(f)}) : f \in \Sigma, q_i \in Q\,\})$ is the transition function. 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 $\overset{\ell}{>}$, a labelled transition relation over generalized configuration by clauses, one for each of the three types[5] of ***labels*** ℓ that annotate $\overset{\ell}{>}$, namely, 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 $\delta(q, top_1\, s) = (f; q_1, \cdots, q_{ar(f)})$, writing $\overline{q} = q_1, \cdots, q_{ar(f)}$

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

A *computation path* of an n-CPDA $\mathcal{A}$ is a finite or infinite transition sequence of the form $\rho = \gamma_0 \overset{\ell_0}{>} \gamma_1 \overset{\ell_1}{>} \gamma_2 \overset{\ell_2}{>} \cdots$ 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 marks the end of the Σ-projection). We say the CPDA $\mathcal{A}$ ***generates*** the Σ-labelled tree t just in case the *branch language*[6] of t coincides with the Σ-projection of computation paths of $\mathcal{A}$.

Remark 2.6 *Are n-CPDA strictly more expressive than n-PDA?* (It follows from the definition that they are at least as expressive as n-PDA.) When viewed as generators of word languages, the answer is no[7] for $n = 2$ but conjectured to be yes for $n > 2$. When viewed as tree generators, the conjecture is yes for all n (this is equivalent to the *Safety Conjecture* [16] in view of Sections 4 and 5). When viewed as generators of directed graphs, the answer is yes for all n – see Section 6.

3 Recursion schemes

Types are generated from the base type o using the arrow constructor $\rightarrow$. Every type A can be written uniquely as $A_1 \rightarrow \cdots \rightarrow A_n \rightarrow o$ (arrows associate to the right), for some $n \geq 0$ which is called its *arity*; we shall often write A simply as $(A_1, \cdots, A_n, o)$. We define the *order* of a type by $ord(o) = 0$ and $ord(A \rightarrow B) = \max(ord(A) + 1, ord(B))$. Let Σ be a *ranked alphabet* i.e. each Σ-symbol f has an arity $ar(f) \geq 0$ which determines its type $(\underbrace{o, \cdots, o}_{ar(f)}, o)$. Further we shall assume that each symbol $f \in \Sigma$ is assigned a finite set $\mathsf{Dir}(f)$ of $ar(f)$ *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

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

[6] 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 $t(d_1 \cdots d_n) = a$.

[7] As language generators, 2-CPDA are equi-expressive with *non-deterministic* 2-PDA (see [1]).

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 $\{\, \alpha\, i : i \in \mathsf{Dir}(t(\alpha)) \,\}$ (i.e. t is a *ranked* tree). We shall assume that the ranked alphabet Σ contains a distinguished nullary symbol $\perp$ which will be used exclusively to label "undefined" nodes.

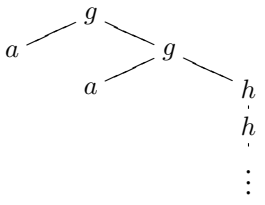
Note. We write $[m]$ as a shorthand for $\{\, 1, \cdots, m \,\}$. *Henceforth we fix a ranked alphabet* Σ *for the rest of the paper*, and set $\mathsf{Dir}(f) = [ar(f)]$ for each $f \in \Sigma$; thus $\mathsf{Dir}(\Sigma) = [ar(\Sigma)]$, writing $ar(\Sigma)$ to mean $\max\{\, ar(f) : f \in \Sigma \,\}$.

For each type A, we assume an infinite collection Var^A of variables of type A, and write Var to be the union of Var^A as A ranges over types; we write $t : A$ to mean that the expression t has type A. A (deterministic) ***recursion scheme*** is a tuple $G = \langle\, \Sigma, \mathcal{N}, \mathcal{R}, S \,\rangle$ where Σ is a ranked alphabet of *terminals*; $\mathcal{N}$ is a set of typed *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 in Var^{A_i}, and $e \in \mathcal{T}^o(\Sigma \cup \mathcal{N} \cup \{\, \xi_1, \cdots, \xi_n \,\})$ i.e. e is an *applicative term* of type o generated from elements of $\Sigma \cup \mathcal{N} \cup \{\, \xi_1, \cdots, \xi_n \,\}$. The *order* of a recursion scheme is the highest order of the types of its non-terminals.

