Collapsible Pushdown Automata and Recursion Schemes
(Extended abstract)

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 \( \text{push}_i \) and \( \text{pop}_i \), CPDA have an important operation called \( \text{collapse} \), whose effect is to “collapse” a stack \( s \) to the prefix as indicated by the link from the top\(_i\)-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.

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

[We direct readers to the (downloadable) long version [10] of this paper in which all proofs are presented.]

∗Oxford University Computing Laboratory (OUGC). Matthew.Hague@comlab.ox.ac.uk
†OUGC. Andrzej.Murawski@comlab.ox.ac.uk
‡OUGC. Luke.Ong@comlab.ox.ac.uk
§LIAFA (CNRS and Université Paris VII). Olivier.Serre@liafa.jussieu.fr
1 Introduction

Higher-order pushdown automata (PDA) were first introduced by Maslov [15, 16] 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 [7, 8]: 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 [13]. 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 [18], algebraic [6] and hyperalgebraic trees [12]. 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. 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 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 [14], 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 \( \text{push}_i \) and \( \text{pop}_i \), 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 \( \text{top}_1 \)-symbol of \( s \). The main result (Theorems 3 and 4) of this paper 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 [17] 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 [20, 3, 14]. 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 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

---

1Higher-order recursion schemes are essentially simply-typed lambda calculus with general recursion and uninterpreted first-order function symbols
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 [17].

In contrast to higher-order pushdown graphs (which do have decidable MSO theories [4]), 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)

Fix a stack alphabet \( \Gamma \) 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-\(\bot\) \( \Gamma\)-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 \( \text{ord}(s) \); and we shall abbreviate order-\(n\) stack to \( n \)-stack. As usual, the bottom-of-stack\(^2\) 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 to avoid clutter (e.g. writing \([ ] [a] \) instead of \([ [\bot] [\bot a] \])

The set \( \text{Op}_n \) of order-\(n\) stack operations consists of the following four types of operations:

1. \( \text{pop}_k \) for each \( 1 \leq k \leq n \)
2. \( \text{collapse} \)
3. \( \text{push}_{a,k}^i \) for each \( 1 \leq k \leq n \) and each \( a \in (\Gamma \setminus \{\bot\}) \), and
4. \( \text{push}_j \) for each \( 2 \leq j \leq n \).

First we introduce auxiliary operations \( \text{top}_i \), which takes a stack \( s \) and returns the top \((i - 1)\)-stack of \( s \); and \( \text{push}_{i}^a \), 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 \( 1 \leq i \leq \text{ord}(s) \), we define

\[
\text{top}_i [s_1 \cdots s_{l+1}]_s = \begin{cases} 
  s_{l+1} & \text{if } i = \text{ord}(s) \\
  \text{top}_i s_{l+1} & \text{if } i < \text{ord}(s)
\end{cases}
\]

(1)

\[
\text{push}_{i}^a [s_1 \cdots s_{l+1}]_s = \begin{cases} 
  [s_1 \cdots s_l \text{push}_{i}^a s_{l+1}] & \text{if } \text{ord}(s) > 1 \\
  [s_1 \cdots s_{l+1} a] & \text{if } \text{ord}(s) = 1
\end{cases}
\]

(2)

We can now explain the four operations in turn. For \( i \geq 1 \) the \( \text{pop}_i \) operation takes a stack and returns it with its top \((i - 1)\)-stack removed. Let \( 1 \leq i \leq \text{ord}(s) \) we define

\[
\text{pop}_i [s_1 \cdots s_{l+1}]_s = \begin{cases} 
  [s_1 \cdots s_l] & \text{if } i = \text{ord}(s) \text{ and } l \geq 1 \\
  [s_1 \cdots s_l \text{pop}_i s_{l+1}] & \text{if } i < \text{ord}(s)
\end{cases}
\]

(3)

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) \( \text{pop} \) operations.

Take an \( n \)-stack \( s \). For \( i \geq 2 \) to construct \( \text{push}_{i}^a \) we first attach a link from a 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 \( \text{collapse} \), suppose the \( \text{top}_1 \)-symbol of \( s \) has a link to (a particular copy of) the \( k \)-stack \( u \) somewhere in \( s \). Then \( \text{collapse} \) causes \( s \) to “collapse” to the prefix \( s_0 \) of \( s \) such that \( \text{top}_{k+1} s_0 \) is that copy of \( u \). Finally, for \( j \geq 2 \), the (higher-order) \( \text{push}_j \) operation simply takes a stack \( s \) and duplicates the top \((j - 1)\)-stack of \( s \), preserving its link structure.

