Recursion Schemes and Logical Reflection

Christopher Broadbent, Arnaud Carayol, Luke Ong, Olivier Serre

▶ **To cite this version:**

Christopher Broadbent, Arnaud Carayol, Luke Ong, Olivier Serre. Recursion Schemes and Logical Reflection. Twenty-Fifth Annual IEEE Symposium on Logic in Computer Science (LICS 2010), 2010, Edinburgh, United Kingdom. pp.120-129. hal-00479818

HAL Id: hal-00479818
https://hal.science/hal-00479818

Submitted on 2 May 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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. [2] 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 [3] and algebraic (or order-1) trees [4]. We now know [5] 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 [6] 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 [1] 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.

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 $[\![R]\!]$ has label f, then the label of node u of $[\![R_{\mathcal{B}}]\!]$ 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 [7] 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. [8] 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 [9] showed that for every order-n recursion scheme $\mathcal{S}$, the set of nodes in $[\![\mathcal{S}]\!]$ 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 [10] 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 $\rightarrow$. Every type A can be written uniquely as $A_1 \rightarrow \cdots \rightarrow A_n \rightarrow o$ (arrows associate to the right), for some $n \geq 0$ which is called its *arity*; we shall often write A simply as $(A_1, \cdots, A_n, o)$. We define the *order* of a type by $ord(o) := 0$ and $ord(A \rightarrow B) := \max(ord(A)+1, ord(B))$. Let Σ be a *ranked alphabet* i.e. each 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 $t : \mathsf{Dom}(t) \rightarrow \Sigma$ 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.

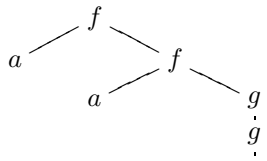
For each type A, we assume an infinite collection Var^A of variables of type A, and write Var to be the union of Var^A as A ranges over types; we write $t : A$ to mean that the expression t has type A. A (deterministic) *recursion scheme* is a tuple $\mathcal{S} = \langle \Sigma, \mathcal{N}, \mathcal{R}, I \rangle$ 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 $F\, \xi_1 \cdots \xi_n \rightarrow e$ 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 $[\![\mathcal{S}]\!]$, 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. [6] 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.

[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 $i = 1$.

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 $[[\,\bot\,][\,\bot\ \overset{\frown}{\alpha}\ \overset{\frown}{\gamma}\,]]$.

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

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

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

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

$$[[[\,\alpha\,]]\overset{\frown}{[}[\,]\overset{\frown}{[}\,\alpha\,\beta\,\gamma\,]]\ [[\,]\overset{\frown}{[}\,\alpha\,\beta\,\gamma\,]]].$$

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

$$[[\bot\alpha]\overset{\frown}{[}\bot\beta]\overset{\frown}{[}\bot\beta]]$$

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

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 [6] (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 [11] 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 [5]. 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

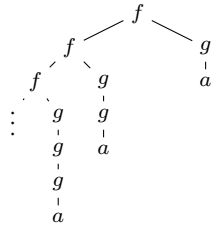
device recognising $\|t\|_\varphi \subseteq \mathsf{Dir}(\Sigma)^*$.

- *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:

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



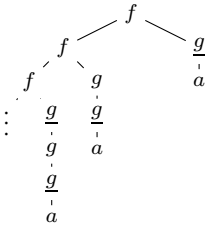
where the arities of the terminals f, g, a are $2, 1, 0$ respectively. The value tree $t = [\![\mathcal{S}]\!]$ is the Σ-labelled tree depicted above.

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:

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



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 [13]. 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, $\widetilde{s} \in (\Sigma \cup \{\mathtt{[}, \mathtt{]}\})^*$:

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

In order to reflect the link structure, we define a partial function $target(s) : \{1, \cdots, |\widetilde{s}|\} \to \{1, \cdots, |\widetilde{s}|\}$ 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 $\bot$, $\mathtt{[}$ and $\mathtt{]}$). Thus with s is associated the pair $\langle\, \widetilde{s}, target(s)\, \rangle$; and with a set S of stacks is associated the set $\widetilde{S} = \{\langle\, \widetilde{s}, target(s)\, \rangle \mid s \in S\}$.

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

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

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 [12] for syntax and semantics of μ-calculus.

of these automata as a deterministic version of Stirling's *dependency tree automata* [14] 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}, ...; 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 [6]. ∎

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 [6], [8] but also extend in a non trivial way their results (decidability of CPDA games of [6] and characterisation of winning region of HOPD games — *i.e.* games generated by CPDA without links — of [8]). The full version [1] 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 [15]). 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 $\mathcal{A}_{\rm rk}$ 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 $\mathcal{A}_{\rm rk}$ one naturally gets a new parity game $\mathbb{G}_{\rm rk}$ and a transformation ν_1 from any vertex v in $\mathbb{G}$ to a vertex $\nu_1(v)$ in $\mathbb{G}_{\rm rk}$ such that Éloïse wins in $\mathbb{G}$ from v iff she wins from $\nu_1(v)$ in $\mathbb{G}_{\rm rk}$. 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 $\mathbb{G}_{\rm rk}$ 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 $\overrightarrow{R} = (R_0, \cdots R_d) \in (2^{Q_{\rm rk}})^{d+1}$ — here $Q_{\rm rk}$ are the control states of $\mathcal{A}_{\rm rk}$ — 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 $\mathcal{A}_{\rm rk}$ 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 $\mathbb{G}_{\rm lf}$ (lf for n-link free) this new game. Then one can define a transformation ν_2 from any vertex v in $\mathbb{G}_{\rm rk}$ to a vertex $\nu_2(v)$ in $\mathbb{G}_{\rm lf}$ such that Éloïse wins in $\mathbb{G}_{\rm rk}$ from v iff she wins from $\nu_2(v)$ in $\mathbb{G}_{\rm lf}$. 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 $\mathbb{G}_{\rm rk}$ from this state with this new stack content: then $\overrightarrow{R} = (R, \cdots, R)$. An automaton deciding whether a configuration in $\mathbb{G}_{\rm lf}$ 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 $\mathbb{G}_{\rm rk}$ 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

$$s = \texttt{[[[}\,\alpha\,\texttt{]]}\ \texttt{[[]}\overset{\frown}{\texttt{[}\,\alpha\,\beta\,\gamma\,\texttt{]]}}\ \texttt{[[]}\overset{\frown}{\texttt{[}\,\alpha\,\beta\,\gamma\,\texttt{]]]}},$$

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

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

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 [8] 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 [8] 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 [12]) 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 $q \in F$.

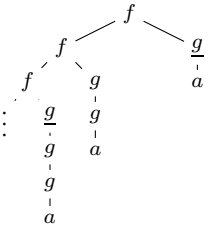
It follows at once that the tree t_φ is defined by $\mathcal{A}'$ with the mapping ρ' defined as follows: for all $q \in Q'$, $\rho'(q) := \rho(q)$ if $q \notin F$, and $\rho'(q) := \underline{\rho(q)}$ otherwise. □

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.* [16]. 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:

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



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 [17]), 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 $path(u) \in (\Sigma \cup \mathsf{Dir}(\Sigma))^*$. 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 $path(u)$, 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 [18]. 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 [19], [20]. This class of trees is conjectured to be a proper subclass of the value trees of recursion schemes.

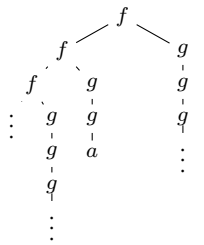
We present here a definition of MSO-interpretations which is tailored to our setting. An MSO-interpretation over Σ-labelled trees is given by a domain formula $\varphi_\delta(x)$, a formula $\varphi_\sigma(x)$ for each $\sigma \in \Sigma$ and a formula $\varphi_d(x, y)$ for each direction $d \in \mathsf{Dir}(\Sigma)$. 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 [3].

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



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 [2], [20]. 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 [21]).*

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] C. Broadbent, A. Carayol, C.-H. L. Ong, and O. Serre, "Recursion schemes and logical reflection," Oxford University Computing Laboratory, Tech. Rep., 2010, preprint, downloable from `www.liafa.jussieu.fr/~serre`.

[2] T. Knapik, D. Niwiński, and P. Urzyczyn, "Higher-order pushdown trees are easy," in *Proceedings of the 5th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS'02)*, ser. Lecture Notes in Computer Science, vol. 2303. Springer-Verlag, 2002, pp. 205–222.

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

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

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

[6] M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre, "Collapsible pushdown automata and recursion schemes," in *Proceeding of 23rd IEEE Symposium on Logic in Computer Science, (LICS 2008)*. IEEE Computer Society Press, 2008, pp. 452–461.

[7] M. Y. Vardi and N. Piterman, "Global model-checking of infinite-state systems," in *Proc. of CAV 2004*, 2004, pp. 387–400.

[8] A. Carayol, M. Hague, A. Meyer, C.-H. L. Ong, and O. Serre, "Winning regions of higher-order pushdown games," in *Proceeding of 23rd IEEE Symposium on Logic in Computer Science, (LICS 2008)*. IEEE Computer Society Press, 2008, pp. 193–204.

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

[10] A. Kartzow, "Collapsible pushdown graphs of level 2 are tree-automatic," in *Proc. of STACS 2010*, 2010, to appear.

[11] D. Martin, "Borel determinacy," *Annals of Mathematics*, vol. 102, no. 363-371, 1975.

[12] J. Bradfield and C. Stirling, "Modal logics and mu-calculi," in *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. Elsevier, North-Holland, 2001, pp. 293–332.