We use recursion schemes as generators of Σ-labelled trees. The ***value tree*** of (or the tree ***generated*** by) a recursion scheme G, denoted $[\![\, G \,]\!]$, is a possibly infinite applicative term, but viewed as a Σ-labelled tree, *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. See e.g. [16] for a formal definition.

Example 3.1 Let G be the order-2 *unsafe* (in the sense of [16]) recursion scheme with rewrite rules where $z : o$ and $\varphi : (o, o)$:

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



where the arities of the terminals g, h, a are $2, 1, 0$ respectively. The value tree $[\![\, G \,]\!]$ (as shown on the right) is the Σ-labelled tree defined by the infinite term $g\, a\, (g\, a\, (h\, (h\, (h\, \cdots))))$. The only infinite *path* in the tree is the node-sequence $\varepsilon \cdot 2 \cdot 22 \cdot 221 \cdot 2211 \cdots$.

4 From CPDA to recursion schemes

In this section we show that there is an effective translation from order-n CPDA $\mathcal{A}$ to order-n recursion schemes $G_{\mathcal{A}}$ (where $n \geq 0$) such that $\mathcal{A}$ and $G_{\mathcal{A}}$ define the same Σ-labelled tree (Theorem 4.3). We begin by introducing a method to represent order-n stacks and configurations by applicative terms constructed from non-terminals of order n. Our approach simplifies somewhat the (order-2) translation in [17] and generalizes it to all finite orders.

Fix a tree-generating n-CPDA $\mathcal{A}$. W.l.o.g. we assume that the state-set of $\mathcal{A}$ is $[m]$ where $m \geq 1$. Let 0 be the base type. Inductively, for $n \geq 0$, we define the type $n+1 = n^m \to n$ where $n^m = \underbrace{n \times \cdots \times n}_{m \text{ times}}$. Thus $n+1 = n^m \to (n-1)^m \to \cdots \to 0^m \to 0$. For each stack symbol a, each $1 \leq e \leq n$ and each state $1 \leq p \leq m$, we introduce a non-terminal

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

that represents the symbol a with a link of order e (in state p). Note that the type of $\mathcal{F}_p^{a,e}$ is not homogeneous in the sense of Knapik *et al.* [16]. In addition, for each $0 \leq i \leq n-1$, we introduce a non-terminal $\Omega_i : i$, and fix a start symbol $S : 0$. Let $\mathcal{N}_{\mathcal{A}}$ be the set of all non-terminals. We shall use the following shorthand: Let $P(i)$ be a term with an occurrence of i; we write $\langle P(i) \,|\, i \rangle$ as a shorthand for 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). We shall 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 configuration (p, s) is represented by $\mathcal{F}_p^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_0} \;:\; 0$. The idea is that for $1 \leq k \leq n$, we have $(p, top_k\, s)$ is represented by

$$\mathcal{F}_p^{a,e}\, \overline{L}\, \overline{M_{n-1}} \cdots \overline{M_{n-(k-1)}} : n - (k-1),$$

$(p, pop_k\, s)$ is represented by $M_{n-k,p}\, \overline{M_{n-k-1}} \cdots \overline{M_0} : 0$, and $(p, collapse\ s)$ is represented by $L_p\, \overline{M_{n-e-1}} \cdots \overline{M_0} : 0$. In particular the 0-configuration $(p, top_1\, s)$ – where the top_1-symbol of s is a with a link to the $(e-1)$-stack that is represented by the m-tuple $\overline{L} : (n-e)^m$ – is represented by $\mathcal{F}_p^{a,e}\, \overline{L} : n$.