\(^2\)Thus we require an order-\(I\) stack to be a non-null sequence \([a_1 \cdots a_l]\) of \( \Gamma \)-symbols such that for all \( 1 \leq i \leq l \), \( a_i = \bot \) iff \( i = 1 \).
One way to define these stack operations formally is to work with an appropriate numeric representation of the links. Knapik et al. have shown how this can be done in the order-2 case in [14]. 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 \( \text{collapse } s_0 = \text{pop}$

\[ k \]

means \( \text{pop}_j \cdots \text{pop}_1 \). 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 if $j = 1$ then $k = 1$. Even though there is no link from $\bot$, for technical convenience, we assume if $a = \bot$ then $j = k = 1$.

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 (3) and (2) respectively). Let $1 \leq i \leq \text{ord}(s)$ and $2 \leq j \leq \text{ord}(s)$ we define

\[
\begin{align*}
\text{collapse } s & = \text{pop}^i s \quad \text{where top$_1$ } s = a^{(i,j)} \\
\text{push}_1^{b,i} s & = \text{push}_1^{b(i,j)} s \\
\text{push}_j[s_1 \cdots s_{l+1}] & = \begin{cases} 
[s_1 \cdots s_{l+1} s^j_{l+1}] & \text{if } j = \text{ord}(s) \\
[s_1 \cdots s_{l+1} \text{push}_j s_{l+1}] & \text{if } j < \text{ord}(s)
\end{cases}
\end{align*}
\]

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

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

\[
\begin{align*}
\text{push}_1^{b,2} s & = \{ [[a]] [[a]] [[ab^{(2,1)}]] \} \\
\text{push}_1^{b,3} \left( \text{push}_1^{b,2} s \right) & = \{ [[a]] [[a]] [[ab^{(2,1)} c^{(3,1)}]] \} \\
\text{push}_2 \left( \text{push}_1^{b,3} \left( \text{push}_1^{b,2} s \right) \right) & = \{ [[a]] [[a]] [[ab^{(2,1)} c^{(3,1)}] [ab^{(2,2)} c^{(3,1)}]] \} \\
\text{push}_3 \left( \text{push}_1^{b,3} \left( \text{push}_1^{b,2} s \right) \right) & = \{ [[a]] [[a]] [[ab^{(2,1)} c^{(3,1)}] [ab^{(2,2)} c^{(3,2)}]] \} \\
\text{collapse} \left( \text{push}_2 \left( \text{push}_1^{b,3} \left( \text{push}_1^{b,2} s \right) \right) \right) & = \text{collapse} \left( \text{push}_3 \left( \text{push}_1^{b,3} \left( \text{push}_1^{b,2} s \right) \right) \right) = \{ [[a]] \}
\end{align*}
\]

**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 [14].

**Definition 2.2** A tree-generating order-$n$ collapsible pushdown automaton (n-CPDA) is a 5-tuple \( \langle \Sigma, \Gamma, Q, \delta, q_0 \rangle \) where $\Sigma$ is a ranked alphabet (i.e. each $\Sigma$-symbol $f$ has an arity $\text{ar}(f) \geq 0$, $\Gamma$ is a stack alphabet, $Q$ is a finite set of states, $q_0$ is the initial state, and $\delta : Q \times \Gamma \rightarrow (Q \times \text{Op}_n + \{(f; q_1, \cdots, q_{\text{ar}(f)}) : f \in \Sigma, q_i \in Q\})$ is the transition function.

Configurations of an $n$-CPDA are pairs of the form $(q, s)$ where $q \in Q$ and $s$ is an $n$-stack over $\Gamma$; we call $(q_0, \bot_n)$ the initial configuration. A generalized configuration (ranged over by $\gamma, \gamma_i$, etc.) is either a configuration or a triple of the form $(f; q_1, \cdots, q_{\text{ar}(f)}; s)$. We define $\overset{\ell}{\rightarrow}$, a one-step labelled transition relation of $\mathcal{A}$ over generalized configuration by clauses, one for each of the three types$^4$ of labels $\ell$ that annotate $\overset{\ell}{\rightarrow}$.

