2010 25th Annual IEEE Symposium on Logic in Computer Science

Recursion Schemes and Logical Reflection

Christopher H. Broadbent*, Arnaud Carayol†, C.-H. Luke Ong* and Olivier Serre‡
*Oxford University Computing Laboratory — {Christopher.Broadbent,Luke.Ong}@comlab.ox.ac.uk
†LIGM (Univ. Paris Est & CNRS) — Arnaud.Carayol@univ-mlv.fr
‡LIAFA (Univ. Paris 7 & CNRS) — Olivier.Serre@liafa.jussieu.fr

Abstract**—**Let $\mathcal{R}$ be a class of generators of node-labelled infinite trees, and $\mathcal{L}$ be a logical language for describing correctness properties of these trees. Given $R \in \mathcal{R}$ and $\varphi \in \mathcal{L}$, we say that R_φ is a *φ-reflection* of R just if (i) R and R_φ generate the same underlying tree, and (ii) suppose a node u of the tree $[\![R]\!]$ generated by R has label f, then the label of the node u of $[\![R_\varphi]\!]$ is $\underline{f}$ if u in $[\![R]\!]$ satisfies φ; it is f otherwise. Thus if $[\![R]\!]$ is the computation tree of a program R, we may regard R_φ as a transform of R that can internally observe its behaviour against a specification φ. We say that $\mathcal{R}$ is (constructively) *reflective* w.r.t. $\mathcal{L}$ just if there is an algorithm that transforms a given pair (R, φ) to R_φ. In this paper, we prove that higher-order recursion schemes are reflective w.r.t. both modal μ-calculus and monadic second order (MSO) logic. To obtain this result, we give the first characterisation of the winning regions of parity games over the transition graphs of collapsible pushdown automata (CPDA): they are regular sets defined by a new class of automata. (Order-n recursion schemes are equi-expressive with order-n CPDA for generating trees.) As a corollary, we show that these schemes are closed under the operation of MSO-interpretation followed by tree unfolding à la Caucal.

I. Introduction

An old model of computation, recursion schemes were originally designed as a canonical programming calculus for studying program transformation and control structures. In recent years, *higher-order recursion schemes* (HORS) have received much attention as a method of constructing rich and robust classes of possibly infinite ranked trees (or sets of such trees) with strong algorithmic properties. The interest was sparked by the discovery of Knapik et al. [12] that HORSs which satisfy a syntactic constraint called *safety* generate the same class of trees as higher-order pushdown automata. Remarkably these trees have decidable monadic second-order (MSO) theories, subsuming earlier well-known MSO decidability results for regular (or order-0) trees [17] and algebraic (or order-1) trees [7]. We now know [16] that the modal μ-calculus (local) model checking problem for trees generated by arbitrary order-n recursion schemes is n-EXPTIME complete (hence these trees have decidable MSO theories); further [9] these schemes are equi-expressive with a new variant class of higher-order pushdown automata, called *collapsible* pushdown automata (CPDA).

Let $\mathcal{T}$ be a class of finitely-presentable infinite structures (such as trees or graphs) and $\mathcal{L}$ be a logical language for describing correctness properties of these structures. The *global model checking problem* asks, given $t \in \mathcal{T}$ and $\varphi \in \mathcal{L}$, whether the set $\|t\|_\varphi$ of nodes defined by φ and t is finitely describable, and if so, whether it is decidable. Our first contribution is a solution of the modal μ-calculus global model checking problem for transition graphs of CPDA (the problem is equivalent to characterising winning regions of parity games played over the transition graphs of CPDA). To this end, we introduce a new kind of finite-state automata. Recall that an order-n *collapsible stack* is an order-n stack in which every symbol (except the bottom-of-stack) has a back pointer to some deeper stack of order less than n. For a fixed n, these (deterministic) automata take as input order-n collapsible stacks represented as well-bracketed sequences of symbols that have back pointers. When reading a symbol, the transition to a new state depends on, not just the current state, but also the state of the automaton when the symbol pointed to was read. These automata are closed under Boolean operations and have decidable acceptance and emptiness problems. We show that (Theorem 4) the winning regions of parity games played over the transition graphs of CPDA are regular i.e. recognizable by these (deterministic) automata. The proof is by induction on the order, and uses a sequence of game reductions that preserve regular sets.

An innovation of our work is a new approach to global model checking, by "internalising" the semantics $\|t\|_\varphi$. Let $\varphi \in \mathcal{L}$, and R be a HORS over Σ (i.e. the node labels of $[\![R]\!]$, the tree generated by R, are elements of the ranked alphabet Σ). We say that R_φ, which is a HORS over $\Sigma \cup \underline{\Sigma}$ (where $\underline{\Sigma}$ consists of a marked copy of each Σ-symbol), is a *φ-reflection*[1] of R just if R and R_φ generate the same underlying tree; further, suppose a node u of $[\![R]\!]$ has label f, then the label of the node u of $[\![R_\varphi]\!]$ is $\underline{f}$ if u in $[\![R]\!]$ satisfies φ, and it is f otherwise. Equivalently we can think of $[\![R_\varphi]\!]$ as the tree that is obtained from $[\![R]\!]$ by distinguishing the nodes that satisfy φ. Our second contribution is the result that HORS are (constructively) *reflective* w.r.t. the modal μ-calculus (Theorem 2). I.e. we give an algorithm that, given a modal μ-calculus formula φ, transforms a HORS to its φ-reflection. The proof relies on the *closure of CPDA under regular tests* (Theorem 3) i.e. we

Proofs are in the (downloadable) long version [2] of this paper.

[1] In programming languages, *reflection* is the process by which a computer program can observe and dynamically modify its own structure and behaviour.

1043-6871/10 $26.00 © 2010 IEEE
DOI 10.1109/LICS.2010.40

can endow the model of CPDA with the ability to test if the current configuration belongs to a given regular set without increasing its expressive power as tree generators.

The class of trees generated by HORSs is closed under two further logical operations. In a ranked tree, a node u may be represented by its unique path from the root, given as a finite word $path(u)$ over an appropriate alphabet. Let $\mathcal{B}$ be a finite-state word automaton over the same alphabet. We say that $R_{\mathcal{B}}$ is a *$\mathcal{B}$-reflection* of R just if R and $R_{\mathcal{B}}$ generate the same underlying tree; further if a node u of $\llbracket R \rrbracket$ has label f, then the label of node u of $\llbracket R_{\mathcal{B}} \rrbracket$ is $\underline{f}$ if $\mathcal{B}$ accepts $path(u)$, and it is f otherwise. We show that if a class $\mathcal{C}$ of tree generators is reflective w.r.t. modal μ-calculus, and w.r.t. regular paths (i.e. there is an algorithm that transforms a given pair $(\mathcal{B}, R)$ to $R_{\mathcal{B}}$), then it is also reflective w.r.t. MSO. We then obtain two pleasing consequences. First, trees that are generated by HORS are reflective w.r.t. MSO (Corollary 2). Secondly, if one starts with a tree t generated by an order-n recursion scheme and some MSO-interpretation I, then the unfolding of the graph $I(t)$ is isomorphic to a tree generated by an order-$(n+1)$ recursion scheme (Corollary 3). It follows that the class of trees generated by HORSs is closed under the operation of MSO-interpretation followed by tree unfolding à la Caucal.