[13] E. Emerson and C. Jutla, "Tree automata, mu-calculus and determinacy (extended abstract)," in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, FoCS'91*. IEEE Computer Society Press, 1991, pp. 368–377.

[14] C. Stirling, "Dependency tree automata," in *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS'09)*, ser. Lecture Notes in Computer Science, vol. 5504. Springer-Verlag, 2009, pp. 92–106.

[15] O. Serre, "Note on winning positions on pushdown games with omega-regular winning conditions," *Information Processing Letters*, vol. 85, pp. 285–291, 2003.

[16] D. Janin and I. Walukiewicz, "On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic," in *Proceedings of Concurrency Theory, 7th International Conference (Concur'96)*, ser. Lecture Notes in Computer Science, vol. 1119. Springer-Verlag, 1996, pp. 263–277.

[17] W. Thomas, "Languages, automata, and logic," in *Handbook of Formal Language Theory*, G. Rozenberg and A. Salomaa, Eds. Springer-Verlag, 1997, vol. III, pp. 389–455.

[18] B. Courcelle, “Monadic second-order definable graph transductions: A survey.” *Theoretical Computer Science*, vol. 126, no. 1, pp. 53–75, 1994.

[19] T. Knapik, D. Niwiński, P. Urzyczyn, and I. Walukiewicz, “Unsafe grammars and panic automata,” in *Proceedings of Automata, Languages and Programming, 32nd International Colloquium (ICALP’05)*, ser. Lecture Notes in Computer Science, vol. 3580. Springer-Verlag, 2005, pp. 1450–1461.

[20] D. Caucal, “On infinite terms having a decidable monadic theory,” in *Proceedings of Mathematical Foundations of Computer Science 2002, 27th International Symposium (MFCS’02)*, ser. Lecture Notes in Computer Science, vol. 2420. Springer-Verlag, 2002, pp. 165–176.

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

Appendix

Non-deterministic automata are strictly more powerful than deterministic automata.

Indeed, 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 one has the following two results.

Proposition 3. *The set L cannot be recognised by a deterministic automaton.*

Proof: Assume $\mathcal{A} = (Q, A, q_{in}, F, \delta)$ accepts L. Take any input $\langle u, \tau \rangle$ in L with strictly more than $|Q|^2$ links. Let r be the (accepting) run of $\mathcal{A}$ over $\langle u, \tau \rangle$. Then by the pigeon hole principal, there are two pairs (i, i') and (j, j') such that:

- $i < i'$ and $j < j'$;
- $i = \tau(i' + 1)$ and $j = \tau(j' + 1)$;
- $r_i = r_j$ and $r_{i'} = r_{j'}$.

Consider the input $\langle u, \tau' \rangle$ where $\tau'(k) = \tau(k)$ if $k \neq j' + 1$ and $\tau'(j' + 1) = \tau'(i' + 1)$: hence $\langle u, \tau \rangle$ is obtained by changing the link from $j' + 1$ to have the same target as the one from $i' + 1$. It follows from how (i, i') and (j, j') were defined that r is also a run over $\mathcal{A}$. However $\langle u, \tau' \rangle$ is not in L, leading a contradiction. ∎

Proposition 4. *The complement $\overline{L}$ of L can be recognised by a non-deterministic automaton.*

Proof: The language $\overline{L}$ consists of the words with two links having the same target: to recognised this language, a non-deterministic automaton guesses by going into a special state the target. Then whenever reading a letter that points to that position one increments a counter from 0 to 1 and then from 1 to 2 and when the counter is 2 it loops in a final state: hence a word is accepted iff it belongs to $\overline{L}$. ∎

Hence one can concludes.

Proposition 5. *Non-deterministic automata are strictly more expressive than deterministic ones.*

Proof: The deterministic model being closed by complementation, if $\overline{L}$ was recognised by a deterministic automaton, it would be the same for L, contradicting Proposition 3. ∎

We aim to establish the following:

Proposition 2. *Given fixed $n \geq 2$ and some automaton $\mathcal{A}$, deciding whether there exists some order-n collapsible stack (resp. configuration) that it accepts is NP-complete.*

A. The upper-bound

First we show that the problem lives in NP.

We extend the notion of a run of an automaton $\mathcal{A}$ on $\langle \widetilde{s}, target(s) \rangle$ for some order-n stack s so that $\mathcal{A}$ can also have a run on a stack of order less than n contained within s without explicit reference to its context. So let $1 \leq k \leq n$ and let t be some k-stack contained within s (if $k = n$ then we must have $s = t$). Of course t might contain some 'dangling links' which are those links bearing an order greater than k. We allow $\mathcal{A}$ to handle such links by means of an $(n - k)$-tuple $(Q_n, \ldots, Q_{k+1})$ so that the automaton is allowed to 'pretend' that an i-link points to a position associated with a state in Q_i.

More formally suppose that $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ where $\Sigma = \{[_i,]_i \ : \ 1 \leq i \leq n\} \cup \Gamma$ for some integer n and some stack-alphabet Γ. Given an order-n stack s over Γ we choose (without loss of generality) to represent $\widetilde{s}$ using indexed brackets so that $[_i$ and $]_i$ delimit an i-stack. We also assume that $target(s)$ is only used to define links of order at least 2 (i.e. 1-links are ignored). We define a partial function

$$ordl(s) \ : \ \{1 \cdots |\widetilde{s}|\} \longrightarrow \{2 \cdots n\}$$

that specifies the order of a link from any given element of the stack s (and is undefined on positions in $\widetilde{s}$ labelled with a bracket). Given an order-k stack t (for $1 \leq k \leq n$) contained in s, we define $\widetilde{t}$ to be the corresponding subsequence of $\widetilde{s}$ and $target(t)$ (resp. $ordl(t)$) to be the natural restriction of $target(s)$ (resp. $ordl(s)$) to the positions in t.

Remark 6. *Note that for some position $i \in \{1 \cdots |\widetilde{t}|\}$ of $\widetilde{t}$, if $i \in Dom(ordl(t))$ with $ordl(t)(i) \leq k$, then $i \in Dom(target(t))$.*

Suppose that $\widetilde{t} = a_1 \ldots a_m$. Given $(n - k)$ subsets $Q_n, \ldots, Q_{k+1}$ of Q, a $(Q_n, \ldots, Q_{k+1})$-run of $\mathcal{A}$ on $\langle \widetilde{t}, target(t), ordl(t) \rangle$ is a sequence of states $q_0 q_1 \ldots q_m \in Q^*$ such that:

- for all $0 \leq i < m$, $q_{i+1} = \delta(q_i, a_{i+1})$ if $i + 1 \notin Dom(ordl(r))$ (and so $i + 1 \notin Dom(target(t))$)
- for all $0 \leq i < m$, $q_{i+1} = \delta(q_i, a_{i+1}, q)$ for $q \in Q_{ordl(t)(i+1)}$ if $i + 1 \in Dom(ordl(t))$ with $ordl(t)(i + 1) > k$
- for all $0 \leq i < m$, $q_{i+1} = \delta(r_i, a_{i+1}, q_{target(t)(i+1)})$ if $i + 1 \in Dom(ordl(t))$ with $ordl(t)(i + 1) \leq k$.

Remark 7. *In the special case when $k = n$ (and so $t = s$) $()$-runs of $\mathcal{A}$ on $\langle \widetilde{t}, target(t), ordl(t) \rangle$ are exactly the (ordinary) runs of $\mathcal{A}$ on $\langle \widetilde{t}, target(t) \rangle$.*

Let $k \geq 2$ and t be a k-stack contained with the n-stack s. The string $\widetilde{t}$ must be of the form:

$$[_k \, [_{k-1} \vec{w_1}]_{k-1} \, [_{k-1} \vec{w_2}]_{k-1} \, \cdots \, [_{k-1} \vec{w_l}]_{k-1}]_k$$

We say that *the k-height of t* is l.

Lemma 1. *Suppose that $\mathcal{A}$ has a $(Q_n, \ldots, Q_{k+1})$-run ρ on $\langle \widetilde{t}, target(t), ordl(t) \rangle$ starting with q and ending with q', then there exists a k-stack t' (residing in an n-stack s') of k-height bounded by $(|Q| + 1).|Q|^2$ such that $\mathcal{A}$ has a $(Q_n, \ldots, Q_{k+1})$-run ρ' on $\left\langle \widetilde{t'}, target(t'), ordl(t') \right\rangle$ starting*

with q and ending with q'. Moreover the $(k-1)$-stacks occurring inside t' all occur inside t as well.

Proof: The run ρ must be of the following form:

$$qp_1\vec{r_1}q_1p_2\vec{r_2}q_2\dots p_{l-1}\vec{r_{l-1}}q_lq'$$

where p_i is the state reached upon reading the ith occurrence of $[_{k-1}$ and q_i is the state reached upon reading the ith occurrence of $]_{k-1}$ for $1 \leq i \leq l$. Let $R_i := \{q_j : 1 \leq j < i\}$. Observe that $R_i \subseteq R_{i+1}$ for every $1 \leq i < l$. Since each $R_i \subseteq Q$ there can be at most $|Q|+1$ distinct R_i (remember $\emptyset$) and thus at most $(|Q|+1).|Q|^2$ distinct triples (p_i, q_i, R_i) with $1 \leq i \leq l$.