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

$^4$I for internal or hidden Player-move, $P$ for Player-move, and $O$ for Opponent-move.
namely, $I$, $P$ and $O$:

$I. \quad (q, s) \not> (q', s')$ if for some $\theta \in Op \rho$, we have $\delta(q, top_1 s) = (q', \theta)$ and $s' = \theta(s)$

$P. \quad (q, s) \not> (f; q_1, \ldots, q_{ar(f)}; s)$ if $\delta(q, top_1 s) = (f; q_1, \ldots, q_{ar(f)})$

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

A computation path of an $n$-CPDA $A$ is a finite or infinite transition sequence of the form $\rho = \gamma_0 \ell_0 > \gamma_1 \ell_1 > \gamma_2 \ell_2 > \cdots$ where $\gamma_0$ 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 $(I^* P O)^e + (I^* P O)^* I^e$ if the sequence is infinite, and $(I^* P O)^* I^e (e + P + P O)$ if the sequence is finite. The $\Sigma$-projection of $\rho$ is the subsequence $\ell_1 \ell_2 \ell_3 \cdots$ of labels of the shape $(f, i)$ (in which case $ar(f) \geq 1$) or of the shape $(f; e)$ (in which case $ar(f) = 0$, and the label marks the end of the $\Sigma$-projection). We say the CPDA $A$ generates the $\Sigma$-labelled tree $t$ just in case the branch language of $t$ coincides with the $\Sigma$-projection of computation paths of $A$.

3 Recursion schemes

Types are generated from the base type $o$ using the arrow constructor $\to$. Every type $A$ can be written uniquely as $A_1 \to \cdots \to A_n \to o$ (arrows associate to the right), for some $n \geq 0$ which is called its arity; we shall often write $A$ simply as $(A_1, \ldots, A_n, o)$. We define the order of a type by $ord(o) = 0$ and $ord(A \to B) = \max\{ord(A) + 1, ord(B)\}$. Let $\Sigma$ be a ranked alphabet i.e. each $\Sigma$-symbol $f$ has an arity $ar(f) \geq 0$ which determines its type $(a_1, \ldots, a_o, o)$. Further we shall assume that each symbol $f \in \Sigma$ is assigned a finite set $\text{Dom}(f)$ of $ar(f)$ directions, and we define $\text{Dir}(\Sigma) = \bigcup_{f \in \Sigma} \text{Dir}(f)$. Let $D$ be a set of directions; a $D$-tree is just a prefix-closed subset of $D^*$, the free monoid of $D$. A $\Sigma$-labelled tree is a function $t : \text{Dom}(t) \longrightarrow$ such that $\text{Dom}(t)$ is a $\text{Dir}(\Sigma)$-tree, and for every node $\alpha \in \text{Dom}(t)$, the $\Sigma$-symbol $t(\alpha)$ has arity $k$ if and only if $\alpha$ has exactly $k$ children and the set of its children is $\{ \alpha i : i \in \text{Dir}(t(\alpha)) \}$ (i.e. $t$ is a ranked tree). We shall assume that the ranked alphabet $\Sigma$ contains a distinguished nullary symbol $\bot$ which will be used exclusively to label "undefined" nodes.

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

For each type $A$, we assume an infinite collection $\text{Var}^{A}$ of variables of type $A$, and write $\text{Var}$ to be the union of $\text{Var}^{A}$ 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 $G = (\Sigma, N, R, S)$ where $\Sigma$ is a ranked alphabet of terminals; $N$ is a set of typed non-terminals; $S \in N$ is a distinguished start symbol of type $o$; $R$ is a finite set of rewrite rules – one for each non-terminal $F : (A_1, \ldots, A_n, o) - \to$ of the form $F \xi_1 \cdots \xi_n - \to e$ where each $\xi_i$ is in $\text{Var}^{A_i}$, and $e \in T^{o}(\Sigma) \cup N \cup \{\xi_1, \ldots, \xi_n\}$ i.e. $e$ is an applicative term of type $o$ generated from elements of $\Sigma \cup N \cup \{\xi_1, \ldots, \xi_n\}$. The order of a recursion scheme is the highest order of the types of its non-terminals.