Related work: Vardi and Piterman [21] studied the global model checking problem for regular trees and prefix-recognizable graphs using two-way alternating parity tree automata. Extending their results, Carayol et al. [4] showed that the winning regions of parity games played over the transition graphs of higher-order pushdown automata (i.e. without collapse) are regular. Recently, using game semantics, Broadbent and Ong [3] showed that for every order-n recursion scheme $\mathcal{S}$, the set of nodes in $\llbracket \mathcal{S} \rrbracket$ that are definable by a given modal μ-calculus formula is recognizable by an order-n (non-deterministic) collapsible pushdown word automaton. (Here we show in Theorem 2(i) that the nodes are recognizable by a *deterministic* CPDA.) In a different but related direction, Kartzow [11] showed that order-2 collapsible stacks can be encoded as trees in such a way that the set of stacks reachable from the initial configuration corresponds to a regular set of trees. (Since his notion of regularity on 2-stacks encompasses ours, it follows from our Theorem 4 that the winning regions of 2-CPDA parity games are regular sets of trees with Kartzow's encoding.)

Outline: In Section II we give the basic definitions. Section III introduces a notion of regular set of collapsible stacks, given by a new kind of finite-state automata. In Section IV, we characterise the winning regions of parity games played over the transition graphs of CPDA. Section V presents the reflection results.

II. Preliminaries

An *alphabet* A is a (possibly infinite) set of letters. In the sequel A^* denotes the set of *finite words* over A, and A^ω the set of *infinite words* over A. The empty word is written ε.

Higher-Order 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 $(A_1, \cdots, A_n, o)$. We define the *order* of a type by $ord(o) := 0$ and $ord(A \to B) := \max(ord(A) + 1, ord(B))$. Let Σ be a *ranked alphabet* i.e. each symbol f has an arity $ar(f) \geq 0$; we assume that f's type is the $(ar(f)+1)$-tuple $(o, \cdots, o, o)$. We further shall assume that each symbol $f \in \Sigma$ is assigned a finite set $\mathsf{Dir}(f)$ of $ar(f)$ *directions* (typically $\mathsf{Dir}(f) = \{ 1, \cdots, ar(f) \}$), 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^*. A *Σ-labelled tree* is a function $t : \mathsf{Dom}(t) \to \Sigma$ such that $\mathsf{Dom}(t)$ is a $\mathsf{Dir}(\Sigma)$-tree, and for every node $\alpha \in \mathsf{Dom}(t)$, the Σ-symbol $t(\alpha)$ has arity k if and only if α has exactly k children and the set of its children is $\{ \alpha\, i \mid i \in \mathsf{Dir}(t(\alpha)) \}$ i.e. t is a *ranked* tree.

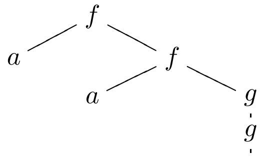
For each type A, we assume an infinite collection Var^A of variables of type A, and write Var to be the union of Var^A as A ranges over types; we write $t : A$ to mean that the expression t has type A. A (deterministic) *recursion scheme* is a tuple $\mathcal{S} = \langle \Sigma, \mathcal{N}, \mathcal{R}, I \rangle$ where Σ is a ranked alphabet of *terminals*; $\mathcal{N}$ is a set of typed *non-terminals*; $I \in \mathcal{N}$ is a distinguished *initial 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 is an *applicative term* of type o generated from elements of $\Sigma \cup \mathcal{N} \cup \{ \xi_1, \cdots, \xi_n \}$. We shall use lower-case roman letters for terminals (*e.g.* a, f, g), and upper-case roman letters for non-terminals (*e.g.* I, F, H). 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 $\mathcal{S}$, denoted $\llbracket \mathcal{S} \rrbracket$, is a possibly infinite applicative term, but viewed as a Σ-labelled tree, *constructed from the terminals in* Σ, that is obtained by rewriting using the rules of $\mathcal{S}$ *ad infinitum*, replacing formal by actual parameters each time, starting from the initial symbol I. See e.g. [9] for a formal definition.

Example 1. Let $\mathcal{S}$ be the order-2 recursion scheme with non-terminals $I : o$, $H : (o, o)$, $F : ((o, o), o)$; variables $x : o$, $\varphi : (o, o)$; terminals f, g, a of arity $2, 1, 0$ respectively;

and the following rewrite rules:

$$\left\{\begin{array}{rcl} I & \to & H\,a \\ H\,x & \to & F\,(f\,x) \\ F\,\varphi & \to & \varphi\,(\varphi\,(F\,g)) \end{array}\right.$$



The value tree $[\![\mathcal{S}]\!]$ (as shown above) is the Σ-labelled tree defined by the infinite term $f\,a\,(f\,a\,(g\,(g\,(g\,\cdots))))$.

Higher-Order Collapsible Stacks: 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 $[\,s_1 \cdots s_\ell\,]$) of order-n stacks such that every Γ-symbol $\gamma \neq \bot$ 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[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 $\bot_{k+1} = [\,\bot_k\,]$.

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

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

$$top_i\,\underbrace{[\,s_1 \cdots s_{\ell+1}\,]}_{s} \;=\; \left\{\begin{array}{ll} s_{\ell+1} & \text{if } i = ord(s) \\ top_i\,s_{\ell+1} & \text{if } i < ord(s) \end{array}\right.$$

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

$$\left\{\begin{array}{ll} [\,s_1 \cdots s_\ell\, push_1^{\alpha}\, s_{\ell+1}\,] & \text{if } ord(s) > 1 \\ [\,s_1 \cdots s_{\ell+1}\,\alpha\,] & \text{if } ord(s) = 1 \end{array}\right.$$

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 $pop_i\,\underbrace{[\,s_1 \cdots s_{\ell+1}\,]}_{s}$ by

$$\left\{\begin{array}{ll} [\,s_1 \cdots s_\ell\,] & \text{if } i = ord(s) \text{ and } \ell \geq 1 \\ [\,s_1 \cdots s_\ell\, pop_i s_{\ell+1}\,] & \text{if } i < ord(s) \end{array}\right.$$

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 $push_1^{\alpha,i}\,s$ we first attach a link from a fresh copy of α 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-j push* operation, $push_j$, simply takes a stack s and duplicates the top $(j-1)$-stack of s, preserving its link structure.

To avoid clutter, when displaying n-stacks in examples, we shall omit the bottom-of-stack symbols and 1-links (indeed by construction they can only point to the symbol directly below), writing e.g. $[\,[\,]\,[\,\alpha\,\gamma\,]\,]$ instead of $[\,[\,\bot\,]\,[\,\bot\;\alpha\;\gamma\,]\,]$.

Example 2. Take the 3-stack $s = [\,[\,[\,\alpha\,]\,]\;\;[\,[\,]\,[\,\alpha\,]\,]\,]$. We have

$$\begin{array}{rcl} push_1^{\beta,2}\,s & = & [\,[\,[\,\alpha\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,]\,]\,] \\ collapse\,(push_1^{\beta,2}\,s) & = & [\,[\,[\alpha\,]\,]\;[\,[\,]\,]\,] \\ \underbrace{push_1^{\gamma,3}(push_1^{\beta,2}\,s)}_{\theta} & = & [\,[\,[\,\alpha\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,\gamma\,]\,]\,]. \end{array}$$

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

$$[\,[\,[\,\alpha\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,\gamma\,]\,[\,\alpha\,\beta\,\gamma\,]\,]\,] \text{ and}$$

$$[\,[\,[\,\alpha\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,\gamma\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,\gamma\,]\,]\,].$$

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

Important Remark. Our definition of collapsible stacks allows *non-constructible* stacks such as

$$[\,[\,\bot\alpha\,]\,[\,\bot\beta\,]\,[\,\bot\beta\,]\,]$$

From now on, by an n-stack s, we mean a *constructible one i.e.* we assume there exists $\theta \in Op_n^*$ such that $s = \theta\bot_n$.