What does it mean for a term to represent a configuration? To give a precise answer, we first consider labelled

rewrite rules of the general form, with q ranging over states and θ over Op_n:

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

where 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, where $2 \leq j \leq n$ and $1 \leq e,k \leq n$: The labelled rewrite rules induce a family of labelled *outermost* transition relations $\overset{(q,\theta)}{\longrightarrow} \subseteq\ \mathcal{T}^0(\mathcal{N}_\mathcal{A}) \times \mathcal{T}^0(\mathcal{N}_\mathcal{A})$. Informally we define $M \overset{(q,\theta)}{\longrightarrow} M'$ 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; since each binary relation $\overset{(q,\theta)}{\longrightarrow}$ is a partial function, we shall write $M\overset{(q,\theta)}{\longrightarrow}$ to mean M'. We shall write $\overset{\theta}{\longrightarrow}$ to mean the set of all transitions $M \overset{(q,\theta)}{\longrightarrow} M'$ that preserves the state q of M. Let $\alpha = \theta_1\,;\cdots;\,\theta_l$ be a (composite) sequence of stack operations. We write $\overset{\alpha}{\longrightarrow}\ \subseteq\ \mathcal{T}^0(\mathcal{N}_\mathcal{A}) \times \mathcal{T}^0(\mathcal{N}_\mathcal{A})$ to be the sequential composition of the partial function $\overset{\theta_1}{\longrightarrow},\cdots,\overset{\theta_l}{\longrightarrow}$ (in this order).

The position of a given stack symbol in an n-stack s can be described by a sequence of (possibly higher-order) pop operations that can "collapse" the stack up to the point where that position becomes the top_1-symbol. For example, the position of b in the 2-stack $[\,[\,a\,a\,]\,[\,a\,b\,a\,]\,[\,a\,a\,]\,[\,a\,]\,]$ 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. We shall call a normalized sequence for a given stack s an *s-probe*. We say that a ground-type term M ***represents*** a configuration (p,s) if for every s-probe α, if the top_1-symbol of $\alpha\,s$ is $a^{(j,k)}$, then the head non-terminal of $M\overset{\alpha}{\longrightarrow}$ is $\mathcal{F}_p^{a,j}$; further $(M\overset{\alpha}{\longrightarrow})\overset{pop_j^k}{\longrightarrow} = (M\overset{\alpha}{\longrightarrow})\overset{coll.}{\longrightarrow}$, and it represents the configuration $(p, collapse(\alpha\,s))$. Note that $\mathcal{F}_p^{\perp,1}\,\overline{\Omega_{n-1}}\,\overline{\Omega_{n-1}}\cdots\overline{\Omega_{n-j}} : n-j$ represents the j-configuration $(p,\perp_{n-j})$. The following Theorem confirms that our notion of representation is the right one.

Theorem 4.1 (Correctness) *Let M be a ground-type term, (p,s) be a configuration, and θ be a stack operation. Suppose M represents (p,s). If $M \overset{\theta}{\longrightarrow} M'$ then M' represents the configuration $(p,\theta\,s)$.*

Definition 4.1 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 $G_\mathcal{A}$, consists of a *start rule*:

$$S \longrightarrow \mathcal{F}_1^{\perp,1}\overline{\Omega_{n-1}}\,\overline{\Omega_{n-1}}\cdots\overline{\Omega_0}$$

and two types of rewrite rules (according to the type of their label), namely, I and P:

I. For each $(q,\theta) \in \delta(p,a)$ and $1 \leq e \leq n$, there is an I-type rewrite rule

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

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

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

$$\mathcal{F}_p^{a,e}\,\overline{\Xi}\quad\overset{(f;\overline{q})}{\longrightarrow}\quad f\,(\mathcal{F}_{q_1}^{a,e}\,\overline{\Xi})\cdots(\mathcal{F}_{q_{ar(f)}}^{a,e}\,\overline{\Xi}).$$

where $\overline{\Xi}$ is a shorthand for $\overline{\Phi}\,\overline{\Psi_{n-1}}\cdots\overline{\Psi_0}$. We write $\longrightarrow \subseteq \mathcal{T}^0(\Sigma\cup\mathcal{N}_\mathcal{A})\times\mathcal{T}^0(\Sigma\cup\mathcal{N}_\mathcal{A})$ for the one-step reduction relation[8] between ground-type applicative terms, defined to be the substitutive and contextual closure of the rewrite rules.

A ground-type term R is called a *redex* if for some term R' we have $R \longrightarrow R'$ is a *substitutive* instance of a rewrite rule $\overset{\ell}{\longrightarrow\!\triangleright}$, 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 $R \overset{\ell}{\longrightarrow\!\triangleright} R'$. A ground-type term is either *head terminal* (i.e. of the shape $f\,N_1\cdots N_{ar(f)}$) or *head non-terminal* (i.e. the head symbol is a non-terminal). 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, it is an I-type or P-type redex. In order to prove the Theorem (Equi-Expressivity 1), we define by rule induction a binary relation $\overset{\ell}{\Rightarrow}$ over pairs of the form (E,R) where ℓ ranges over I-, P- and O-labels (as defined in Definition 2.5), E ranges over *active contexts*[9], and R over redexes and head-terminal ground-type terms, as follows:

$$\frac{\ell \text{ is } I\text{- or } P\text{-type}\quad R \overset{\ell}{\longrightarrow\!\triangleright} R'}{(E,R)\overset{\ell}{\Rightarrow}(E,R')}$$

$$\frac{\ell = (f,i) \text{ is } O\text{-type}}{(E,f\,\overline{N})\overset{\ell}{\Rightarrow}(E[f\,N_1\cdots N_{i-1}\,[\text{-}]\,N_{i+1}\cdots N_{ar(f)}],N_i)}$$

Thus, suppose $(E,R)\overset{\ell}{\Rightarrow}(E',R')$; it follows from definition that if ℓ is I- or P-type, then $E[R]\longrightarrow E[R']$ (i.e. $E = E'$); otherwise ℓ is O-type and $E[R] = E'[R']$. Set $E_0 = [\,]$ and $R_0 = \mathcal{F}_1^{\perp,1}\overline{\Omega_{n-1}}\,\overline{\Omega_{n-1}}\cdots\overline{\Omega_0}$ (note that $S \longrightarrow E_0[R_0]$). Thanks to Theorem 4.1, we can now prove the following lemma (from which the Equi-Expressivity Theorem 1 follows):

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

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

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 $\Xi_{(q,\theta)}$

Lemma 4.2 *There is a 1-1 correspondence between (finite or infinite) computation path of $\mathcal{A}$ of form $\gamma_0 \overset{\ell_0}{>} \gamma_1 \overset{\ell_1}{>} \gamma_2 \overset{\ell_2}{>} \cdots$ and $\overset{\ell}{\Rightarrow}$-reduction sequences $(E_0,R_0) \overset{\ell_0}{\Rightarrow} (E_1,R_1) \overset{\ell_1}{\Rightarrow} (E_2,R_2) \overset{\ell_2}{\Rightarrow} \cdots$ such that for every $i \geq 0$, if R_i is head-normal, then R_i represents γ_i.*

Theorem 4.3 (Equi-Expressivity 1) *Let $\mathcal{A}$ be a tree-generating CPDA. The recursion scheme $G_{\mathcal{A}}$ (as defined in Definition 4.1) generates the same Σ-labelled tree as the CPDA $\mathcal{A}$.*