We use recursion schemes as generators of $\Sigma$-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 $\Sigma$-labelled tree,

\footnote{The branch language of $t : \text{Dom}(t) \longrightarrow$ 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 $(f_1, d_1) \cdots (f_n, d_n)\theta$ just if for $0 \leq i < n$, we have $t(d_1 \cdots d_i) = f_{i+1}$ and $t(d_1 \cdots d_n) = \theta$.}

\footnote{Applicative terms are constructed from the generators using the application rule: if $d : A \to B$ and $e : A$ then $(de) : B$. Standardly we identify finite $\Sigma$-labelled terms with applicative terms of type $o$ generated from $\Sigma$-symbols endowed with 1st-order types as given by their arities.}
constructed from the terminals in \( \Sigma \), 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. [13] for a formal definition.

**Example 3.1** Let \( G \) be the order-2 unsafe (in the sense of [13]) recursion scheme with rewrite rules where \( z : o \) and \( \varphi : (o, o) \):

\[
\begin{align*}
S & \rightarrow H a \\
H z & \rightarrow F (g z) \\
F \varphi & \rightarrow \varphi (F (F h))
\end{align*}
\]

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 \( \Sigma \)-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 \( G_A \) (where \( n \geq 0 \)) such that \( \mathcal{A} \) and \( G_A \) define the same \( \Sigma \)-labelled tree (Theorem 3). We begin by introducing a method to represent higher-order stacks and configurations by applicative terms constructed from non-terminals of the same order. Our approach simplifies the (order-2) translation in [14] and generalizes it to all finite orders in a non-trivial way.

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 \( n + 1 = n^m \rightarrow n \) where \( n^m = n \times \cdots \times n \). Thus

\[ n + 1 = n^m \rightarrow (n - 1)^m \rightarrow \cdots \rightarrow 0^m \rightarrow 0. \]

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

\[ F^{a,e}_p : (n - e)^m \rightarrow (n - 1)^m \rightarrow \cdots \rightarrow 0^m \rightarrow 0 \]

that represents the symbol \( a \) with a link of order \( e \) (in state \( p \)). Note that the type of \( F^{a,e}_p \) is not homogeneous in the sense of Knapik et al. [13]. In addition, for each \( 0 \leq i \leq n - 1 \), we introduce a non-terminal \( M_i : i \), and fix a start symbol \( S : 0 \). Let \( N_A \) 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 \( \langle P(i) \mid i \rangle \) to mean the \( m \)-tuple \( \langle P(1), \cdots, P(m) \rangle \). E.g. \( \langle F^{a,e}_i \mid i \rangle \) means \( \langle F^{a,e}_1, \cdots, F^{a,e}_m \rangle : ((n - e)^m \rightarrow n)^m \).

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 \( F^{a,e}_p \) i.e. \( M \) has the shape \( F^{a,e}_p \, \langle M_{n-1}, \cdots, M_{n-j} \rangle \), for some \( a, e \) and \( p \) and for some vectors of terms \( L, M_{n-1}, \cdots M_{n-j} \) of the appropriate types; we shall call \( F^{a,e}_p \) 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

\[ \langle F^{a,e}_i \, \langle L \, M_{n-1}, \cdots M_{n-j} \rangle \mid i \rangle : (n - j)^m \]

to represent \( j \)-configurations. Suppose the configuration \( (p, s) \) is represented by \( F^{a,e}_p \, \langle L \, M_{n-1}, \cdots M_{n-j} \rangle : 0 \). The idea is that for \( 1 \leq k \leq n \), we have

\[
\begin{align*}
(p, top_k s) & \text{ is represented by } \langle F^{a,e}_p \, \langle L \, M_{n-1}, \cdots M_{n-(k-1)} \rangle \rangle : n - (k - 1) \\
(p, pop_k s) & \text{ is represented by } \langle M_{n-k,p} \, M_{n-k-1} \cdots M_0 \rangle : 0 \\
(p, collapse s) & \text{ is represented by } \langle L_p \, M_{n-e-1} \cdots M_0 \rangle : 0
\end{align*}
\]