Collapsible Pushdown Automata: An *order-n (deterministic) collapsible pushdown automaton* (n-CPDA) is a 6-tuple $\langle\, A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F\,\rangle$ where A is an input alphabet and ε is a special symbol, Γ is a stack alphabet, Q is a finite set of states, q_0 is the initial state, $F \subseteq Q$ is the set of final states and $\delta : Q \times \Gamma \times (A \cup \{\varepsilon\}) \to Q \times Op_n$ is a transition (partial) function such that, for all $q \in Q$ and $\gamma \in \Gamma$, if $\delta(q, \gamma, \varepsilon)$ is defined then for all $a \in A$, $\delta(q, \gamma, a)$ is undefined (i.e. if some ε-transition can be taken, then no other transition is possible).

In the special case where $\delta(q, \gamma, \varepsilon)$ is undefined for all $q \in Q$ and $\gamma \in \Gamma$ we refer to $\mathcal{A}$ as an ε-free n-CPDA

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

and we omit ε in the definition of $\mathcal{A}$ *i.e.* we denote it as $\mathcal{A} = \langle A, \Gamma, Q, \delta, q_0, F \rangle$.

Configurations of an n-CPDA are pairs of the form (q, s) where $q \in Q$ and s is an n-stack over Γ; the *initial configuration* is $(q_0, \bot_n)$ and *final configurations* are those whose control state belongs to F.

An n-CPDA $\mathcal{A} = \langle A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F \rangle$ naturally defines an $(A \cup \{\varepsilon\})$-labelled transition graph $G(\mathcal{A}) := (V, E)$ whose vertices V are the configurations of $\mathcal{A}$ and whose edge relation E is given by: $((q, s), a, (q', s')) \in E$ iff $\delta(q, top_1 s, a) = (q', op)$ and $s' = op(s)$. Such a graph is called an *n-CPDA graph*.

In this paper we will use n-CPDA for three different purposes: as words acceptors, as generators for infinite trees and as generators of the graph underlying a parity game.

Using an n-CPDA as a Words Acceptor: A order-n CPDA $\mathcal{A} = \langle A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F \rangle$ accepts the set of words $w \in A^*$ labeling a run from the initial configuration to a final configuration (interpreting ε as a silent move). We write $L(\mathcal{A})$ for the accepted language.

Using an n-CPDA as an Infinite Tree Generator: Fix an n-CPDA $\mathcal{A} = \langle A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F \rangle$. Take the *$\varepsilon$-closure* $G_\varepsilon(\mathcal{A})$ of $G(\mathcal{A})$ defined as follows: first add an a-labelled edge from v_1 to v_2 whenever there is a path from v_1 to v_2 labelled by a word that matches $a\varepsilon^*$, and there is no outgoing ε-labelled from v_2; then remove any vertex (in the path) that is the source of an ε-labelled edge. Owing to the restriction we imposed on δ, the resulting graph is deterministic and ε-free.

In $G(\mathcal{A})$ there exists a unique configuration v_0 which is reachable from the initial configuration by a (possibly empty) sequence of ε-labelled edges, and the source of a non-ε-labelled edge. Trivially, v_0 is a vertex of $G_\varepsilon(\mathcal{A})$. Now, let T be the tree obtained by unfolding $G_\varepsilon(\mathcal{A})$ from v_0. Then T is deterministic.

Finally, in order to define a Σ-labelled tree t for a ranked alphabet Σ, it suffices to identify a total function $\rho : Q \to \Sigma$ such that for all $q \in Q$ and $\gamma \in \Gamma$, $\{a \mid (q, \gamma, a) \in \mathsf{Dom}(\delta)\} = \mathsf{Dir}(\rho(q))$, and then to define t by $t(u) := \rho(q_u)$ for every node $u \in \mathsf{Dom}(T)$, where q_u is the state of the last configuration of u.

In [9] (a version of) the following equi-expressivity result was proved.

Theorem 1. *(i) Let $\mathcal{S}$ be an order-n recursion scheme over Σ and let t be its value tree. Then there is an order-n CPDA $\mathcal{A} = \langle A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F \rangle$, and $\rho : Q \to \Sigma$ such that t is the tree generated by $\mathcal{A}$ and ρ.*

(ii) Let $\mathcal{A} = \langle A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F \rangle$ be an order-n CPDA, and let t be the Σ-labelled tree generated by $\mathcal{A}$ and a given map $\rho : Q \to \Sigma$. Then there is an order-n recursion scheme over Σ whose value tree is t.

Moreover the inter-translations between schemes and CPDA are polytime computable.

Using an n-CPDA to Define a Parity Game: We start by recalling the definition of parity game. Let $G = (V, E \subseteq V \times V)$ be a graph. Let $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ be a partition of V between two players, Éloïse and Abelard. A *game graph* is such a tuple $\mathcal{G} = (G, V_{\mathbf{E}}, V_{\mathbf{A}})$. A colouring function ρ is a mapping $\rho : V \to C \subset \mathbb{N}$ where C is a finite set of colours. An *infinite two-player parity game* on a game graph $\mathcal{G}$ is a pair $\mathbb{G} = (\mathcal{G}, \rho)$.

Éloïse and Abelard play in $\mathbb{G}$ by moving a token between vertices. A *play* from some initial vertex v_0 proceeds as follows: the player owning v_0 moves the token to a vertex v_1 such that $(v_0, v_1) \in E$. Then the player owning v_1 chooses a successor v_2 and so on. If at some point one of the players cannot move, she/he loses the play. Otherwise, the play is an infinite word $v_0 v_1 v_2 \cdots \in V^\omega$ and is won by Éloïse just in case $\liminf(\rho(v_i))_{i \geq 0}$ is even. A *partial play* is just a prefix of a play.

A strategy for Éloïse is a function assigning, to every partial play ending in some vertex $v \in V_{\mathbf{E}}$, a vertex v' such that $(v, v') \in E$. Éloïse *respects a strategy* Φ during a play $\Lambda = v_0 v_1 v_2 \cdots$ if $v_{i+1} = \Phi(v_0 \cdots v_i)$, for all $i \geq 0$ such that $v_i \in V_{\mathbf{E}}$. A strategy Φ for Éloïse is *winning* from a position $v \in V$ if she wins every play that starts from v and respects Φ. Finally, a vertex $v \in V$ is *winning* for Éloïse if she has a winning strategy from v, and the winning region for Éloïse consists of all winning vertices for her. Symmetrically, one defines the corresponding notions for Abelard. It follows from Martin's Theorem [14] that, from every position, either Éloïse or Abelard has a winning strategy.

Now let $\mathcal{A} = \langle A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F \rangle$ be an order-n CPDA and let (V, E) be the graph obtained from $G(A)$ by removing edge-labels. Let $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ be a partition of Q and let $\rho : Q \to C \subset \mathbb{N}$ be a colouring function (over states). 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 $\rho : V \to C$ where a vertex is assigned the colour of its control state. The structure $\mathcal{G} = (G(A), V_{\mathbf{E}}, V_{\mathbf{A}})$ defines a game graph and the pair $\mathbb{G} = (\mathcal{G}, \rho)$ defines a parity game (that we call a *n-CPDA parity game*).

The Global Model-Checking Problem: Fix a Σ-labelled tree t given by a recursion scheme or by a CPDA, and a logical formula φ (*e.g.* a μ-calculus formula, or an MSO formula with a single free first-order variable). We denote by $\|t\|_\varphi$ the set of nodes of t described by φ.

The *local model checking problem* asks whether $u \in \|t\|_\varphi$ for a given node u. Decidability of this problem was first proved in [16]. The *global model checking problem* asks for a *finite* description of the set $\|t\|_\varphi$, if there is one. As $\|t\|_\varphi$ is in general an infinite set, there are several *non-equivalent* ways to represent it finitely. However there are two natural approaches.