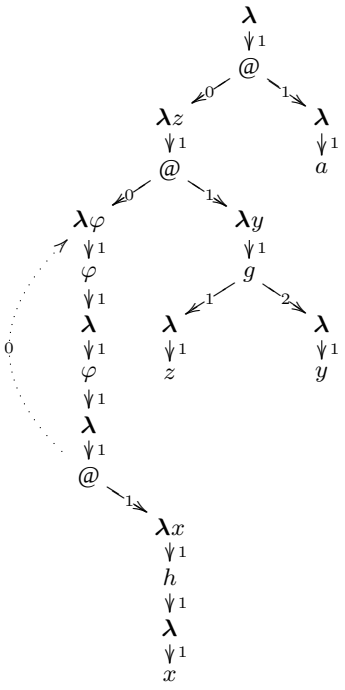
5 From recursion schemes to CPDA

The previous section shows that order-n recursion schemes are at least as expressive as order-n CPDA. In this section we shall sketch a proof of the converse. Hence CPDA and recursion schemes are equi-expressive. We have already mentioned related results by Damm and Goerdt and by Knapik *et al.* Note that in both these cases, correspondence is established with recursion schemes that are subject to highly non-trivial syntactic constraints; further the translation techniques depend on the constraints in a crucial way. Our translation from recursion schemes to CPDA is novel; it is based on (innocent) game semantics [14] and, in particular, the notions of *long transform* and *traversal* introduced in [19].

Let G be an order-n recursion scheme. The long transform of G, written $\overline{G}$, is another recursion scheme (of order 0) obtained from G by a series of syntactic transformations. First we replace the right-hand sides e of all G-rules by their η-long forms[10] $\ulcorner e \urcorner$. Then explicit application symbols are introduced: Each ground-type subterm $Fe_1\cdots e_n$, where F is a non-terminal, is replaced by $@_A F e_1 \cdots e_n$ for a suitable type A. Finally, to arrive at $\overline{G}$, we *curry* each of the transformed rules: $F\xi_1\cdots\xi_n \to e'$ is replaced by $F \to \boldsymbol{\lambda}\xi_1\cdots\xi_n.e'$. By renaming we can ensure that for each variable name φ_i there is a unique node $\boldsymbol{\lambda}\overline{\varphi}$ such that φ_i occurs in $\overline{\varphi}$. E.g. the long transform of the scheme from Example 3.1 is

$$\begin{cases} S &= \boldsymbol{\lambda}.@\,H\,(\boldsymbol{\lambda}.a) \\ H &= \boldsymbol{\lambda}z.@\,F\,(\boldsymbol{\lambda}y.g\,(\boldsymbol{\lambda}.z)\,(\boldsymbol{\lambda}.y)) \\ F &= \boldsymbol{\lambda}\varphi.\varphi\,(\boldsymbol{\lambda}.\varphi\,(\boldsymbol{\lambda}.@\,F\,(\boldsymbol{\lambda}x.h\,(\boldsymbol{\lambda}.x)))). \end{cases}$$

Given $\overline{G}$, we further define a labelled directed graph $\mathsf{Gr}(G)$, which will serve as a blueprint for the eventual definition of $\mathsf{CPDA}(G)$, the CPDA corresponding to G. To construct $\mathsf{Gr}(G)$, we first take the forest consisting of all syntactic trees of the right-hand sides of $\overline{G}$. We orient the edges towards the leaves and enumerate the outgoing edges of any node from 1 to $ar(f)$, where f is the node label, except that edges from nodes labelled by @ are numbered from 0. Let us write $v = E_i(u)$ iff (u,v) is an edge enumerated by i. Next, for any non-terminal F, we identify ("glue together") the root rt_F of the syntactic tree of the right-hand side of the rule for F with all nodes labelled F (which were leaves in the forest). The node rt_S, where S is the start symbol of $\overline{G}$, will be called the root of $\mathsf{Gr}(G)$. The graph $\mathsf{Gr}(G)$ for the order-2 recursion scheme in Example 3.1 is given on the right.