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 \( L : (n - e)^m \) – is represented by \( F^{a,e}_p \, \langle L \rangle : n \).
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 \( \theta \) over \( Op_n \):

\[
F_p^{a,e} \Phi \Psi_{n-1} \cdots \Psi_0 \xrightarrow{(q, \theta)} \Xi_{(q, \theta)}
\]

where for each \( 0 \leq j \leq n-1 \), we have \( \Psi_j = \Psi_j, \cdots, \Psi_{jm} \) is a vector of variables, with each \( \Psi_{jm} : j \); similarly \( \Phi = \Phi_1, \cdots, \Phi_m \) is a vector of variables, with each \( \Phi_i : n - e \). The shape of \( \Xi_{(q, \theta)} \) depends on the pair \((q, \theta)\) 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 \( (q, \theta) \subseteq T^0(\mathcal{N}_A) \times T^0(\mathcal{N}_A) \). Informally we define \( M^{(q, \theta)} \sim M' \) 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 \( (q, \theta) \) is a partial function, we shall write \( M^{(q, \theta)} \) to mean \( M' \). We shall write \( \theta \) to mean the set of all transitions \( M^{(q, \theta)} \) that preserves the state \( q \) of \( M \). Let \( \alpha = \theta_1 ; \cdots ; \theta_l \) be a (composite) sequence of stack operations. We write \( \alpha \subseteq T^0(\mathcal{N}_A) \times T^0(\mathcal{N}_A) \) to be the sequential composition of the partial function \( \theta_1, \cdots, \theta_l \) (in this order).

The position of a given stack symbol in an \( n \)-stack \( s \) can be described by a sequence of \( \text{pop} \) operations that can “collapse” the stack up to the point where that position becomes the top1-symbol. For example, the position of \( b \) in the 2-stack \( [[aa] [ab]a] [a] [a] \) is \( \text{pop}_2 ; \text{pop}_1 \). In general such sequences are not unique, though they can be normalized to one in which the respective orders of the \( \text{pop} \) 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 \( \alpha \), if the \( \text{top}_1 \)-symbol of \( \alpha \) \( s \) is \( a^{(j,k)} \), then the head non-terminal of \( M \alpha \) is \( F_p^{a,j} \); further \( (M \alpha)^{\text{pop}_k} \) is \( (M \alpha)^{\text{coll}} \), and it represents the configuration \((p, \text{collapse}(\alpha s))\). Note that \( F_p^{a,1} \Omega_{n-1-1} \Omega_{n-1-2} \cdots \Omega_{n-j} : n-j \) represents the \( j \)-configuration \((p, \perp_{n-j})\). The following Theorem confirms our notion of representation is the right one.

**Theorem 1 (Correctness)** Let \( M \) be a ground-type term, \((p, s)\) be a configuration, and \( \theta \) be a stack operation. Suppose \( M \) represents \((p, s)\), If \( M \xrightarrow{\theta} M' \) then \( M' \) represents the configuration \((p, s)\).

**Definition 4.1** Fix a tree-generating order-\( n \) CPDA \( \mathcal{A} = (\Sigma, \Gamma, Q, \delta, q_0) \) 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: \( S \rightarrow F_1^{1,1} \Omega_{n-1} \Omega_{n-2} \cdots \Omega_0 \), and two types of rewrite rules (according to the type of their label), namely, \( I \) and \( P \): \n
**I.** For each \((q, \theta) \in \delta(p, a) \) and \( 1 \leq e \leq n \), there is an \( I \)-type rewrite rule \( \mathcal{F}_{p}^{a,e} \Phi \Psi_{n-1} \cdots \Psi_0 \xrightarrow{(q, \theta)} \Xi_{(q, \theta)} \), where \( \Xi_{(q, \theta)} \) is as given in Table 1.

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

\[
\mathcal{F}_{p}^{a,e} \Phi \Psi_{n-1} \cdots \Psi_0 \xrightarrow{(f, \pi)} f(\mathcal{F}_{q_1}^{a,e} \Phi \Psi_{n-1} \cdots \Psi_0) \cdots (\mathcal{F}_{q_{a(f)}}^{a,e} \Phi \Psi_{n-1} \cdots \Psi_0).
\]
We write \( \rightarrow \subseteq T^0(\Sigma \cup N_A) \times T^0(\Sigma \cup N_A) \) for the one-step reduction relation\(^7\) 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 \( R \rightarrow R' \) is a \( P \)-type or \( I \)-type according to the type of \( \ell \); by abuse of notation, we shall write \( R \rightarrow^{\ell} R' \). 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 \( \Omega \)) or it is head normal (i.e. the head symbol is of the form \( F_{p+} \)), 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 \( \rightarrow^\ell \) over pairs of the form \((E, R)\) where \( \ell \) ranges over \( I, P \) and \( O \)-labels (as defined in Definition 2.2), \( E \) ranges over active contexts\(^8\), and \( R \) over redexes and head-term ground-type terms, as follows:

\[
\ell \text{ is } I- \text{ or } P- \text{type } \quad R \rightarrow^\ell R' \\
\begin{array}{ll}
(E, R) \rightarrow^\ell (E, R') & (E, f N_1 \cdots N_{ar(f)}) \rightarrow^\ell (E[f N_1 \cdots N_i \vDash N_{i+1} \cdots N_{ar(f)}], N_i)
\end{array}
\]