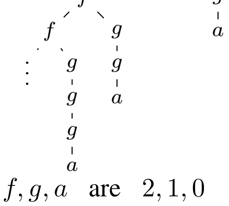
- *Exogeneous:* Given a Σ-labelled tree t and a formula φ, output a description by means of a word acceptor device recognising $\|t\|_\varphi \subseteq \mathsf{Dir}(\Sigma)^*$.

- *Endogeneous:* Given a Σ-labelled tree t and a formula φ, output a finite description of the $(\Sigma \cup \underline{\Sigma})$-labelled tree t_φ — where $\underline{\Sigma} = \{\underline{\sigma} \mid \sigma \in \Sigma\}$ is a marked copy of Σ — such that $\mathsf{Dom}(t_\varphi) = \mathsf{Dom}(t)$, and $t_\varphi(u) = \underline{t(u)}$ if $u \in \|t\|_\varphi$ and $t_\varphi(u) = t(u)$ otherwise.

In case the Σ-labelled tree t is generated by an order-n recursion scheme, it is natural to consider order-n CPDA both as words acceptors for $\|t\|_\varphi$ (in the exogeneous approach) and as tree genarator for t_φ (in the endogeneous approach). In the latter case, order-n schemes and CPDA can be used interchangeably.

Example 3. Let $\mathcal{S}$ be the order-2 recursion scheme with non-terminals $I : o$, $F : ((o,o),o,o)$ (and variables and terminals as in Example 1) and the following rewrite rules:

$$\begin{cases} I & \rightarrow \quad F\,g\,(ga) \\ F\,\varphi\,x & \rightarrow \quad f\,(F\,\varphi\,(\varphi\,x))\,x \end{cases}$$



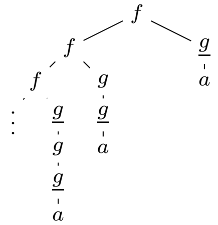
where the arities of the terminals f, g, a are $2, 1, 0$ respectively. The value tree $t = [\![\mathcal{S}]\!]$ is the Σ-labelled tree depicted above.

Let $\varphi = p_g \wedge \mu X.(\diamond_1 p_a \vee \diamond_1 \diamond_1 X)$, where p_g (resp. p_a) is a propositional variable asserting that the current node is labelled by g (resp. a), be the μ-calculus formula[3] defining the nodes which are labelled by g such that the length of the (unique) path to an a-labelled node is odd.

An exogeneous approach to the global model checking problem is to output a 2-CPDA accepting the set $\|t\|_\varphi = \{1^n 2 1^k \mid n{+}k \text{ is odd}\}$, which in this special case is regular.

An endogeneous approach to this problem is to output the following recursion scheme:

$$\begin{cases} I & \rightarrow \quad H\,\underline{g}\,a \\ H\,z & \rightarrow \quad f\,(\underline{H}\,g\,z)\,z \\ \underline{H}\,z & \rightarrow \quad f\,(H\,\underline{g}\,z)\,z \end{cases}$$



with non-terminals $I : o$, $H : (o,o)$; and a variable $z : o$. The value tree of this new scheme is depicted on the right.

Our first contribution of the paper addresses the global model checking problem for trees generated by recursion scheme. (The theorem will be proved in Section V).

Theorem 2 (μ-Calculus Reflection). *Let t be a Σ-labelled tree generated by an order-n recursion scheme $\mathcal{S}$ and φ be a μ-calculus formula.*

(i) There is an algorithm that transforms $(\mathcal{S}, \varphi)$ to an order-n CPDA $\mathcal{A}$ such that $L(\mathcal{A}) = \|t\|_\varphi$.

(ii) There is an algorithm that transforms $(\mathcal{S}, \varphi)$ to an order-n recursion scheme that generates t_φ.

Remark 1. Note that *(ii)* implies *(i)*. To see why this is so, assume that we can construct an order-n recursion scheme generating t_φ. Thanks to Theorem 1, we can construct in polynomial time an order-n CPDA $\mathcal{A}$ which, together with a mapping $\rho : Q \mapsto \Sigma \cup \underline{\Sigma}$, generates t_φ. Taking $\{q \in Q \mid \rho(q) \in \underline{\Sigma}\}$ as a set of final states, $\mathcal{A}$ accepts $\|t\|_\varphi$.

Winning Regions: The key ingredient of the proof of Theorem 2 is a precise characterisation of the winning regions of parity games defined by CPDA. This exploits the close connection between μ-calculus and parity games [8]. Hence, an important part of this article is devoted to an effective characterisation of the winning regions of n-CPDA parity games. Section III introduces a new class of automata accepting sets of configurations of n-CPDA, and in Section IV we prove that for any n-CPDA parity game one can effectively represent the winning regions by such an automaton.

III. Regular Sets of Collapsible Stacks

We start by introducing a class of automata with a finite state-set that can be used to recognize sets of collapsible stacks. Let s be an order-n collapsible stack. We first associate with $s = s_1, \cdots, s_\ell$ a well-bracketed word of depth n, $\widetilde{s} \in (\Sigma \cup \{\,[\,,\,]\,\})^*$:

$$\widetilde{s} \;:=\; \begin{cases} [\,\widetilde{s_1} \cdots \widetilde{s_\ell}\,] & \text{if } n \geq 1 \\ s & \text{if } n = 0 \ (\textit{i.e. } s \in \Sigma) \end{cases}$$

In order to reflect the link structure, we define a partial function $\mathit{target}(s) : \{1, \cdots, |\widetilde{s}|\} \rightarrow \{1, \cdots, |\widetilde{s}|\}$ that assigns to every position in $\{1, \cdots, |\widetilde{s}|\}$ the index of the end of the stack targeted by the corresponding link (if exists; indeed this is undefined for $\bot$, [and]). Thus with s is associated the pair $\langle\, \widetilde{s}, \mathit{target}(s)\,\rangle$; and with a set S of stacks is associated the set $\widetilde{S} = \{\langle\, \widetilde{s}, \mathit{target}(s)\,\rangle \mid s \in S\}$.

Example 4. Let $s = [\,[\,[\bot\,\alpha\,]\,]\;[\,[\bot\,]\,[\,\bot\,a\,\beta\,\gamma\,]\,]\,]$. Then

$\widetilde{s} = [\,[\,[\,\bot\,\alpha\,]\,]\;[\,[\bot\,]\,[\,\bot\,\alpha\,\beta\,\gamma\,]\,]\,]$ and $\mathit{target}(5) = 4$, $\mathit{target}(14) = 13$, $\mathit{target}(15) = 11$ and $\mathit{target}(16) = 7$.

We consider *deterministic* finite automata working on such representations of collapsible stacks. The automaton reads the word $\widetilde{s}$ from left to right. On reading a letter that does not have a link (i.e. target is undefined on its index) the automaton updates its state according to the current state and the letter; on reading a letter that has a link, the automaton updates its state according to the current state, the letter and the state it was in after processing the targeted position. A run is accepting if it ends in a final state. One can think

[3] We refer the reader to [1] for syntax and semantics of μ-calculus.

of these automata as a deterministic version of Stirling's *dependency tree automata* [19] restricted to words.

Formally, an automaton is a tuple $\langle Q, A, q_{in}, F, \delta \rangle$ where Q is a finite set of states, A is a finite input alphabet, $q_{in} \in Q$ is the initial state, $F \subseteq Q$ is a set of final states and $\delta : (Q \times A) \cup (Q \times A \times Q) \to Q$ is a transition function. With a pair $\langle u, \tau \rangle$ where $u = a_1 \cdots a_n \in A^*$ and τ is a partial map from $\{1, \cdots n\} \to \{1, \cdots n\}$, we associate a unique run $r = r_0 \cdots r_n$ as follows:
- $r_0 = q_{in}$;
- for all $0 \leq i < n$, $r_{i+1} = \delta(r_i, a_{i+1})$ if $i + 1 \notin Dom(\tau)$;
- for all $0 \leq i < n$, $r_{i+1} = \delta(r_i, a_{i+1}, r_{\tau(i+1)})$ if $i + 1 \in Dom(\tau)$.