Let ρ' be the subsequence of ρ:

$$\rho' := qp_{i_1}\vec{r_{i_1}}q_{i_1}p_{i_2}\vec{r_{i_2}}q_{i_2}\dots p_{i_m}\vec{i_m}q_{i_m}q'$$

such that:

- i_1 is the greatest j such that $(p_j, q_j, R_j) = (p_1, q_1, \emptyset)$
- For $1 \leq j' < l$, $i_{j'+1}$ is the greatest j such that $p_j = p_{i_{j'}+1}$, $q_j = q_{i_{j'}+1}$ and $R_j = R_{j'} \cup \{q_{i_{j'}}\}$.

We can easily check the following two properties of ρ':

Property 2.
- *Given a triple $(p_{j'}, q_{j'}, R_{j'})$ for some $1 \leq j' \leq l$, there is at most one j such that $(p_{i_j}, q_{i_j}, R_{i_j}) = (p_{j'}, q_{j'}, R_{j'})$.*
  Thus $l \leq (|Q|+1).|Q|^2$
- *Given a $1 \leq j \leq m$, if $u \in R_{i_j}$, there exists a $1 \leq j' < j$ such that $u = q_{i_{j'}}$. Let us write $\mathbf{witness}(u, j)$ to denote the position of u.*

The first item is due to the fact that we always select the right-most element of the sequence that satisfies a given equality and the second follows immediately from the definition of R_{i_j} and the way that R_{i_j+1} is generated from R_{i_j} and q_{i_j}.

We now define the k-stack t in terms of $\widetilde{t'}$, $target(t')$ and $ordl(t')$ as follows:

- The string $\widetilde{t'}$ is given by:

  $$\widetilde{t'} := [_k\, [_{k-1}\vec{w_{i_1}}]_{k-1}\, [_{k-1}\vec{w_{i_2}}]_{k-1}\, \dots\, [_{k-1}\vec{w_{i_l}}]_{k-1}]_k$$

  For any position a' in $\widetilde{t'}$ let us write a to denote the corresponding position in $\widetilde{t}$.
- $ordl(t')$ is the restriction of $ordl(t)$ to t'
- For any stack-alphabet position γ' in t' we have $target(t')(\gamma') = target(t)(\gamma)$, if $ordl(t')(\gamma') \leq k-1$. If $ordl(t')(\gamma) = k$, then $target(t') := \mathbf{witness}(u, j)$ where u is the state in position $target(t)(\gamma)$ in ρ.

Links internal to the $(k-1)$-stacks are preserved from t. Now consider a position γ' in $\widetilde{t'}$ (and ρ') which resides in a w_{i_j} (r_{i_j}). Since the state p at position $target(t)(\gamma)$ in ρ occurs at a position prior to γ in ρ, it must be a state $u := q_{j'}$ with $j' < i_j$. This means that $u \in R_{i_j}$. By the second item of Property 2, $\mathbf{witness}(u, j)$ is thus well-defined. Moreover, since this occurs at a $]_{k-1}$-labelled position, t' is a correctly formed k-stack.

We claim that ρ' is a $(Q_n, \dots, Q_{k+1})$-run of $\left\langle \widetilde{t'}, target(t'), ordl(t') \right\rangle$ on $\mathcal{A}$. This is verified by an easy induction with the hypothesis that an initial segment of ρ' is a $(Q_n, \dots, Q_{k+1})$-run of the corresponding initial segment of $\left\langle \widetilde{t'}, target(t'), ordl(t') \right\rangle$ on $\mathcal{A}$. The induction-step prevails since the construction of ρ' together with the handling of k-links in the definition of $target(t')$ ensures that every transition made by $\mathcal{A}$ when reading $\left\langle \widetilde{t'}, target(t'), ordl(t') \right\rangle$ with run ρ' was made when reading $\langle \widetilde{t}, target(t), ordl(t) \rangle$ with run ρ.

Finally note that the first item of Property 2 tells us that t has k-height bounded by $(|Q|+1).|Q|^2)$, as required. ∎

Note that the following lemma also holds and is established by the standard pumping-argument for finite-automata. (As links are ignored in $(Q_n, \dots, Q_2)$-runs $\mathcal{A}$ may as well be a conventional finite automaton for the purposes of this lemma):

Lemma 2. *If $\mathcal{A}$ has a $(Q_n, \dots, Q_2)$-run (for some $Q_n, \dots, Q_2 \subseteq Q$) starting in state q and ending in state q' on a 1-stack t contained in s, then it has a $(Q_n, \dots, Q_2)$-run starting in state q and ending in state q' on a 1-stack t' with height at most $|Q|$.*

Given an integer k, let us define a sequence of integers $(\psi_i(k))_{1 \leq i}$ by $\psi_1(k) := k+2$ and $\psi_{i+1} := ((k+1).k^2)^{k-1}\psi_i(k) + 2$. As a consequence of the previous two lemmas we get:

Lemma 3. *Let t be a k-stack contained within an n-stack s (for $1 \leq k \leq n$) and let $Q_n, \dots, Q_{k+1} \subseteq Q$. If an automaton $\mathcal{A}$ has a $(Q_n, \dots, Q_{k+1})$-run from q to q' on $\langle \widetilde{t}, target(t), ordl(t) \rangle$, then it has a $(Q_n, \dots, Q_{k+1})$-run from q to q' on $\left\langle \widetilde{t'}, target(t'), ordl(t') \right\rangle$ for some k-stack t' such that $|\widetilde{t}| \leq \psi_k(|Q|)$.*

Proof: Argue by induction on k. The base case ($k = 1$) is given by Lemma 2 (adding 2 to the length to account for the opening and closing brackets $[_1$ and $]_1$).

For the induction step suppose that the result holds for $k < n$. Let t be a $(k+1)$-stack contained in s where

$$\widetilde{t} = [_{k+1}\, u_1 \dots u_m\,]_{k+1}$$

Suppose further that there is a $(Q_n, \dots, Q_{k+2})$-run ρ on $\langle \widetilde{t}, target(t), ordl(t) \rangle$. We write p_i and q_i respectively for the state in ρ that arises for entering and exiting the k-stack s_i (for $1 \leq i \leq m$).

Let $R_i := \{q_j : 1 \leq j \leq i\}$. Since $(k+1)$-links from elements in a stack s_i must point to the end of a stack s_j for $1 \leq j < i$, we have it that there is a $(Q_n, \dots, Q_{k+2}, R_i)$-run on s_i starting in p_i and ending in q_i for each $1 \leq i \leq m$.

The induction hypothesis thus tells us that we can replace each s_i with a stack u'_i such that $|\widetilde{s'_i}| \leq \psi_k(|Q|)$.

It follows that there is a $(k+1)$-stack t' containing k-stacks u with $|u| \leq \psi_k(|Q|)$ such that there is a $(Q_n, \ldots, Q_{k+2})$-run on $\left\langle \widetilde{t'}, target(t'), ordl(t') \right\rangle$. To finish the induction-step we appeal to Lemma 1. ∎

As a corollary to the previous lemma, taking the case when $k = n$, we may conclude that if an automaton $\mathcal{A}$ recognises any stack, then it must accept a small stack:

Lemma 4. *Let $\mathcal{A}$ be an automaton recognising some n-stack. It must accept an n-stack s such that $|\widetilde{s}| \leq \psi_n(|Q|)$, where Q is the state-space of $\mathcal{A}$.*

Since $\psi_n(x)$ is a polynomial it follows that an automaton $\mathcal{A}$ recognising some stack must have a witness to this fact that is polynomial in the size of $\mathcal{A}$. Since membership can be decided in linear-time, it follows that the stack-emptiness problem is indeed in NP.

B. The lower-bound

Now we show NP-hardness. We do this by reducing 3-SAT to the emptiness problem.

We say that a propositional formula φ is in 3-*conjunctive normal form (*3-*CNF)* if it is of the form:

$$(x_1 \vee y_1 \vee z_1) \wedge (x_2 \vee y_2 \vee z_2) \wedge \cdots \wedge (x_k \vee y_k \vee z_k)$$

where x_i, y_i and z_i (for $1 \leq i \leq k$) are either propositional atoms or negations of propositional atoms. The problem 3-SAT takes as input a propositional formula in 3-CNF and asks whether there is a valuation satisfying it. It is well known that this problem is NP-complete.

Consider an alphabet Γ:

$$\Gamma_\varphi := \{tt, ff, \circ_1, \bullet_1, \ldots, \circ_k, \bullet_k\}$$

. Take a propositional formula φ in 3-CNF. Without loss of generality we may assume that every conjunctive clause contains *exactly* three atoms – if need be we can repeat a (negation of an) atom in a disjunctive clause without affecting satisfiability. So we have:

$$\varphi = (x_1 \vee y_1 \vee z_1) \wedge (x_2 \vee y_2 \vee z_2) \wedge \cdots \wedge (x_k \vee y_k \vee z_k)$$

Let $p_1, \ldots, p_m$ be the propositional variables occurring in φ. Let v be a valuation assigning a boolean value to each of the p_i. We now define a string $\Gamma(\varphi, v) \in \Gamma^*$ together with a partial function $target(\varphi, v)$ from positions in $\Gamma(\varphi, v)$ to other positions therein:

$$\Gamma(\varphi, v) = v_1 \ldots v_m x'_1 y'_1 z'_1 \ldots x'_k y'_k z'_k$$

where:

- $v_i = v(p_i)$ for each $1 \leq i \leq m$.
- $x'_i = \circ_j$ if $x_i = p_j$ and $x'_i = \bullet_j$ if $x_i = \neg p_j$
- $y'_i = \circ_j$ if $y_i = p_j$ and $y'_i = \bullet_j$ if $y_i = \neg p_j$
- $z'_i = \circ_j$ if $z_i = p_j$ and $z'_i = \bullet_j$ if $z_i = \neg p_j$
- $target(\varphi, v)(x'_i) := v_j$ where x_i is the atom p_j or the negation of the atom p_j (and similarly for y'_i and z'_i) for $1 \leq i \leq k$.