Thus, suppose \((E, R) \rightarrow^\ell (E', R')\); it follows from definition that if \( \ell \) is \( I \)- or \( P \)-type, then \( E[R] \rightarrow E'[R'] \) (i.e. \( E = E' \)); otherwise \( \ell \) is \( O \)-type and \( E[R] = E'[R'] \). Set \( E_0 = [] \) and \( R_0 = F_{1}^{1} \Omega_{n-1} \Omega_{n-2} \cdots \Omega_{0} \) (note that \( S \rightarrow E_0[R_0] \)). Thanks to Theorem 1, we can now prove the following lemma (from which the Equi-Expressive Theorem 1 follows):

**Lemma 2** There is a 1-1 correspondence between (finite or infinite) computation path of \( A \) of form \( \gamma_0 \rightarrow^\ell \gamma_1 \rightarrow^\ell \gamma_2 \rightarrow^\ell \cdots \) and \( \rightarrow^\ell \)-reduction sequences \((E_0, R_0) \rightarrow^\ell (E_1, R_1) \rightarrow^\ell (E_2, R_2) \cdots\) such that for every \( i \geq 0 \), if \( R_i \) is head-normal, then \( R_i \) represents \( \gamma_i \).

**Theorem 3** (Equi-Expressivity 1) Let \( A \) be a tree-generating CPDA. The recursion scheme \( G_A \) (as defined in Definition 4.1) generates the same \( \Sigma \)-labelled tree as the CPDA \( A \).

### 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 [11] and, in particular, the notions of long transform and traversal introduced in [17].

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 \( \eta \)-long forms\(^9\) \( \eta^\gamma \). Then explicit application symbols are introduced: Each ground-type subterm \( Fe_1 \cdots e_n \), where \( F \) is a non-terminal, is replaced by \( \@_A Fe_1 \cdots e_n \) for a suitable type \( A \). Finally, to arrive at \( \overline{G} \), we currying each of the transformed rules: \( F \xi_1 \cdots \xi_n \rightarrow e' \) is replaced by \( F \rightarrow \lambda \xi_1 \cdots \xi_n. e' \). By renaming we can ensure that for each variable name \( \varphi_i \) there is a unique node \( \lambda \varphi_i \) such that \( \varphi_i \) occurs in \( \varphi \).

The long transform of the scheme from Example 3.1 is \( S = \lambda \@ H (\lambda.a), H = \lambda z. @ F (\lambda y.g (\lambda.z) (\lambda.y)), F = \lambda \varphi. \varphi (\lambda. \varphi (\lambda. @ F (\lambda x. h (\lambda.x)))) \).

---

\(^7\)When defining \( \rightarrow \) and the tree generated by the recursion scheme \( G_A \), we ignore the labels \( \ell \) that annotate the rules \( \rightarrow^\ell \).

\(^8\)An active context is just an ground-type applicative term that contains a ground-typed hole, into which a term may be inserted.

\(^9\)Given \( \vdash A \cdots s_m : (A_1, \cdots, A_n, o) \), we define \( \vdash \vdash s_1 \cdots s_m \gamma = \lambda \varphi_1 \cdots \varphi_n. \vdash s_1 \cdots \vdash s_m \cdash \varphi_1 \cdots \cdash \varphi_n \).
Given $\overline{G}$, we further define a labelled directed graph $Gr(G)$, which will serve as a blueprint for the eventual definition of CPDA($G$), the CPDA corresponding to $G$. To construct $Gr(G)$, 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. Exceptionally, 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 $Gr(G)$. The graph $Gr(G)$ for Example 3.1 is given below.