The run is *accepting* just if $r_n \in F$, and the pair (u, τ) is *accepted* just if the associated run is accepting.

To recognize configurations instead of stacks, we use the same machinery but now add the control state at the end of the coding of the stack. We code a configuration (p, s) as the pair $\langle \widetilde{s} \cdot p, target(s) \rangle$ (hence the input alphabet of the automaton also contains a copy of the control state of the corresponding CPDA).

Finally, we say that a set K of n-stacks over alphabet Γ is *regular* just if there is an automaton $\mathcal{B}$ such that for every n-stack s over Γ, $\mathcal{B}$ accepts $\langle \widetilde{s}, target(s) \rangle$ iff $s \in K$. Regular sets of configurations are defined in the same way.

Remark 2. Non-deterministic automata are strictly more powerful than deterministic automata. Let L be the set of words with links $\langle \widetilde{s}, target(s) \rangle$ such that $target(s)$ is injective: $\forall x, y,\ target(s)(x) = target(s)(y) \Rightarrow x = y$. Then L is not accepted by a deterministic automaton but its complement is accepted by a non-deterministic automaton. Since L is also not accepted by a non-deterministic automaton, the model of non-deterministic automaton is not closed under complement.

Closure Properties: Regular sets of stacks (resp. configurations) form an effective Boolean algebra.

Property 1. *Let H, K be regular sets of n-stacks over an alphabet Γ. Then $L \cup K$, $L \cap K$ and $Stacks(\Gamma) \setminus L$ are also regular (here $Stacks(\Gamma)$ denotes the set of all stacks over Γ). The same holds for regular sets of configurations.*

We can endow the model of CPDA with the ability to test if the current configuration belongs to a given regular set without increasing its expressive power as tree generators.

Theorem 3. *Given an order-n CPDA $\mathcal{A}$ with a state-set Q and an automaton $\mathcal{B}$ (that takes $\mathcal{A}$-configurations as input), there exist an order-n CPDA $\mathcal{A}[\mathcal{B}]$ with a state-set Q', a subset $F \subseteq Q'$ and a mapping $\chi : Q' \to Q$ such that:*

(i) restricted to the reachable configurations, the respective ε-closures of $G(\mathcal{A})$ and $G(\mathcal{A}[\mathcal{B}])$ are isomorphic

(ii) for every configuration (q, s) of $\mathcal{A}[\mathcal{B}]$, the corresponding configuration of $\mathcal{A}$ has state $\chi(q)$ and belongs to $L(\mathcal{B})$ if and only if $q \in F$.

Proof (Sketch): Fix an order-n CPDA $\mathcal{A}$ and an automaton $\mathcal{B}$. We wish to construct a new order-n CPDA $\mathcal{A}[\mathcal{B}]$ that simulates $\mathcal{A}$ and in the meantime computes the state reached by $\mathcal{B}$ after processing the current stack. To this end, we associate with every stack a finite amount of information describing the behaviour of $\mathcal{B}$ when reading it.

Let Q be the state set of $\mathcal{B}$. Let S be an order-n stack and let s_k be its top k-stack. If s_k was simply a stack without links, it could be described, from the point of view of $\mathcal{B}$, by the mapping τ from Q to Q such that if $\mathcal{B}$ starts reading s_k in state q then it finishes reading it in state $\tau(q)$. However, if one simply extracts s_k from S, there may be "dangling links" of order greater than k. As the number of these links is unbounded, it is impossible to specify individually the Q-state that should be attached to the target of each of these links. Our idea is to associate with s_k a mapping τ_k^S which abstracts the behaviour of $\mathcal{B}$ on s_k but *in the context* of S (i.e. the information will only be pertinent when s_k is the top k-stack of S).

Thus s_k gives rise to a mapping $\tau_k^S : Q^{n-k} \to (Q \to Q)$ that, given a tuple $(q_n, \cdots, q_{k+1})$, defines a transformation from Q to Q. We use states $q_n, \cdots, q_{k+1}$ to define the values of the states attached to the respective targets of the links (of order $n, \cdots, k+1$ respectively) in s_k: for n-links, we consider the run induced by reading S (we stop when s_k is reached) starting from q_n (this gives the value for the respective targets of the n-links), for $(n-1)$-links, we consider the run induced by reading $top_n(S)$ (we stop when s_k is reached) starting from $q_{n-1}, \ldots$; and for $(k+1)$-links, we consider the run induced by reading $top_{k+2}(S)$ (again we stop when s_k is reached) starting from q_{k+1}.

At any point in the computation of the CPDA $\mathcal{A}[\mathcal{B}]$ where a stack S of $\mathcal{A}$ is simulated, the top_1 symbol is a pair, consisting of the stack symbol $top_1(S)$, and an n-tuple $(\tau_{n-1}, \cdots, \tau_0)$ where τ_i is equal to $\tau_i^{pop_i S}$ – for technical reasons, we do not care for the top $(i-1)$-stack of S when defining τ_i).

The result is finally obtained by first showing that the state of $\mathcal{B}$, after reading the whole stack, can be recovered from the τ_i, and then proving that the values of the τ_i can be maintained (for the top elements only) when simulating any stack action.

■

Emptiness: The closure under regular tests implies decidability of emptiness of automata with respect to constructible stacks.

Proposition 1. *For a fixed $n \geq 2$, the question of whether there is an order-n* constructible *stack that is accepted by a given automaton is decidable in $(n-1)$-*EXPTIME.

Proof (Sketch): Consider the stateless n-CPDA that allows us to construct all possible stacks. Now take its closure $\mathcal{A}'$ under regular test with respect to $\mathcal{B}$ and use $\mathcal{A}'$ as a words acceptor (final states are the set F as given in Theorem 3). Then, the given automaton $\mathcal{B}$ accepts at least one constructible stack iff $L(\mathcal{A}') \neq \emptyset$. As the latter is decidable in $(n-1)$-EXPTIME, we get the expected result [9]. ■

It is to be noted that if we no longer require the accepted stack to be constructible the problem becomes less intractable.

Proposition 2. *For a fixed $n \geq 2$, the question of whether there is an order-n* possibly non-constructible *stack that is accepted by a given automaton is NP-complete.*

Proof (Sketch): Upper-bound is by a small model property argument. Lower-bound is by reducing 3-SAT. ■

IV. WINNING REGIONS OF CPDA GAMES

The main result of this section is a characterisation of winning regions by regular sets.

Theorem 4. *Let $\mathbb{G}$ be an n-CPDA parity game. Then the winning region for Éloïse (resp. for Abelard) is a regular set which can be effectively constructed.*

Proof (Sketch): As the complete proof of Theorem 4 requires a lot of machinery, we will only focus on the key steps. Let us also stress that this proof borrows several ideas [9], [4] but also extend in a non trivial way their results (decidability of CPDA games of [9] and characterisation of winning region of HOPD games — *i.e.* games generated by CPDA without links — of [4]). The full version [2] provides a self contained proof of the result.