Remark 8. *It should be clear that v satisfies φ just in case for every $1 \leq i \leq k$ at least one of the following holds of $\Gamma(\varphi)$:*

- $x'_i = \circ$ *and* $target(\varphi, v)(x'_i) = tt$ *or* $x'_i = \bullet$ *and* $target(\varphi, v)(x'_i) = ff$
- $y'_i = \circ$ *and* $target(\varphi, v)(y'_i) = tt$ *or* $y'_i = \bullet$ *and* $target(\varphi, v)(y'_i) = ff$
- $z'_i = \circ$ *and* $target(\varphi, v)(z'_i) = tt$ *or* $z'_i = \bullet$ *and* $target(\varphi, v)(z'_i) = ff$

Remark 9. *We can easily build a (not-necessarily constructible)* 2-*CPDA stack $s_{\varphi,v}$ over the stack-alphabet Γ such that $\langle \widetilde{s_{\varphi,v}}, target(s_{\varphi,v}) \rangle$ encodes $\langle \Gamma(\varphi, v), target(\varphi, v) \rangle$ in a trivial manner. Where*

$$\Gamma(\varphi, v) = v_1 \ldots v_m x'_1 y'_1 z'_1 \ldots x'_k y'_k z'_k$$

we take $\widetilde{s_{\varphi,v}}$ to be:

$$[_2[_1 v_1]_1 \ldots [_1 v_m]_1 [_1 x'_1 y'_1 z'_1 \ldots x'_k y'_k z'_k]_1]_2$$

and $target(s_{\varphi,v})$ to be derived from $target(\varphi, v)$ by shifting each target one step to the right to point to a $]_1$ position.

Bearing in mind that this is possible, we will continue to use the string with pointers $\langle \Gamma(\varphi, v), target(\varphi, v) \rangle$ as this carries less baggage than $\langle \widetilde{s_{\varphi,v}}, target(s_{\varphi,v}) \rangle$.

We now construct an automaton $\mathcal{A}$ that recognise strings encoding valuations satisfying φ. The pointers allow us to build such an automaton with only a polynomial number of states as they provide non-local access to the values assigned to atoms by a valuation. We thus do not need to have this information to hand locally, which would require an exponential number of states to represent all possible truth-value assignments.

We define:

$$\mathcal{A}_\varphi = \langle Q_\varphi, \Gamma, q_0, \delta_\varphi, F_\varphi \rangle$$

where

- $Q_\varphi := \{q_0, q_1^{tt}, q_1^{ff}, \ldots, q_m^{tt}, q_m^{ff}, r_1^{tt}, r_1^{ff}, s_1^{tt}, s_1^{ff}, t_1^{tt}, \ldots, r_k^{tt}, r_k^{ff}, s_k^{tt}, s_k^{ff}, t_k^{tt}, \mathbf{fail}\}$
- $F_\varphi := \{t_k^{tt}\}$
- We state once and for all that we only allow a transition *to* r_i whilst reading a symbol $\circ_j$ if $x_i = p_j$. Likewise we only allow a transition *to* r_i whilst reading a symbol $\bullet_j$ if $x_i = \neg p_j$. A similar restriction applies to s_i with y_i and t_i with z_i. We do not re-state below these restrictions explicitly to assist with readability.
  - $\delta_\varphi(q_i, tt) := q_{i+1}^{tt}$ and $\delta_\varphi(q_i, ff) := q_{i+1}^{ff}$ for $0 \leq i \leq m-1$.
  - $\delta_\varphi(q_m, \circ_j, q_j^{tt}) := r_1^{tt}$ and $\delta_\varphi(q_m, \bullet_j, q_j^{ff}) := r_1^{tt}$
  - $\delta_\varphi(q_m, \circ_j, q_j^{ff}) := r_1^{ff}$ and $\delta_\varphi(q_m, \bullet_j, q_j^{tt}) := r_1^{ff}$

- $\delta_\varphi(r_i^{\mathit{tt}}, a, q_j^b) := s_i^{\mathit{tt}}$ with $a \in \{\circ_j, \bullet_j\}$ and $b \in \{\mathit{tt}, \mathit{ff}\}$ for each $1 \leq j \leq m$ and $1 \leq i \leq k$.
- $\delta_\varphi(s_i^{\mathit{tt}}, a, q_j^b) := t_i^{\mathit{tt}}$ with $a \in \{\circ_j, \bullet_j\}$ and $b \in \{\mathit{tt}, \mathit{ff}\}$ for each $1 \leq j \leq m$ and $1 \leq i \leq k$.
- $\delta_\varphi(t_i^{\mathit{tt}}, \circ_j, q_j^{\mathit{tt}}) := r_{i+1}^{\mathit{tt}}$ and $\delta_\varphi(t_i^{\mathit{tt}}, \bullet_j, q_j^{\mathbf{f}}) := r_{i+1}^{\mathit{tt}}$ for $1 \leq i \leq k-1$ and $1 \leq j \leq m$.
- $\delta_\varphi(t_i^{\mathit{tt}}, \circ_j, q_j^{\mathbf{f}}) := r_{i+1}^{\mathbf{f}}$ and $\delta_\varphi(t_i^{\mathit{tt}}, \bullet_j, q_j^{\mathit{tt}}) := r_{i+1}^{\mathbf{f}}$ for $1 \leq i \leq k-1$ and $1 \leq j \leq m$.
- $\delta_\varphi(r_i^{\mathbf{f}}, \circ_j, q_j^{\mathit{tt}}) := s_i^{\mathit{tt}}$ and $\delta_\varphi(r_i^{\mathbf{f}}, \bullet_j, q_j^{\mathbf{f}}) := s_i^{\mathit{tt}}$ for $1 \leq i \leq k$ and $1 \leq j \leq m$.
- $\delta_\varphi(s_i^{\mathbf{f}}, \circ_j, q_j^{\mathit{tt}}) := t_i^{\mathit{tt}}$ and $\delta_\varphi(s_i^{\mathbf{f}}, \bullet_j, q_j^{\mathbf{f}}) := t_i^{\mathit{tt}}$ for $1 \leq i \leq k$ and $1 \leq j \leq m$.
- $\delta_\varphi(r_i^{\mathbf{f}}, \circ_j, q_j^{\mathbf{f}}) := s_i^{\mathbf{f}}$ and $\delta_\varphi(r_i^{\mathbf{f}}, \bullet_j, q_j^{\mathit{tt}}) := s_i^{\mathbf{f}}$ for $1 \leq i \leq k$ and $1 \leq j \leq m$.
- $\delta_\varphi(s_i^{\mathbf{f}}, \circ_j, q_j^{\mathbf{f}}) :=$ **fail** and $\delta_\varphi(s_i^{\mathbf{f}}, \bullet_j, q_j^{\mathit{tt}}) :=$ **fail** for $1 \leq i \leq k$ and $1 \leq j \leq m$.
- Everywhere else the transition function is defined to map to **fail**

Lemma 5. *The $(j+1)$th state in a run of $\mathcal{A}_\varphi$ on $\langle \Gamma(\varphi, v), target(\varphi, v)\rangle$ is $q_j^{v(p_j)}$ for $1 \leq j \leq m$.*

Proof: An easy induction on m. ■

Lemma 6. *The automaton $\mathcal{A}_\varphi$ reaches the state t_j^{tt} when reading $\langle \Gamma(\varphi, v), target(\varphi, v)\rangle$ iff v satisfies all clauses (x_i, y_i, z_i) for $1 \leq i \leq j$.*

Proof: Given Remark 8 and Lemma 5 this is a straightforward induction on the length of the runs ending in t_j^{tt}. ■

The special case of Lemma 6 when $j = k$ gives the following:

Lemma 7. *The automaton $\mathcal{A}_\varphi$ recognises the language:*

$$\mathcal{L}_\varphi = \{\langle \Gamma(\varphi, v), target(\varphi, v)\rangle \ : \ v \text{ satisfies } \varphi\}$$

.

Note that $\mathcal{L}$ is non-empty iff φ is satisfiable. Moreover note that $|Q_\varphi| \leq 4.|\varphi|$ and so (taking into account the size of δ_φ) the size of $\mathcal{A}_\varphi$ is $O(|Q_\varphi|^3)$ and so can be constructed in polynomial time. Given Remark 9 this thus reduces the polynomial-time reduction of 3-SAT to the regular stack emptiness problem, thereby showing it to be NP-hard.

Let us first recall the statement of Theorem 3.

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 any 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:

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 computes $\mathcal{A}$ and $\mathcal{B}$ in parallel.