We are now ready to describe $\mathsf{CPDA}(G)$. The set of nodes of $\mathsf{Gr}(G)$ will become the stack alphabet of $\mathsf{CPDA}(G)$. The initial configuration will be the n-stack $push_1^{v_0,1} \perp_n$, where v_0 is the root of $\mathsf{Gr}(G)$. For ease of

[10] Given $\dagger s_1\cdots s_m : (A_1,\cdots,A_n,o)$, we define $\ulcorner \dagger\, s_1\cdots s_m \urcorner = \boldsymbol{\lambda}\varphi_1\cdots\varphi_n.\dagger \ulcorner s_1\urcorner\cdots\ulcorner s_m\urcorner\ulcorner\varphi_1\urcorner\cdots\ulcorner\varphi_n\urcorner$.

explanation, we define the transition map δ as a function that takes a node $u \in \mathsf{Gr}(G)$ to a sequence of stack operations, by a case analysis of the label l_u of u. When l_u is not a variable, the action is just $push_1^{v,1}$, where v is an appropriate successor of the node u. More precisely, v is defined to be $E_0(u)$ (if $l_u = @$), $E_1(u)$ (if $l_u = \boldsymbol{\lambda}\overline{\varphi}$) or $E_i(u)$ (if $l_u \in \Sigma$ and i is the direction that the automaton is to explore in the generated tree). Finally, suppose l_u is a variable φ_i and its binder is a lambda node $\boldsymbol{\lambda}\overline{\varphi}$ which is in turn a j-child. Then, assuming φ is of order $l \geq 1$, the action will be $\delta(u)$ which is defined to be

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

if $j = 0$, and

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

otherwise, where $push_1^{E_i(top_1),k}$ is defined to be the operation $s \mapsto push_1^{E_i(top_1\, s),k}\, s$. If the variable has order 0 we use pop_1^{p+1} ; $push_1^{E_i(top_1),1}$ if $j = 0$, and pop_1^{p} ; $collapse$; $push_1^{E_i(top_1),1}$ otherwise. It can be shown that runs of $\mathsf{CPDA}(G)$ are in 1-1 correspondence with traversals, as defined in [19]. Since traversals are simply *uncoverings* (in the sense of [14]) of paths in the value tree $[\![G]\!]$ we have the following theorem:

Theorem 5.1 (Equi-Expressivity 2) *For any order-n recursion scheme G, the CPDA determined by it, $\mathsf{CPDA}(G)$, generates the value tree $[\![G]\!]$.*

Remark 5.1 The proof of the preceding Theorem is effective. W. Blum [4] has constructed a tool (in $F\#$) called HOG (**H**igher-**O**rder **G**rammar), downloadable from his homepage, which implements (among other things) the algorithm that transforms an order-n recursion scheme G to the order-n CPDA, $\mathsf{CPDA}(G)$, that generates $[\![G]\!]$.

6 Games over collapsible pushdown graphs

We are interested in solving parity games over collapsible pushdown graphs i.e. we want to know whether we 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***[11] (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: $(q, s) \overset{(q',\theta)}{>} (q', s')$ iff 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[12] 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) in Table 2.

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 $\Omega : Q \to C \subset \mathbb{N}$ 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 $\Omega : V \to C$ 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 a ***n-CPDS parity game***. A play in $\mathbb{G}$ from the initial vertex $v_0 = (q_0, \bot_n)$ works as follows: the player who 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 some player if he has a winning strategy from it. See [21, 24, 23] for more details.

In this section we consider the problem:

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

The Problem ($\mathbf{P}_1$) is closely related to the following: problems:

($\mathbf{P}_2$) *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 hold at the root of the unravelling of G?*

From the well-known techniques of [11], it follows that Problem ($\mathbf{P}_1$) is polynomially equivalent to Problems ($\mathbf{P}_2$) and ($\mathbf{P}_3$); and 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 in the other direction.