We are now ready to define CPDA($G$). The set of nodes of $Gr(G)$ will become the stack alphabet of CPDA($G$). The initial configuration will be the $n$-stack $push_{v_0,1}n$, where $v_0$ is the root of $Gr(G)$. For ease of explanation, we define the transition map $\delta$ as a function that takes a node $u \in Gr(G)$ 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 $push_{v,1}^n$, where $v$ is an appropriate successor of the node $u$. More precisely, $v$ is defined to be $E_0(u)$ (for $l_u = @$), $E_1(u)$ (for $l_u = \lambda \overline{\varphi}$) 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 $\varphi_i$ and its binder is a lambda node $\lambda \overline{\varphi}$ which is in turn a $j$-child. Then, assuming $\varphi$ is of order $l \geq 1$, the action will be

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

where $push_1^{E_i(top_1),k}$ 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 $pop_{p+1}^1 \; ; \; push_1^{E_i(top_1),1}$ if $j = 0$, and $pop_1^p \; ; \; collapse \; ; \; push_1^{E_i(top_1),1}$ otherwise. It can be shown that runs of CPDA($G$) in 1-1 correspondence with traversals, as defined in [17]. Since traversals are simply uncoverings (in the sense of [11]) of paths in the value tree $\lfloor G \rfloor$ we have the following theorem:

**Theorem 4 (Equi-Expressivity 2)** For any order-$n$ recursion scheme $G$, the CPDA determined by it, CPDA($G$), generates the value tree $\lfloor G \rfloor$.

### 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\(^{10}\) (n-CPDS) is given by a quadruple $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$ where $\Gamma$ 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 $\Gamma$. We define a one-step labelled transition relation of the CPDS $\mathcal{A}$, written $\triangleright$ where $\ell \in Q \times Op_n$, which is a family of binary relations over configurations, as follows: $(q, s) \triangleright (q', s')$ 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 $\triangleright$ restricted to the reachable configurations.

\(^{10}\)We use collapsible pushdown system (as opposed to automaton) whenever the device is used to generate a graph.
Example 6.1 Take the 2-CPDS with state-set \( \{0, 1, 2\} \), stack alphabet \( \{a, b, \perp\} \) and transition relation given by

\[
(0, -1, t), (1, -0, a), (1, -2, b), (2, \uparrow, 2), (2, \uparrow, 0, 0)
\]

where \(-\) means any symbol, \(\uparrow\) means any non-\(\perp\) symbol, and \(t, a, b, 0\) and \(1\) are shorthand for the stack operations \(push_2, push_a, push_b, collapse\) and \(pop_1\) respectively. We present its configuration graph (with edges labelled by stack operations only) as follows:

\[
\begin{array}{c}
\text{0[[]]} \xrightarrow{a} \text{1[[]]} \xrightarrow{a} \text{0[[]][a]} \xrightarrow{a} \text{1[[]][a][a][a]} \xrightarrow{\uparrow} \text{1[[]][a][a][a]} \\
\text{2[[]]} \xrightarrow{\uparrow} \text{0[[]]} \xrightarrow{\uparrow} \text{2[[]][a][a]} \\
\text{2[[]][b]} \xrightarrow{\uparrow} \text{0[[]][a][a]} \\
\text{2[[]][a][a]} \xrightarrow{\uparrow} \text{2[[]][a][a]} \\
\text{2[[]][a]} \xrightarrow{\uparrow} \text{2[[]][a][a]} \\
\text{2[[]][a][a]} \xrightarrow{\uparrow} \text{2[[]][a][a][a][a]} \\
\end{array}
\]

Let \(G = (V, A)\) denote the configuration graph of \(A\), let \(Q_E \cup Q_A\) be a partition of \(Q\) and let \(\Omega : Q \rightarrow C \subset \mathbb{N}\) be a colouring function. Altogether they define a partition \(V_E \cup V_A\) of \(V\) whereby a vertex belongs to \(V_E\) iff its control state belongs to \(Q_E\), and a colouring function \(\Omega : V \rightarrow C\) where a vertex is assigned the colour of its control state. The structure \(G = (G, V_E, V_A)\) is an \(n\)-CPDS game graph and the pair \(G = \langle G, \Omega \rangle\) is a \(n\)-CPDS parity game. A play in \(G\) from the initial vertex \(v_0 = (q_0, \perp)\) works as follow: the player that controls \(v_0\) (Éloïse if \(v_0 \in V_E\) 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 \(v_0v_1 \cdots\) and is won by Éloïse iff \(\lim \inf (\Omega(v_i) : i \geq 0)\) is even. Finally, \(v_0\) is winning for some player if he has a winning strategy from it. See [19, 22, 21] for more details.