The proof is by induction on the order, and the induction step can be divided in three sub-steps (for order-1, the result is a classical one [18]). Assume one starts with an n-CPDA parity game $\mathbb{G}$ (using colours $\{0, \ldots, d\}$) generated by some n-CPDA $\mathcal{A}$. One does the following steps:

1) One builds a new n-CPDA $\mathcal{A}_{\mathrm{rk}}$ that mimics $\mathcal{A}$ and that is rank-aware in the following sense. Take an n-CPDA and assume that states are coloured by integers. Consider a finite run λ of $\mathcal{A}$ and assume that the top_1-element in the last configuration of λ has an n-link: then the *link rank* is defined as the smallest colour encountered since the creation of the original copy of the current n-link. An n-CPDA is *rank-aware* just if there is some function ρ from its stack alphabet into the set of colours such that at any point in a run of the automaton, if the top_1-element has an n-link, then applying ρ to it gives the link rank. Then from $\mathcal{A}_{\mathrm{rk}}$ one naturally gets a new parity game $\mathbb{G}_{\mathrm{rk}}$ and a transformation ν_1 from any vertex v in $\mathbb{G}$ to a vertex $\nu_1(v)$ in $\mathbb{G}_{\mathrm{rk}}$ such that Éloïse wins in $\mathbb{G}$ from v iff she wins from $\nu_1(v)$ in $\mathbb{G}_{\mathrm{rk}}$. One also proves that regular sets of configurations are preserved by ν_1^{-1}: hence it suffices to prove that winning regions are regular for games generated by rank-aware n-CPDA.

2) We now construct a new n-CPDA game that makes no use of n-links. This game mimics $\mathbb{G}_{\mathrm{rk}}$ except that whenever a player wants to perform a $push_1^{\gamma,n}$ action on the stack, this is replaced by the following negotiation between the players:

- Éloïse has to provide a vector $\overrightarrow{R} = (R_0, \cdots R_d) \in (2^{Q_{\mathrm{rk}}})^{d+1}$ — here Q_{rk} are the control states of $\mathcal{A}_{\mathrm{rk}}$ — whose intended meaning is the following: she claims that she has a strategy such that if the newly created link (or a copy of it) is eventually used by some collapse then it leads to a state in R_i where i is the smallest colour visited since the original copy of the link was created.
- Abelard has two choices. He can agree with Éloïse's claim, pick a state q in some R_i and perform a pop_n action whilst going to state q (through an intermediate dummy vertex coloured by i): this is the case where Abelard wants to simulate a collapse involving the link. Alternatively Abelard can decide to push the symbol $(\gamma, \overrightarrow{R})$ without appending a link to it.

Later in the play, if the top_1-element is of the form $(\gamma, \overrightarrow{R})$, and if the player controlling the current configuration wants to simulate a move to state q that collapses the stack, then this move is replaced by one that goes to a dead end vertex. This is deemed winning for Éloïse iff $q \in R_i$ where i is the *link rank* found on the current top_1-element, which corresponds to the smallest colour visited since the original copy of symbol $(\gamma, \overrightarrow{R})$ was pushed onto the stack (recall that $\mathcal{A}_{\mathrm{rk}}$ is rank-aware). The intuitive idea is that, when simulating a collapse (involving an order-n link), Éloïse wins iff her initial claim on the possible reachable states by following the link was correct. Otherwise she loses.

Call $\mathbb{G}_{\mathrm{lf}}$ (lf for n-link free) this new game. Then one can define a transformation ν_2 from any vertex v in $\mathbb{G}_{\mathrm{rk}}$ to a vertex $\nu_2(v)$ in $\mathbb{G}_{\mathrm{lf}}$ such that Éloïse wins in $\mathbb{G}_{\mathrm{rk}}$ from v iff she wins from $\nu_2(v)$ in $\mathbb{G}_{\mathrm{lf}}$. One also proves that regular sets of configurations are preserved by ν_2^{-1}: hence it suffices to prove that winning regions are regular for order-n games that have no n-links.

Let us briefly explain how ν_2 works as it motivated our definition of automata recognising collapsible stackss. ν_2 takes a collapsible stacks and transforms it into a stack where every symbol γ with an n-link is replaced by some symbol $(\gamma, \overrightarrow{R})$ without any link. Hence, one needs to explain how $\overrightarrow{R}$ is defined. Consider the stack obtained by removing every symbol above γ and by collapsing (hence the new top_n stack is the targeted one), and let R be the set of states such that Éloïse wins in $\mathbb{G}_{\mathrm{rk}}$ from this state with this new stack content: then $\overrightarrow{R} = (R, \cdots, R)$. An automaton deciding whether a configuration in $\mathbb{G}_{\mathrm{lf}}$ is winning will process the stack and encode on its control state a subset of states (of the CPDA) that are the winning ones at every position of the stack. To decide if a configuration is winning in $\mathbb{G}_{\mathrm{rk}}$ one computes on-the-fly its image under ν_2 and simulates the previous automaton. This image can be inferred as the only information needed (*i.e.* R) is precisely what is computed by the automaton and the information is available following the n-links in our model of automata.

Example 5. Assume we are playing a two-colour parity game. Let

$$s = [\,[\,[\,\alpha\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,\gamma\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,\gamma\,]\,]\,],$$

$R = \{r \mid (r, [[[\alpha]]])$ is winning for Éloïse in $\mathbb{G}_{\mathrm{rk}}\}$ and $\overrightarrow{R} = (R, R)$. Then

$$\nu_2(s) = [\,[\,[\,\alpha\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,(\gamma, \overrightarrow{R})\,]\,]\;[\,[\,]\,[\,\alpha\,\beta\,(\gamma, \overrightarrow{R})\,]\,]\,].$$

3) The last step is to construct an $(n-1)$-CPDA game from which one can reconstruct the winning region in $\mathbb{G}_{\mathrm{lf}}$. This can be done using the concept of abstract pushdown games developed in [4] and noting that order-n games that have no n-links are a special class of such games. Then using induction hypothesis and extending the results in [4] one concludes that the winning regions are regular in $\mathbb{G}_{\mathrm{lf}}$. ■

Since the class of n-CPDA graphs is closed under Cartesian product with finite structures, Theorem 4 directly leads to a characterisation of μ-calculus definable sets over those graphs.

Corollary 1. *The μ-calculus definable sets over CPDA-graphs are regular.*

Proof (Sketch): Take a CPDA-graph G and a μ-calculus formula φ. From φ, it is well known (see for instance [1]) how to construct a finite rooted graph G_φ and a parity game $\mathbb{G}$ over the synchronized product of G and G_φ such that, for any vertex v in G the formula φ holds at v iff Éloïse wins in $\mathbb{G}$ from (v, r) where r is the root of G_φ. As the class of CPDA graphs is closed under Cartesian product with finite graphs, $\mathbb{G}$ is a CPDA parity game. Hence to decide whether φ holds in a configuration v it suffices to simulate on (v, r) the automaton (constructed in Theorem 4) accepting the (regular) winning region for Éloïse in $\mathbb{G}$. This easily implies that the set of vertices where φ holds in G is itself regular. ■

V. Modal μ-Calculus and MSO Reflections

Our first task is to prove Theorem 2.

Proof of Theorem 2: We concentrate on *(ii)* as it implies *(i)* (cf. Remark 1). Fix an order-n recursion scheme $\mathcal{S} = \langle\, \Sigma, \mathcal{N}, \mathcal{R}, I\,\rangle$ and let t be its value tree. Let φ be a μ-calculus formula. Using Theorem 1, we can construct an n-CPDA $\mathcal{A} = \langle\, A \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F\,\rangle$ and a mapping $\rho : Q \to \Sigma$ such that t is the tree generated by $\mathcal{A}$ and ρ.

Let U be the unfolding of $G(\mathcal{A})$ from its initial configuration and U_ε be the ε-closure of U. A node π of U_ε is a path in $G(\mathcal{A})$ starting from the initial configuration of $\mathcal{A}$ and ending in some configuration (q_π, s_π). By definition, there exists an ismorphism h from U_ε to $\mathsf{Dom}(t)$ such that for all nodes π of U_ε, $t(h(\pi)) = \rho(q_\pi)$.