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

[12] This is inspired by an example in [7].

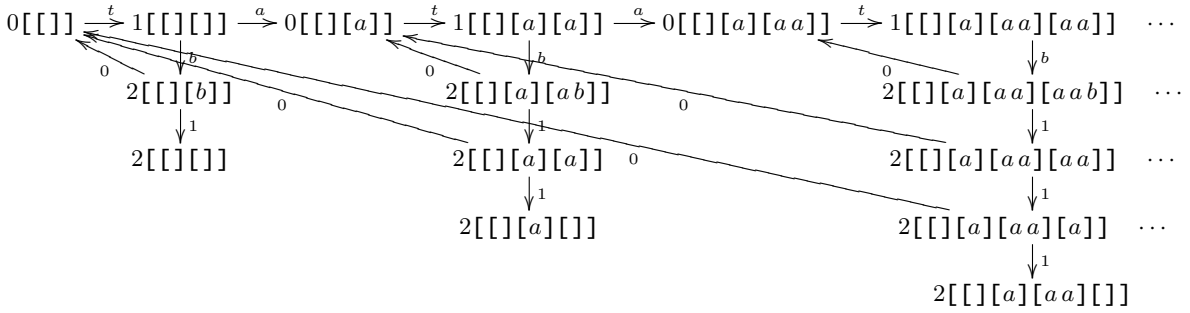


Table 2. Configuration graph of a 2-CPDS

A 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). Thus an important consequence of the Equi-Expressivity Theorems is the following.

Theorem 6.1 *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 [19], we obtain the following as an immediate consequence.

Theorem 6.2 *Problems $(\mathbf{P}_1)$, $(\mathbf{P}_2)$, $(\mathbf{P}_3)$ and $(\mathbf{P}_4)$ are decidable with complexity n-EXPTIME complete.*

Another remarkable consequence of the Equi-Expressivity Theorems is that they give totally new techniques for model-checking or solving games played on infinite structures generated by automata. In particular they lead to new proofs / optimal algorithms for the special cases that have been considered previously [22, 5, 17]. Conversely, as the Equi-Expressivity Theorems work in both directions, we note that a solution of Problem $(\mathbf{P}_1)$ would give a new proof of the decidability of Problems $(\mathbf{P}'_2)$, $(\mathbf{P}'_3)$ and $(\mathbf{P}'_4)$, and would give a new approach to problems on recursion schemes. Actually, the techniques of [22, 17] can be generalized to solve n-CPDS parity games without reference to [19]. Further they give effective winning strategies for the winning player (which was not the case in [17] where the special case $n = 2$ was considered).

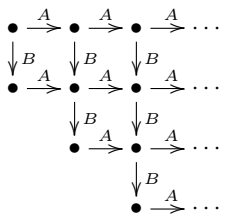
Theorem 6.3 *The problem of solving an n-CPDS parity game is n-EXPTIME complete and it can be achieved without reference to the decidability result in [19]. Further one can build an n-CPDA with output that realizes a winning strategy for the winning player.*

Remark 6.2 This result can easily be generalized to the case where the game has an arbitrary ω-regular winning condition, and is played on the ε-closure of the configuration graph of an n-CPDS graph. Consequently parity games on Caucal graphs [6, 5] are a special case of this problem.

The Caucal graphs have decidable MSO theories [6]. Do the configuration graphs of CPDS also have decidable MSO theories?

Theorem 6.4 (Undecidability) *MSO theories of configuration graphs of CPDS are undecidable. Hence the class of ε-closure of configuration graphs of CPDS strictly contains the Caucal graphs.*

For a proof, recall that MSO interpretation preserves MSO decidability. Now consider the following MSO interpretation I of the configuration graph of the 2-CPDS in Example 6.1:



$$I_A(x,y) \;=\; x \xrightarrow{C} y \wedge x \xrightarrow{R} y$$
$$I_B(x,y) \;=\; x \xrightarrow{1} y$$

with $C = \overline{1}^* \, \overline{b} \, a \, t \, b \, 1^*$ and $R = 0 \, t \, a \, \overline{0} \;\vee\; \overline{1} \, 0 \, t \, a \, \overline{0} \, 1$. 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. Observe that I's image is the "infinite half-grid" which has an undecidable MSO theory.