Let Q be the state set of $\mathcal{B}$. Let S be an order-n stack. Fix an internal k-stack s of S. Reading $pop_k s$, the automaton $\mathcal{B}$ induces a run (for some technical reason we do not care of top $(k-1)$ stack in s). We want to associate with s a function describing its behaviour. However, if one simply extracts s from S, there may be some "dangling link" of order greater than k. To define the transformation s induces on Q, we need to know the Q-state that should be attached to the target of each of these links. Thus s gives rise to a mapping $\tau_k : Q^{n-k} \times Q \to Q$ 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 value of the states attached to the respective targets of the links (of order $n, \cdots, k+1$ respectively) in s: for n-links, we consider the run induced by reading S (we stop when s is reached) starting from q_n (this give 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 is reached) starting from q_{n-1}, ...; and for $(k+1)$-links, we consider the run induced by reading $top_{k+2}(S)$ (again we stop when s is reached) starting from q_{k+1}. We refer the reader to Table B for an illustration in the case where s is the top k-stack (this is actually the only relevant case in the following as we will maintain the validity of the information on the τ_i only in that case) where we have more formally:

- $\tau_0[x_n, \cdots, x_2](x_1)$ is the function from Q to Q induced by reading (the segment of) S starting from $top_2(S)$ (with state given by the input x_1 to the function), and stopping just after reading $top_1(S)$. (For each $k \geq 2$, the targets of the k-links emanating from the segment are attached Q-states according to the run induced by reading S starting from $top_{k+1}(S)$ with state x_k.)

- For each $1 \leq i \leq n-2$, $\tau_i[x_n, \cdots, x_{i+2}](x_{i+1})$ is the function from Q to Q induced by reading (the segment of) S starting from $top_{i+2}(S)$ (with state given by the input x_{i+1} to the function) and stopping just before reading $top_{i+1}(S)$.

- $\tau_{n-1}(x_n)$ is the function from Q to Q induced by reading S (with state given by the input x_n to the function), and stoping just before reading $top_n(S)$.

A stack symbol of the CPDA $\mathcal{A}[\mathcal{B}]$, is a pair, consisting of a symbol a, which is a stack symbol of $\mathcal{A}$, and an n-tuple of the form $t = (\tau_{n-1}, \cdots, \tau_0)$ where the τ_is are as above (*i.e.* τ_i is associated with the top i-stack).

For further use, we define $\tau_0^+[x_n \cdots x_2](x_1)$ to be the same as $\tau_0[x_n \cdots x_2](x_1)$; and for each $0 \leq k \leq n-3$,

$$\begin{array}{rl} & \tau_{k+1}^+[x_n \cdots x_{k+3}](x_{k+2}) \\ := & \tau_k^+[x_n \cdots x_{k+2}](\tau_{k+1}[x_n \cdots x_{k+3}](x_{k+2})). \end{array}$$

Thus each τ_k^+ is a function from Q to Q induced by reading (the segment of) S starting from $top_{k+2}(S)$ (with state given by the input x_{k+1} to the function), and stopping just after

reading $top_1(S)$, as indicated in Table B(ii). As each τ_k^+ can be obtained from the τ_is we assume that we can access them directly on reading the top_1 element of the stack. Note that, considering τ_n^+ applied to the initial state of $\mathcal{B}$ we deduce whether the current stack is accepted by $\mathcal{B}$: hence this information will be maintained in the control state of $\mathcal{A}[\mathcal{B}]$ and is used to define F. The function Ξ is the one erasing all auxiliary information used by $\mathcal{A}[\mathcal{B}]$ in its control state.

Now suppose $top_1(\widehat{S}) = (a, (\tau_{n-1}, \cdots, \tau_0))$. For each order-$n$ stack action θ of $\mathcal{A}$, we define the corresponding stack action of $\mathcal{A}[P]$, $\widehat{\theta}$, in Table B(iii). This complete the description of $\mathcal{A}[\mathcal{B}]$.

Correctness: The only case that are not trivial are $\widehat{pop_k}$ and $\widehat{collapse}$ (actually they are rigorously identical, hence we only prove the first one). Suppose $top_1(\widehat{S}) = (a, (\tau_{n-1}, \cdots, \tau_0))$ and $top_1(pop_k(\widehat{S})) = (a', (\tau'_{n-1}, \cdots, \tau'_0))$. Note that for each $0 \leq i \leq k-1$, we have τ'_i is correct because it is preserved by $push_j$, for each $j \geq k+1$. For each $k \leq i \leq n-1$, we have $\tau_i = \tau'_i$ as required, because $pop_{k+1}(\widehat{S}) = pop_{k+1}(pop_k\widehat{S})$.

■

We give here a fully detailed proof of Theorem 4:

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

In the sequel, n-CPDA are used exclusively for defining games and therefore, we will omit initial state and final states when defining an n-CPDA, *i.e.* denote an n-CPDA as a tuple $\langle A, \Gamma, Q, \delta \rangle$ (hence considering it as a process defining an infinite graph rather than an accepting device).

Mainly for technically and to improve readability, we consider a version of CPDA in which we can rewrite the top_1 element [5]. Hence the statement of Theorem 4 is to be understood in this (richer) setting. Formally, if s is a stack with links, then the stack $rew_1^\beta s$ is the one obtained by replacing the top_1 element of s by β *without modifying the link from this element.*

Example 6. *Take the following 3-stack*

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

then

$$rew_1^\beta s = [[[\,\alpha\,]]\,[[]\,[\,\alpha\,\beta\,\beta\,]]]$$

C. First step: making a CPDA rank aware

Fix an n-CPDA $\mathcal{A} = \langle A, \Gamma, Q, \delta \rangle$, a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q and a colouring function $\rho : Q \to C \subset \mathbb{N}$. Denote by $\mathbb{G}$ the induced parity game.

[5]This can be simulated in the original model (mainly by pushing the new version of a symbol) thanks to ε-moves.

A partial play Λ is a non-empty sequence of configurations $v_0 v_1 \dots v_m$ such that for all $i \in [m-1]$, there is an edge in $\mathbb{G}$ from v_i to v_{i+1}. Note that we do not require v_0 to be the initial configuration.

We first define a generalisation of n-stacks, *indexed n-stacks*, in which every internal k-stack (for $0 < k < n$), *i.e.* a k-stack that is not the current top_k stack, is labelled with a natural number. The *erasure* of an indexed n-stack is the n-stack obtained by erasing all the indices of its internal stacks. An indexed configuration is a pair formed by a control state and an indexed stack. We extend the notion of erasure to indexed configuration in the obvious way.

With any play $\Lambda = v_0 v_1 \cdots$ we inductively associate a sequence of indexed configurations $\Lambda' = v'_0 v'_1 \cdots$ such that the erasure of Λ' equals Λ (the erasure of a sequence of indexed configurations being defined as the sequence of the respective erasures).

The initial configuration v'_0, is obtained by indexing every internal stack by 0. Assume now that $v'_1 \cdots v'_m$ has been constructed, then we have the following cases.

- A $push_k$ operation is applied at configuration v_m in Λ. Then all indices of the existing internal stacks are simply inherited; and indices for the new internal stacks are defined as follows. The indices of the internal stacks in the top $(k-1)$-stack of configuration v_{m+1} are copied from the former top $(k-1)$-stack (hence one can think of this as a generalization of $push_k$ to indexed stacks); and, for each relevant i, the indices of the $top_i(pop_k(s))$ stacks (where s denotes the stack in v'_{m+1}) are assigned index $(m+1)$.
- A $push_1^{a,k}$ operation is applied at configuration v_m in Λ. Then all previous indices are inherited and no new indices are needed.
- A pop_k operation or a $collapse$ operation is applied at configuration v_m in Λ. Then all indices are inherited (except those corresponding to the top_i stacks in v'_{m+1} which have now disappeared).

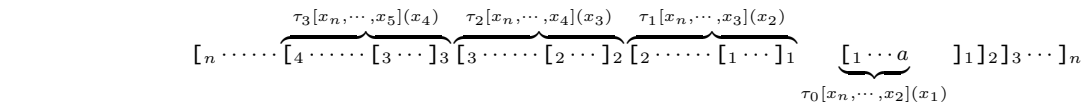
In the sequel, Λ' will denote the indexed version of Λ. Then the following holds:

- The erasure of Λ' equals Λ.
- For any indexed configuration v'_m, for any k-stack s inside the indexed n-stack associated with v'_m, the index of s, if defined, is greater or equal to all indices of the j-stacks ($j < k$) contained in s.

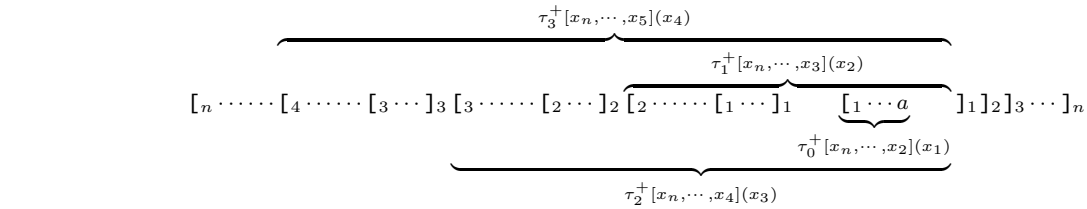
The following proposition is crucial to the rest of the proof. In particular, it means that if we store some information on the stack, the index gives the "expiry date" of the stored information, that is the step in the computation starting from which the information has no longer been updated.

Proposition 6. *Let Λ and Λ' be as above. For any indexed configuration v'_m, for any internal k-stack s inside the indexed n-stack associated with v'_m let x be the index of*

(i) Illustration of $\tau_k[x_n, \cdots, x_{k+2}](x_{k+1})$



(ii) Illustration of $\tau_k^+[x_n, \cdots, x_{k+2}](x_{k+1})$



(iii) Definition of $\widehat{\theta}$. *Notation.* Let t and t' be triples. We write $rw\, t'$ for the action of replacing the top-of-stack element (a, t), say, by (a, t'); and let $2 \leq k \leq n$.