In this section we consider the following problem:

\((P_1)\) Given an \(n\)-CPDS parity game decide if Éloïse has a winning strategy from the initial configuration.

From the well-known techniques of [9], it follows that (i) Problem \((P_1)\) is polynomially equivalent to Problems \((P_2)\) and \((P_3)\) in the following: and (ii) Problem \((P_1)\) is equivalent to Problem \((P_4)\) – the reduction from \((P_1)\) to \((P_4)\) is polynomial, but non-elementary one in the other direction:

\[(P_2)\) Given an \(n\)-CPDS graph \(G\), and a mu-calculus formula \(\varphi\), does \(\varphi\) hold at the initial configuration of \(G\)?

\[(P_3)\) Given an alternating parity tree automaton and \(n\)-CPDS graph \(G\), does it accept the unravelling of \(G\)?:

\[(P_4)\) Given an MSO formula \(\varphi\) and an \(n\)-CPDS graph \(G\), does \(\varphi\) 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). Thus an important consequence of the Equi-Expressivity Theorems is the following.

Theorem 5 Let \(t\) be a tree generated by an order-\(n\) recursion scheme. Consider the following problems:

\[(P'_2)\) Given \(t\) and a modal mu-calculus formula \(\varphi\), does \(\varphi\) hold at the root of \(t\)?

\[(P'_3)\) Given \(t\) and an alternating parity tree automaton, does the automaton accept \(t\)?

\[(P'_4)\) Given \(t\) and an MSO formula \(\varphi\), does \(\varphi\) hold at the root of \(t\)?

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

Since the Modal Mu-Calculus Model Checking Problem for trees generated by (higher-order) recursion schemes is decidable [17], we obtain the following as an immediate consequence.

\footnote{This is inspired by an example in [5].}
Theorem 6 Problems $(P_1) - (P_4)$ are decidable with complexity $n$-EXPTIME complete.

Another remarkable consequence of the Equi-Expressivity Theorems 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 have been considered previously [20, 3, 14]. Conversely, as the Equi-Expressivity Theorems works in both directions, we note that a solution of Problem $(P_1)$ would give a new proof of the decidability of Problems $(P_2) - (P_4)$, and would give a new approach to problems on recursion schemes. Actually, the techniques of [20, 14] can be generalized to solve $n$-CPDS parity games without reference to [17]. Further it gives effective winning strategies for the winning player (which was not the case in [14] where the special case $n = 2$ was considered).

Theorem 7 Solving an $n$-CPDS parity game is $n$-EXPTIME complete and it can be achieved without reference to the decidability result in [17]. 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 $\omega$-regular winning condition, and is played on the $\varepsilon$-closure of the configuration graph of an $n$-CPDS graph. Consequently parity games on Cauca graphs [4, 3] are a special case of this problem.

As the class of $\varepsilon$-closure of configuration graphs of CPDS admits decidable mu-calculus theories, and as it contains the class of Cauca graphs (which enjoy decidable MSO theories [4], one can consider the MSO theories of configuration graphs of CPDS.

Theorem 8 (Undecidability) MSO theories of configuration graphs of CPDS are undecidable. Hence the class of $\varepsilon$-closure of configuration graphs of CPDS strictly contains the Cauca 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: $I_A(x, y) = x \cdot y \wedge x \rightarrow y$ and $I_B(x, y) = x \downarrow y$, with $C = T \cdot b \cdot a \cdot b \cdot 1^*$ and $R = 0 \cdot t \cdot a \cdot 1 \lor T \cdot 0 \cdot t \cdot a \cdot 1$. 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” on the right, which has an undecidable MSO theory.

Conclusions. 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 recursion 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 8) that order-$n$ CPDS are strictly more expressive than order-$n$ pushdown systems for generating graphs. As for further directions: (i) 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 Urzy- czyn tree [2] is definable by a 2-CPDA but not by any 2-PDA. (ii) Is there a finite way to represent the set of winning positions of an $n$-CPDS parity game (equivalently represent the set of vertices where a given modal mu-calculus formula holds)? (iii) Is there an $a$ la Cauca definition for the $\varepsilon$-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 $T$ from trees to graphs such that every $(n+1)$-CPDS graph is obtained by applying a $T$-transformation to some tree generated by an $n$-CPDA? Note that a $T$-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 we believe challenging) problem to obtain a translation that uses only first principles.
References