Assume that for every state q of $\mathcal{A}$, we have a predicate p_q that holds at a node π of U_ε iff $q = q_\pi$. Then the formula φ can be translated to a formula φ' on U_ε (i.e. $h(\|U_\varepsilon\|_{\varphi'}) = \|t\|_\varphi$) as follows: for each $a \in \Sigma$, replace every occurrence of the predicate p_a in φ by the disjunction $\bigvee_{q \in Q, \rho(q) = a} p_q$.

In turn φ' can be translated to a formula φ_ε on U (i.e. $h(\|U_\varepsilon\|_{\varphi'}) = \|U\|_{\varphi_\varepsilon}$). Take the formula φ_ε obtained by replacing in φ every sub-formula of the form $\diamond_a \psi$ by $\diamond_a(\mu X.[(\psi \wedge \neg(\diamond_\varepsilon\, true)) \vee \diamond_\varepsilon X])$, *i.e.* replace the assertion "take an a-edge to a vertex where ψ holds" by the assertion "take an a-edge to some vertex from which one can reach, via a finite sequence of ε-edges, a vertex where ψ holds and which is not the source vertex of an ε-labelled edge".

As unfolding preserves μ-calculus definable properties, we have that $\pi \in \|U_\varepsilon\|_{\varphi'}$ iff $\pi \in \|U\|_{\varphi_\varepsilon}$ iff $(q_\pi, s_\pi) \in \|G(\mathcal{A})\|_{\varphi_\varepsilon}$. Using Corollary 1 we know that the set of configurations of $G(\mathcal{A})$ that satisfy φ_ε is regular, *i.e.* $\|G(\mathcal{A})\|_{\varphi_\varepsilon}$ is accepted by some automaton $\mathcal{B}$.

Using Theorem 3, we construct a new n-CPDA $\mathcal{A}'$ with a set Q' of state together with a set $F \subseteq Q'$ and a mapping $\chi : Q' \to Q$ such that:

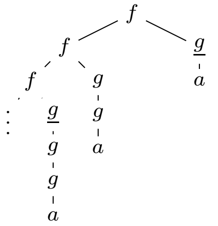
- restricted to the reachable configurations, the respective ε-closures of $G(\mathcal{A})$ and $G(\mathcal{A}')$ are isomorphic
- for any configuration (q, s) of $\mathcal{A}'$, the corresponding configuration of $\mathcal{A}$ has state $\chi(q)$ and belongs to $L(\mathcal{B})$ if and only if $q \in F$.

It follows at once that the tree t_φ is defined by $\mathcal{A}'$ with the mapping ρ' defined as follows: for all $q \in Q'$, $\rho'(q) := \rho(q)$ if $q \notin F$, and $\rho'(q) := \underline{\rho(q)}$ otherwise. □

Remark 3. *There are two natural questions concerning complexity. The first one concerns the algorithm in Theorem 2: it is n time exponential in both the size of the scheme and the size of the formula. This is because we need to solve an order-n CPDA parity game built by taking a product of an order-n CPDA equi-expressive with $\mathcal{S}$ (thanks to Theorem 1 its size is polynomial in the one of $\mathcal{S}$) with a finite transition system of polynomial size in that of φ. The second issue concerning complexity is how the size of the new scheme (obtained in the second point of Theorem 2) relates to that of $\mathcal{S}$ and φ. For similar reasons, it is n time exponential in the size of $\mathcal{S}$ and φ.*

It is natural to ask if trees generated by HORS are reflective w.r.t. MSO. (Modal μ-calculus and MSO are equivalent for expressing properties of a deterministic tree at the *root*, but not other nodes; see *e.g.* [10]. Indeed one would need backwards modalities to express all of MSO in μ-calculus.) Consider the following property (definable in MSO but not in μ-calculus) on nodes u of a tree: "u is the right son of an f-labelled node, and there is a path from u to an a-labelled node which contains an odd occurrences of g-labelled nodes". Returning to the scheme of Example 1 one would expect the following answer to the global model-checking problem for the corresponding MSO formula:

$$\left\{\begin{array}{rcl} I & \rightarrow & \underline{F}\, g\, a \\ \underline{F}\, \varphi\, x & \rightarrow & f\,(F\, g\,(\varphi\, x))\,(\underline{g}\, x) \\ F\, \varphi\, x & \rightarrow & f\,(\underline{F}\, g\,(\varphi\, x))\,(g\, x) \end{array}\right.$$



Corollary 2 (MSO Reflection). *Let t be a Σ-labelled tree generated by an order-n recursion scheme $\mathcal{S}$, and $\varphi(x)$ be an MSO-formula.*

(i) There is an algorithm that transforms $(\mathcal{S}, \varphi)$ to an order-n CPDA $\mathcal{A}$ such that $L(\mathcal{A}) = \|t\|_\varphi$.

(ii) There is an algorithm that transforms $(\mathcal{S}, \varphi)$ to an order-n recursion scheme that generates t_φ.

Proof (Sketch): As before, we concentrate on *(ii)* which implies *(i)*. Using the well-known equivalence between MSO and automata (see [20]), the question of whether a node u of t satisfies $\varphi(x)$ can be reduced to whether a given parity tree automaton $\mathcal{B}$ accepts the tree t_u that is obtained from t by marking the node u (and no other node).

In order to construct t_φ, we first annotate t with information on the behaviour of $\mathcal{B}$ on the subtrees of t. We mark t by μ-calculus definable sets to obtain an enriched tree denoted $\bar{t}$. With each pair $(q, d) \in Q \times \mathsf{Dir}(\Sigma)$, we associate a formula $\psi_{q,d}$ such that $t, u \models \psi_{q,d}$ iff the d-son of u exists and $\mathcal{B}$ has an accepting run on $t[u\, d]$ starting from q (here $t[v]$ is the subtree of t rooted at v). By Theorem 2, $\bar{t}$ can be generated by an n-CPDA.

Let Σ' be the alphabet of $\bar{t}$. For every node u, one can decide, using the annotations on $\bar{t}$ and considering only the path from the root to u, whether $\mathcal{B}$ accepts t_u. Precisely, there is a regular $L \subseteq (\Sigma' \cup \mathsf{Dir}(\Sigma'))^*$ such that a node u of t satisfies φ iff the word obtained by reading in $\bar{t}$ the labels and directions from the root to the node u belongs to L.

Finally an n-CPDA generating t_φ is obtained by taking a synchronised product between an n-CPDA accepting $\bar{t}$ and a finite *deterministic* automaton recognising L. ■

Remark 4. In a Σ-labelled tree, a node u may be identified with the word obtained by reading the node-labels and directions along the unique path from the root to u. Call this word $\mathit{path}(u) \in (\Sigma \cup \mathsf{Dir}(\Sigma))^*$. Let $\mathcal{R}$ be a class of generators of Σ-labelled trees, and $\mathcal{B}$ be a finite-state word automaton over the alphabet $\Sigma \cup \mathsf{Dir}(\Sigma)$. Let $R \in \mathcal{R}$ and we write $[\![R]\!]$ for the tree defined by R. We say that $R_\mathcal{B}$ is a *$\mathcal{B}$-reflection* of R just if (i) $\mathsf{Dom}(R) = \mathsf{Dom}(R_\mathcal{B})$, and (ii) suppose a node u of $[\![R]\!]$ has label f, then the label of node u of $[\![R_\mathcal{B}]\!]$ is $\underline{f}$ if $\mathcal{B}$ accepts $\mathit{path}(u)$, and it is f otherwise. We say that $\mathcal{R}$ is *reflective w.r.t. regular paths* just if there is an algorithm that transforms a given pair $(R, \mathcal{B})$ to $R_\mathcal{B}$. The proof of Corollary 2 can be trivially adapted to obtain the following (more general) result.