θ	$\widehat{\theta}$
$push_k$	$push_k$; $rw\,(\tau_{n-1}, \cdots, \tau_k, t, \tau_{k-2}, \cdots, \tau_0)$ where $t[x_n, \cdots, x_{k+1}](x_k) := \delta(\tau_{k-1}^+[x_n, \cdots, x_{k+1}](x_k),]_1 \cdots]_{k-1})$
$push_1^{b,k}$	$push_1^{(b,\,(\tau_{n-1},\cdots,\tau_1,t)),k}$ where $t[x_n, \cdots, x_2](x_1) := \delta(\tau_0^+[x_n, \cdots, x_2](x_1), b, \tau_{k+1}[x_n, \cdots, x_{k+3}](x_k))$
pop_k	pop_k ; $rw\,(\tau_{n-1}, \cdots, \tau_{k+1}, \tau_k, \tau'_{k-1}, \cdots, \tau'_0)$ where $top_1(pop_k(S)) = (a', (\tau'_{n-1}, \cdots, \tau'_0))$
$collapse$	$collapse$; $rw\,(\tau_{n-1}, \cdots, \tau_{k+1}, \tau_k, \tau'_{k-1}, \cdots, \tau'_0)$ where $top_1(collapse(S)) = (a', (\tau'_{n-1}, \cdots, \tau'_0))$

where δ is the transition function of the automaton $\mathcal{B}$.

Table I
ILLUSTRATIONS AND DEFINITIONS.

s. If $x \neq 0$, then $top_{k+1}(v_{x-1}) = s$. In particular one also has that $top_1(v_{x-1}) = top_1(s)$.

We now introduce the notion of k-ancestor. Fix a partial play $\Lambda = v_0 v_1 \cdots v_m$, let $v_m = (q, s)$ be some configuration in Λ and let s' be some internal k-stack in s. Then the *k-ancestor* of s' is the configuration v_i where i is the index of s' in the indexed version of v_m.

We now introduce the notion of *collapse rank*. Fix a partial play $\Lambda = v_0 v_1 \cdots v_m$ and assume that the top_1 element of v_m has a $(k+1)$-link for some k. Then the collapse rank in v_m is the smallest colour visited since the k-ancestor of the pointed k-stack.

Finally, we give a notion of *pop rank*. Fix a partial play $\Lambda = v_0 v_1 \cdots v_m$ and a configuration $v_m = (q, s)$ in Λ. Then the pop rank for k (for any $1 \leq k \leq n$), when defined, is the smallest colour visited since the k-ancestor of $pop_k(s)$. In particular, the pop rank for n is the smallest colour visited since the stack has height at least the height of s.

Consider a partial play $\Lambda = v_0 v_1 \cdots v_m$ in $\mathbb{G}$ ending in a configuration $v_m = (q, s)$ such that $top_1(s)$ has an n-link (if the link is a k-link for some $k < n$ the following concepts are not relevant). The *link ancestor* of v_m is the configuration v_j where the original copy of the n-link in $top_1(s)$ was created, or v_0 if the link was present in the stack of the configuration v_0. The *link rank* of v_m is the minimum colour of a state occurring in Λ between v_m and its link ancestor v_j (inclusive) *i.e.* $\min\{\rho(v_j), \cdots \rho(v_m)\}$.

Definition 1. An n-CPDA equipped with a colouring function is *rank-aware* from a configuration v_0 if there exist functions $ColRk : \Gamma \to \mathbb{N}$ and $LinkRk : \Gamma \to \mathbb{N}$ such that for any partial play $\Lambda = v_0 v_1 \cdots v_\ell$, the collapse rank and the link rank (if defined) of the configuration $v_\ell = (q, s)$ are respectively equal to $ColRk(top_1(s))$ and $LinkRk(top_1(s))$. In other words, the collapse rank and the link rank are stored in the top_1-element of the stack.

Remark 10. *In the current setting, if the collapse ancestor (*resp *the pop ancestor / the link ancestor) refers to a stack that was internal in the initial configuration (i.e. the k-ancestor is v_0) then the collapse rank (*resp *the pop rank / the link rank) is simply the smallest priory seen since the beginning of the play. Hence, it does not make much sense but it permits the construction to remain uniform.*

The next lemma shows that we can restrict our attention

to CPDA games where the underlying CPDA is rank-aware.

Lemma 8. *For any n-CPDA $\mathcal{A}$ and any parity game $\mathbb{G}$ on it, one can construct an n-CPDA $\mathcal{A}_{\mathrm{rk}}$ and an associated parity game $\mathbb{G}_{\mathrm{rk}}$ such that there exists a mapping ν_1 from the configurations of $\mathcal{A}$ to that of $\mathcal{A}_{\mathrm{rk}}$ satisfying the following conditions:*

- *for any configuration v_0 of $\mathcal{A}$, $\mathcal{A}_{\mathrm{rk}}$ is rank-aware from $\nu_1(v_0)$;*
- *Éloïse has a winning strategy in $\mathbb{G}$ from some configuration v_0 iff she has a winning strategy in $\mathbb{G}_{\mathrm{rk}}$ from $\nu_1(v_0)$.*
- *both ν_1 and ν_1^{-1} preserve regular sets of configurations;*

The proof is a non-trivial generalisation of [19, Lemma 6.3] (which concerns 2-CPDA) to the general setting of n-CPDA and starting from an arbitrary configuration, and it occupies the rest of the section.

Fix an n-CPDA $\mathcal{A} = \langle A, \Gamma, Q, \delta \rangle$, a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q and a colouring function $\rho : Q \to C \subset \mathbb{N}$. Denote by $\mathbb{G}$ the induced parity game. We define a rank-aware (to be proven) n-CPDA $\mathcal{A}_{\mathrm{rk}} = \langle \langle A, \Gamma_{\mathrm{rk}}, Q_{\mathrm{rk}}, \delta_{\mathrm{rk}} \rangle \rangle$ such that $Q \times C \subset Q_{\mathrm{rk}}$ and

$$\Gamma_{\mathrm{rk}} = \Gamma \times (C \cup \{\circlearrowleft\}) \times (C \cup \{\circlearrowleft, \dagger\}) \times (C^{\{1,\dots,n\}} \cup \{\circlearrowleft\})$$

The *main* configurations $((q,\theta),s)$ of $\mathcal{A}_{\mathrm{rk}}$ (by *main* we mean a configuration that is reached after simulating a transition of $\mathcal{A}$: indeed simulating one transition of $\mathcal{A}$ need several steps in $\mathcal{A}_{\mathrm{rk}}$, hence goes through intermediate configurations that we do not care about when stating our invariant) will satisfy the following invariant. First, θ is the minimal colour visited from the beginning of the path/run/play. Second, if $top_1(s) = (\alpha, m_c, m_l, \tau)$ then the following holds.

- m_c is the collapse rank.
- m_l is the link rank if it makes sense (*i.e.* there is an n-link in the current top symbol) or is $\dagger$ otherwise.
- τ is the *pop rank*, that is, for every $i = 1, \cdots, n$, $\tau(i)$ is the pop rank for i.

Let us now explain when one uses the $\circlearrowleft$ symbol and how ν_1 is defined. Let (q,s) be some configuration in $\mathcal{A}$. Then $\nu_1(q,s) = ((q,\rho(q),s')$ where s' is obtained by:

- Replacing every internal symbol γ (*i.e.* that is not the top_1-element) by $(\gamma, \circlearrowleft, \circlearrowleft, \circlearrowleft)$ if it has an n-link and by $(\gamma, \circlearrowleft, \dagger, \circlearrowleft)$ otherwise.
- Replacing the top_1 element γ by $(\gamma, \rho(q), \rho(q), \rho(q))$ if it has an n-link and by $(\gamma, \rho(q), \dagger, \rho(q))$ otherwise.

Hence at the beginning of the run the invariant holds.

The transition function of $\mathcal{A}_{\mathrm{rk}}$ mimics that of $\mathcal{A}$ and updates the ranks as explained below. First, let us explain the meaning of symbols $\circlearrowleft$. Such symbols will never be created using a $push_1^{-,k}$ action: hence they can only be duplicated (using $push_k$) from symbols originally in the stack. The meaning of a symbol $\circlearrowleft$ is that the corresponding object (collapse rank, link rank or pop rank) has not been yet settled. However, it can be very easily computed as it necessarily equals the smallest colour visited so far (as noted in Remark 10): this is why we made the computation of the minimal colour visited so far in the control state of $\mathcal{A}_{\mathrm{rk}}$.

In order to save space and to make the construction more understandable, we do not formally describe δ_{rk} but rather explain how $\mathcal{A}_{\mathrm{rk}}$ is supposed to behave. It should be clear that δ_{rk} can be formally described to fit this informal description (and that some extra control states are actually needed). Note that the following description also contains the inductive proof of its validity, namely that m_c, m_l and τ are as stated above. To avoid case distinction on whether the link rank is defined or not, we take the following convention that $\min(\dagger, i) = \dagger$ for every $i \in \mathbb{N}$.

Assume $\mathcal{A}_{\mathrm{rk}}$ is in some configuration $((q,\theta),s)$ with $top_1(s) = (\alpha, m_c, m_l, \tau)$ and let $v_0 v_1 \cdots v_\ell$ be the beginning of the run of $\mathcal{A}_{\mathrm{rk}}$ where we denote $v_i = ((q_i,\theta_i),s_i)$ (hence $q_\ell = q$ and $s_\ell = s$). The following behaviours are those allowed in $((q,\theta),s)$.