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 think 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 schemes on the one hand, and solvability of games on collapsible pushdown graphs on the other. We show (Theorem 6.4) that order-n CPDS are strictly more expressive than order-n pushdown systems for generating graphs.

There are a number of **further directions**:

(i) The most pressing open problem is whether order-n CPDA are equi-expressive with order-n PDA *for generating trees* (see Remark 2.6). The conjecture is that the former are strictly more expressive. Specifically the *Urzyczyn tree* is definable by a 2-CPDA [1] but we conjecture that it is not definable by an n-PDA for any $n \geq 2$.

(ii) Is it possible to give a finite description of the set of winning positions of an n-CPDS parity game? Over *generalized* configuration graphs (whose vertices are *all* configurations, not just the reachable), we believe that the set of winning positions of an n-CPDS parity game is representable by a finite automaton that reads a stack with links from bottom to top, and that, when processing a link, has access to the state it was in after reading the stack pointed to.

(iii) 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 mu-calculus decidability of trees generated by n-CPDA.

(iv) The algorithm that transforms recursion schemes to CPDA (briefly sketched in Section 5) uses ideas in game semantics. It would be an interesting (and challenging) to obtain a translation based on 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. In *Proc. FOSSACS'05*, LNCS 3411, 2005, pp. 490-501.

[2] A. Aho. Indexed grammars - an extension of context-free grammars. J. ACM 15:647-671, 1968.

[3] R. Alur and P. Madhusudan. Languages of nested trees. In *Proc. CAV'06*, 2006.

[4] W. Blum. A tool for constructing structures generated by higher-order recursion schemes and collapsible pushdown automata. `web.comlab.ox.ac.uk/oucl/work/william.blum/`, 2007.

[5] T. Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *Proc. ICALP'03*, LNCS 2719, pp. 556-569, 2003.

[6] D. Caucal. On infinite terms having a decidable monadic theory. In *Proc. MFCS'02*, LNCS 2420, pp. 165-176, 2002.

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

[8] B. Courcelle. The monadic second-order logic of graphs IX: machines and their behaviours. *TCS* 151:125-162, 1995.

[9] W. Damm. The IO- and OI-hierarchy. *TCS* 20:95-207, 1982.

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

[11] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. FOCS'91*, pp. 368-377, 1991.

[12] J. Engelfriet. Interated stack automata and complexity classes. *Info. & Comp.* 95:21-75, 1991.

[13] M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. 2007. 56 pages, downloadable from `users.comlab.ox.ac.uk/luke.ong/publications/cpda-long.pdf`.

[14] 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. *Info. & Comp.* 163:285-408, 2000.

[15] T. Knapik, D. Niwiński, and P. Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *Proc. TLCA'01*, LNCS 2044, pp. 253-267, 2001.

[16] T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Proc. FOSSACS'02*, LNCS 2303, pp. 205-222, 2002.

[17] T. Knapik, D. Niwiński, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *Proc. ICALP'05*, LNCS 3580, pp. 1450-1461, 2005.

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

[19] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proc. LICS'06*, pp. 81-90, 2006.

[20] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. AMS* 141:1-35, 1969.

[21] W. Thomas. On the synthesis of strategies in infinite games. In *Proc. STACS'95*, LNCS 900, pp. 1-13, 1995.

[22] I. Walukiewicz. Pushdown processes: games and model-checking. *Info. & Comp.* 157:234-263, 2001.

[23] I. Walukiewicz. A landscape with games in the backgroung. In *Proc. LICS'04*, pp. 356-366, 2004.

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