Theorem 5. *Let $\mathcal{R}$ be a class of generators of Σ-labelled trees. If $\mathcal{R}$ is reflective w.r.t. modal μ-calculus and w.r.t. regular paths, then it is also reflective w.r.t. MSO.*

A natural extension of this result is to use MSO to define new edges in the structure and not simply to mark certain nodes. This corresponds to the well-know mechanism of MSO-interpretations [6]. Furthermore to obtain trees, we unfold the obtained graph from one of its nodes. As MSO-interpretations and unfolding are graph transformations which preserve the decidability of MSO, we obtain a tree with a decidable MSO-theory. Combining these two transformations provides a very powerful mechanism for constructing infinite graphs with a decidable MSO-theory. If we only use MSO-interpretations followed by unfolding to produce trees starting from the class of finite trees, we obtain the class of value trees of safe recursive schemes [13], [5]. This class of trees is conjectured to be a proper subclass of the value trees of recursion schemes.

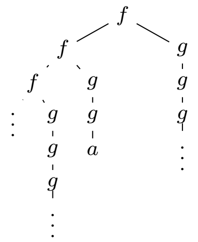
We present here a definition of MSO-interpretations which is tailored to our setting. An MSO-interpretation over Σ-labelled trees is given by a domain formula $\varphi_\delta(x)$, a formula $\varphi_\sigma(x)$ for each $\sigma \in \Sigma$ and a formula $\varphi_d(x, y)$ for each direction $d \in \mathsf{Dir}(\Sigma)$. When applied to a Σ-labelled tree t, $\mathcal{I}$ produces a graph, denoted $\mathcal{I}(t)$, whose vertices are the vertices of t satisfying $\varphi_\delta(x)$. A vertex u of $\mathcal{I}(t)$ is coloured by σ iff u satisfies $\varphi_\sigma(x)$ in t. Similarly there exists an edge labelled by $d \in \mathsf{Dir}(\Sigma)$ from a vertex u to a vertex v iff the pair (u, v) satisfies the formula $\varphi_d(x, y)$ in t.

We say that $\mathcal{I}$ is *well-formed* if for all Σ-labelled trees t, every vertex u of $\mathcal{I}(t)$ is coloured by exactly one $\sigma \in \Sigma$ and has exactly one out-going edge for each direction in $\mathsf{Dir}(\sigma)$. Here we restrict our attention to well-formed interpretations,[4] which ensures that after unfolding of the interpreted graph, we obtain a deterministic tree respecting the arities of Σ.

Consider the MSO-interpretation $\mathcal{I}$ which removes all nodes below a node labelled by $\underline{g}$. All colours are preserved except for $\underline{g}$ which is renamed to g. Finally all edges are preserved and a loop labelled by g is added to every node previously coloured by $\underline{g}$. It is easily seen that $\mathcal{I}$ is a well-formed interpretation. By applying $\mathcal{I}$ to the tree t of the example above and then unfolding it from its root, we obtain the tree on the right which is generated by the scheme on the left:

[4] Given an MSO-interpretation $\mathcal{I}$, we can decide if it is well-form. In fact, we can construct an MSO-formula $\varphi_\mathcal{I}$ which holds on the complete binary tree iff $\mathcal{I}$ is well-formed [17].

$$\begin{cases} I & \rightarrow & \underline{F}\, g\, (g\, a) \\ G & \rightarrow & g\, G \\ \underline{F}\, \varphi\, x & \rightarrow & f\, (F\, g\, (\varphi\, x))\, G \\ F\, \varphi\, x & \rightarrow & f\, (\underline{F}\, g\, (\varphi\, x))\, x \end{cases}$$



More generally, we have the following result.

Corollary 3. *Let t be a Σ-labelled tree given by an order-n recursion scheme $\mathcal{S}$ and let $\mathcal{I}$ be a well-formed MSO-interpretation. The unfolding of $\mathcal{I}(t)$ from any vertex u can be generated by an order-$(n+1)$ recursion scheme.*

Remark 5. *A natural question is whether every tree generated by order-$(n+1)$ recursion scheme can be obtained by unfolding a well-formed MSO-interpretation of a tree generated by an order-n recursion scheme. This is for instance true when considering the subfamily of safe recursion schemes [12], [5]. A positive answer for general recursion schemes would imply safe schemes of any given order are as expressive (for generating trees) as unsafe ones of the same level. This can be established by induction on the order with the base case following from the definition of safety. However already at order 2, unsafe recursion schemes are widely conjecture to generate more trees then safe ones (see for instance the so-called Urzyczyn language in [15]).*

Conclusions and Further Directions: Using a constructive notion of *logical reflection*, we have shown: (i) The global model checking problem may be approached fruitfully from a new, internal angle. (ii) The class of trees generated by HORS is robust: it is closed under both modal μ-calculus and MSO reflections, and the operation à la Caucal of MSO-interpretation followed by tree unfolding.

We believe that our results on reflection is relevant to verification and program transformation; demonstrating that it is so is our most pressing future work.

REFERENCES

[1] J. Bradfield and C. Stirling. Modal logics and mu-calculi. In *Handbook of Process Algebra*, pages 293–332. Elsevier, North-Holland, 2001.

[2] C. Broadbent, A. Carayol, C.-H. L. Ong, and O. Serre. Recursion schemes and logical reflection. Preprint, 2010. `www.liafa.jussieu.fr/~serre`.

[3] C. Broadbent and C.-H. L. Ong. On global model checking trees generated by higher-order recursion schemes. In *Proc. of FoSSaCS 2009*, pages 107–121, 2009.

[4] A. Carayol, M. Hague, A. Meyer, C.-H. L. Ong, and O. Serre. Winning regions of higher-order pushdown games. In *Proc. of LICS'08*, pages 193–204. IEEE, 2008.

[5] D. Caucal. On infinite terms having a decidable monadic theory. In *Proc. of MFCS'02*, pages 165–176, 2002.

[6] B. Courcelle. Monadic second-order definable graph transductions: A survey. *TCS*, 126(1):53–75, 1994.

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

[8] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of FoCS'91*, pages 368–377.

[9] M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In *Proc. of LICS'08*, pages 452–461. IEEE, 2008.

[10] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *Proc. of Concur'96*, pp. 263–277. 1996.

[11] A. Kartzow. Collapsible pushdown graphs of level 2 are tree-automatic. In *Proc. of STACS 2010*, pp. 501-512. 2010.

[12] T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Proc. of FoSSaCS02*, pp 205–222.

[13] T. Knapik, D. Niwiński, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *Proc. of ICALP'05*, volume 3580 of *LNCS*, pages 1450–1461. Springer, 2005.

[14] D.A. Martin. Borel determinacy. *Annals of Mathematics*, 102(363-371), 1975.

[15] J. de Miranda. Structures generated by higher-order grammars and the safety constraint. Ph.D. dissertation, University of Oxford, 2006.

[16] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proc. of LICS'06*, pages 81–90.

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

[18] O. Serre Note on winning positions on pushdown games with omega-regular winning conditions. *IPL*, 85:285-291, 2003.

[19] C. Stirling. Dependency tree automata. In *Proc. of FoSSaCS09*, pp 92–106.

[20] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Language Theory*, volume III, pages 389–455. Springer, 1997.

[21] M. Y. Vardi and N. Piterman. Global model-checking of infinite-state systems. In *Proc. of CAV 2004*, pages 387–400.