1) For every $\delta(q,\alpha,a) = (q', pop_k)$ with $1 \le k \le n$, let $pop_k(s) = s'$ and let $top_1(s') = (\alpha', m_c', m_l', \tau')$. Then on reading a, $\mathcal{A}_{\mathrm{rk}}$ goes to the configuration $((q',\theta'),s'')$ where $\theta' = \min(\theta, \rho(q'))$ and s'' is obtained from s' by replacing $top_1(s')$ by
   a) $(\alpha', \theta', \theta', (\theta', \dots, \theta'))$ if $m_c' = \circlearrowleft$, $m_l' = \circlearrowleft$ and $\tau' = (\circlearrowleft, \dots, \circlearrowleft)$.
   b) $(\alpha', \theta', \dagger, (\theta', \dots, \theta'))$ if $m_c' = \circlearrowleft$, $m_l' = \dagger$ and $\tau' = (\circlearrowleft, \dots, \circlearrowleft)$.
   c) $(\alpha', \min(m_c', \tau(k), \rho(q')), \min(m_l', \tau(k), \rho(q')), \tau'')$, with
   $$\tau''(i) = \begin{cases} \min(\tau'(i), \tau(k), \rho(q')) & \text{if } i \le k \\ \min(\tau(i), \rho(q')) & \text{if } i > k \end{cases}$$
   Cases (a) and (b) correspond to the case where one reach (possibly a copy) of a symbol that was in the stack from the very beginning and that never appeared as a top_1 element: then the value of the collapse rank, link rank (if defined this is case (a) otherwise it is case (b)) and pop ranks are all equal to θ'.
   We now explain case (c). Let v_x be the k-ancestor of $top_k(pop_k(s))$. Then $x > 0$ as we would be otherwise in case (a) or (b). By Proposition 6, it follows that $top_k(pop_k(s)) = top_k(s_{x-1})$, and by induction hypothesis, at step $(x-1)$, m_c', m_l' and τ' had the expected meaning. Let y be the index of the pointed stack in s': y is also the index of the pointed stack in s_{x-1}, and moreover $y < x$. The collapse rank in $v_{\ell+1}$ is $\min\{\rho(q_y), \dots, \rho(q_{x-1}), \rho(q_x) \dots, \rho(q_n), \rho(q')\} = \min\{m_c', \tau(k), \rho(q')\}$. Similarly, when defined, the link ancestor of s' is the same as the one in s_{x-1}: hence the pop rank in $v_{\ell+1}$ is $\min\{m_l', \tau(k), \rho(q')\}$.

For any $i \leq k$, $top_i(pop_i(s')) = top_i(s_{x-1})$ and therefore the pop rank for i in $v_{\ell+1}$ is obtained by updating $\tau'(i)$ to take care of the minimum colour seen since v_x which (as for the collapse rank) is $\min\{\tau(k), \rho(q')\}$: therefore the pop rank for i in $v_{\ell+1}$ equals $\min\{\tau'(i), \tau(k), \rho(q')\}$.

For any $i > k$, $pop_i(s') = pop_i(s)$ and thus $top_i(pop_i(s')) = top_i(pop_i(s))$. Therefore the pop rank for i in $v_{\ell+1}$ is obtained by updating the one in v_ℓ to take care of the new visited colour $\rho(q')$: hence the pop rank for i in $v_{\ell+1}$ equals $\min\{\tau(i), \rho(q')\}$.

2) For every $\delta(q, \alpha, a) = (q', push_j)$ with $2 \leq j \leq n$, let $push_j(s) = s'$ and let $top_1(s') = (\alpha, m_c, m_l, \tau)$ (note that $\circlearrowleft$ does not appear in $top_1(s')$). Then, on reading a, $\mathcal{A}_{\text{rk}}$ can go to the configuration $((q', \theta'), s'')$ where $\theta' = \min(\theta, \rho(q'))$ and s'' is obtained from s' when replacing $top_1(s')$ by $(\alpha, \min(m_c, \rho(q')), \min(m_l, \rho(q')), \tau')$ with

$$\tau'(i) = \begin{cases} \min(\tau(i), \rho(q')) & \text{if } i \neq j \\ \rho(q') & \text{if } i = j \end{cases}$$

Indeed, the ancestor of the pointed stack in the new configuration is the same as the one in s. As by induction hypothesis m_c is the collapse rank in v_ℓ, the collapse rank in $v_{\ell+1}$ is obtained by updating m_c to take care of the new visited colour, namely by taking $\min\{m_c, \rho(q')\}$. Similarly, if defined, the link ancestors in v_ℓ and $v_{\ell+1}$ are identical and then the link rank in $v_{\ell+1}$ is $\min\{m_c, \rho(q')\}$.

For any $i \neq j$, as $pop_i(s) = pop_i(s')$, the i-ancestor of $top_i(pop_i(s)')$ and the i-ancestor of $top_i(pop_i(s'))$ are the same. Again using the induction hypothesis one directly gets that the pop rank for i in $v_{\ell+1}$ equals $\min\{\tau(i), \rho(q')\}$.

The index of the j-ancestor of $top_j(pop_j(s'))$ is by definition $\ell + 1$. Hence as the only colour visited since $v_{\ell+1}$ is $\rho(q')$ it equals the pop rank for j.

3) For every $\delta(q, \alpha, a) = (q', push_1^{\beta,k})$ with $1 \leq k \leq n$, and $\beta \in (\Gamma \setminus \{\bot\})$, on reading a, $\mathcal{A}_{\text{rk}}$ goes to (q', θ'), where $\theta' = \min(\theta, \rho'(q'))$, and applies $push_1^{(\beta, m_c', m_l', \tau'), k}$ where $m_c' = \min(\tau(k), \rho(q'))$, $m_l' = \rho(q')$ if $k = n$ and $m_l' = \dagger$ otherwise, and $\tau'(i) = \min(\tau(i), \rho(q'))$ for every $i \geq 2$ and $\tau(1) = \rho(q')$.

Indeed, the pointed stack in s' is $top_k(pop_k(s))$ and therefore the collapse rank in $v_{\ell+1}$ is the minimum of the pop rank for k in s and of the new visited colour $\rho(q')$, that is $\min\{\tau(k), \rho(q')\}$.

If $k = n$, the link ancestor of $v_{\ell+1}$ is $v_{\ell+1}$ itself and hence the link rank is the colour of the current configuration, namely $\rho(q')$.

For any $i \geq 2$, as $pop_i(s) = pop_i(s')$ one also has $top_i(pop_i(s')) = top_i(pop_i(s))$ and therefore the pop rank for i in $v_{\ell+1}$ equals the minimum of the one in v_ℓ with the new visited colour $\rho(q')$, that is $\min\{\tau(i), \rho(q')\}$. Finally as the 1-ancestor of $pop_1(s')$ is v_ℓ then the pop rank for 1 is the current colour, namely $\rho(q')$.

4) For every $\delta(q, \alpha, a) = (q', collapse)$, let $collapse(s) = s'$ and let $top_1(s') = (\alpha', m_c', m_l', \tau')$. Then, on reading a, $\mathcal{A}_{\text{rk}}$ goes to the configuration $((q', \theta'), s'')$ where $\theta' = \min(\theta, \rho(q'))$ and s'' is obtained from s' by replacing $top_1(s')$ by
   a) $(\alpha', \theta', \theta', (\theta', \ldots, \theta'))$ if $m_c' = \circlearrowleft$, $m_l' = \circlearrowleft$ and $\tau' = (\circlearrowleft, \ldots, \circlearrowleft)$.
   b) $(\alpha', \theta', \dagger, (\theta', \ldots, \theta'))$ if $m_c' = \circlearrowleft$, $m_l' = \dagger$ and $\tau' = (\circlearrowleft, \ldots, \circlearrowleft)$.
   c) $(\alpha', \min(m_c', m_c, \rho(q')), \min(m_l', m_c, \rho(q')), \tau'')$ with

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

Cases (a) and (b) correspond to the case where one reach (possibly a copy) of a symbol that was in the stack from the very beginning and that never appeared as a top_1 element: then the value of the collapse rank, link rank (if defined this is case (a) otherwise it is case (b)) and pop ranks are all equal to θ'.

We now explain case (c). Let x be the index of the pointed stack in (q, s). Then $x > 0$ as we would be otherwise in case (a) or (b). By induction hypothesis, m_c' and τ' give the collapse rank / link rank / pop ranks in v_{x-1}. Moreover the k-ancestor of the target of the top link in s' is the same as the one in v_{x-1}. Therefore the collapse rank is obtained by taking the minimum of the one in v_{x-1} with $\min\{\rho(q_x), \ldots \rho(q_n), \rho(q')\} = \min\{m_c, \rho(q')\}$. Similarly (if defined) the link ancestor in s' being the same as the one in v_{x-1}, the link rank is obtained by taking the minimum of the one in v_{x-1} with $\min\{\rho(q_x), \ldots \rho(q_n), \rho(q')\} = \min\{m_c, \rho(q')\}$.

Let $i \leq k$. The i-ancestor of $top_i(pop_i(s'))$ is the same as the i-ancestor of $top_i(pop_i(s_{x-1}))$. Therefore the pop rank for i in $v_{\ell+1}$ is obtained by taking the minimum of the one in v_{x-1} with $\min\{\rho(q_x), \ldots \rho(q_n), \rho(q')\} = \min\{m_c, \rho(q')\}$.

Let $i > k$. Then the i-ancestor of $top_i(pop_i(s'))$ is the same as the i-ancestor of $top_i(pop_i(s_n))$: indeed the collapse also modified that top_k stack. Therefore the pop rank for i in $v_{\ell+1}$ is obtained by taking the minimum of the one in v_ℓ with the new visited colour $\rho(q')$.

5) For every $\delta(q, \alpha, a) = (q', rew_1^\beta)$ with $\beta \in (\Gamma \setminus \{\bot\})$, on reading a $\mathcal{A}_{\text{rk}}$ goes in state (q', θ') where $\theta' = \min(q, \theta)$ and applies $rew_1^{(\beta, m_c', m_l', \tau')}$ where $m_c' = \min(m_c, \rho(q'))$, $m_l' = \min(m_l, \rho(q'))$ and $\tau'(k) = \min(\tau(k), \rho(q'))$ for every $1 \leq k \leq n$, if

we let $top_1(s) = (\alpha, m_c, m_l, \tau)$.
This case is trivial as we just need to update all information by considering the colour of the new control state.

From the previous description (and the included inductive proof) we conclude that, for any configuration v_0 of $\mathcal{A}$, $\mathcal{A}_{\rm rk}$ is rank-aware from $\nu_1(v_0)$.

Remark 11. *Note that building a rank-aware n-CPDA from a non-aware one increases the stack alphabet by C^{n+3} and the state set by C^{n+1} (recall that we need extra states, that where hidden in the previous description, mainly to store τ).*

Now, in order to proof the second point of Lemma 8, one considers the parity game $\mathbb{G}_{\rm rk}$ on $\mathcal{A}_{\rm rk}$ defined by

- using a similar partition as the one in $\mathbb{G}$ from Q: $Q_{\rm rk_{\mathbf{E}}} = Q_{\mathbf{E}} \times C$ (the control states in $Q_{\rm rk} \setminus Q \times C$ inducing configurations with exactly one successor can be controlled by any player),
- and extending ρ to $Q_{\rm rk}$ by letting $\rho((q,\theta)) = \rho(\theta)$ and assigning the maximal colour to states in $Q_{\rm rk} \setminus Q \times C$ (hence not modifying the winner).

It is immediate that Éloïse has a winning strategy in $\mathbb{G}$ from some configuration v_0 iff she has a winning strategy in $\mathbb{G}_{\rm rk}$ from $\nu_1(v_0)$.

Finally, the fact that both ν_1 and ν_1^{-1} preserve regular sets of configurations is immediate: for this one basically needs to simulate an automaton on the image by ν_1 (or ν_1^{-1}) that can be computed on-the-fly (except for the very last steps of ν_1 where one needs to know the control state before deducing the top_1 stack element as it as information on the colour of the control state. However, this is not a problem to have a slight — finite — delay in the final steps of the simulation).

D. Second step: from $\mathbb{G}_{\rm rk}$ to $\mathbb{G}_{\rm lf}$. Removing the n-links

Let $\mathcal{A}_{\rm rk} = \langle A_{\rm rk}, \Gamma_{\rm rk}, Q_{\rm rk}, \delta_{\rm rk} \rangle$ be the (rank-aware) n-CPDA generating the game $\mathbb{G}_{\rm rk}$ obtained in the previous step

Consider the following informal description of a new game $\mathbb{G}_{\rm lf}$ defined from $\mathbb{G}_{\rm rk}$. The new games mimics $\mathbb{G}_{\rm rk}$ 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 $\overrightarrow{R} = (R_0, \cdots R_d) \in (2^{Q_{\rm rk}})^{d+1}$ — here $Q_{\rm rk}$ are the control states of $\mathcal{A}_{\rm rk}$ — 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 which is winning for Éloïse iff $q \in R_i$ where i is the *link rank* found on the current top_1-element and which corresponds to the smallest colour visited since the original copy of symbol $(\gamma, \overrightarrow{R})$ was pushed onto the stack (recall that $\mathcal{A}_{\rm rk}$ 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.

There are now two tasks. The first one is to prove that the previous simulation game can be generated by an n-CPDA with the extra property that it never creates n-links. The second one is to prove that this game correctly simulates the original one (*i.e.* Éloïse wins in $\mathbb{G}_{\rm rk}$ from some vertex v iff she wins in the $\mathbb{G}_{\rm lf}$ from the configuration $\nu_2(v)$ for some mapping ν_2 — to be defined — transforming vertices of the first game into vertices of the second one). The first task (see Section D1) is simple as the initial n-CPDA defining $\mathbb{G}_{\rm rk}$ is rank aware and therefore comes with a function $LinkRk$ as in Lemma 8. The second task (see Section D2) is more involved because we have to define ν_2 and to prove that it preserves (arbitrary) winning configurations.

1) The simulation game: $\mathbb{G}_{\rm lf}$: As in $\mathbb{G}_{\rm rk}$ the player that control the current vertex has to make a move, *i.e.* apply a transition of the n-CPDA. In case this player wants to simulate a transition going to q and performing a $push_1^{\gamma,n}$ action on the stack, the play goes as follows.

- The CPDA goes in a new control state q^γ and no operation on the stack is performed.
- From q^γ, Éloïse has to move to a new control state $q^?$ and can push any symbol of the form $(\gamma, \overrightarrow{R})$ where $\overrightarrow{R} = (R_0, \cdots R_d) \in (2^Q)^{d+1}$. Here we assume that the symbol comes with no links (alternatively, we could add a dummy link and state that no collapse can be performed from a configuration with top_1 element being of the form $(\gamma, \overrightarrow{R})$.
- From $q^?$, Abelard has to play and choose one of the two possible options: either go to state q and perform no action on the stack, or pick any state q in some R_i, go to an intermediate new state q^i (of colour i) without changing the stack and from this new configuration go to state q and finally perform a pop_n action.

The intended meaning of such a decomposition of the $push_1^{\gamma,n}$ operation is the following: when choosing the sets in $\overrightarrow{R}$, Éloïse is claiming that she has a strategy such that if the n-link created by pushing γ is eventually used for

collapsing the stack then the control state after collapsing will belong to R_i where i is meant to be the smallest colour from the creation of the link to the collapse of the stack (equivalently it will be the link rank — as computed in $\mathcal{A}_{\rm rk}$ — the just before collapsing). Note that R_i are sets because of the fact that we have a game and hence Éloïse has not a full control of the play. Then Abelard is offered to simulate the collapse (here state q^i is only used for going through a state of colour i). If he does not want to simulate a collapse then one stores $\overrightarrow{R}$ for possibly checking its truth later.

Later, in case some player wants to simulate a collapse transition involving an n-link and going to state q, the top_1 element is necessarily of the form $(\gamma, \overrightarrow{R})$. Then the simulation is done by going to a dead-end vertex that is winning for Éloïse iff $q \in R_{LinkRk(\gamma)}$, *i.e.* Éloïse wins iff her former claim on $\overrightarrow{R}$ was correct.

For all other kind of transitions (pop_k, $push_k$, rew or $collapse$ involving $< n$ links), the simulation is immediate (either the top element is a single letter from Γ or it is an element from $\Gamma \times (2^Q)^{d+1}$ in which case one simply ignores the $(2^Q)^{d+1}$ component).

Formally we set $\mathcal{A}_{\rm lf} = \langle\, A_{\rm lf}, \Gamma_{\rm lf}, Q_{\rm lf}, \delta_{\rm lf} \,\rangle$ with

- $A_{\rm lf} = A_{\rm rk} \cup (2^{Q_{\rm rk}})^{d+1} \cup \{q^i \mid q \in Q_{\rm rk},\ 0 \le i \le d\} \cup \{go, \#, \mathit{ff}\}$
- $\Gamma_{\rm lf} = \Gamma_{\rm rk} \cup \Gamma_{\rm rk} \times (2^{Q_{\rm rk}})^{d+1}$
- $Q_{\rm lf} = Q_{\rm rk} \cup \{q^a \mid q \in Q_{\rm rk},\ a \in \Gamma_{\rm rk}\} \cup \{q^? \mid q \in Q_{\rm rk}\} \cup \{q^i \mid q \in Q_{\rm rk},\ 0 \le i \le d\} \cup \{\#, \mathit{ff}\}$
- $\delta_{\rm lf}$ is defined as follows.
  - $\forall q \in Q_{\rm rk},\ \forall \gamma \in \Gamma_{\rm rk},\ \forall a \in A$ if $\delta(q, \gamma, a) = (q', op)$ is such that op is neither of the form $push_1^{\beta,n}$ nor rew_1^β nor a collapse using an n-link, then $\delta(q, \gamma, a) = \delta(q, (\gamma, \overrightarrow{R}), a) = (q', op)$.
  - $\forall q \in Q_{\rm rk},\ \forall \gamma \in \Gamma_{\rm rk},\ \forall a \in A$ if $\delta(q, \gamma, a) = (q', rew_1^\beta)$, then $\delta(q, \gamma, a) = (q', rew_1^\beta)$ and $\delta(q, (\gamma, \overrightarrow{R}), a) = (q', rew_1^{(\beta, \overrightarrow{R})})$.
  - $\forall q \in Q_{\rm rk},\ \forall \gamma \in \Gamma_{\rm rk},\ \forall a \in A$ if $\delta(q, \gamma, a) = (p, push_1^{\beta,n})$ then $\delta(q, \gamma, a) = \delta(q, (\gamma, \overrightarrow{R}), a) = (p^\beta, id)$
  - $\forall p \in Q_{\rm rk},\ \forall \gamma \in \Gamma_{\rm lf}, \forall \overrightarrow{R} \in (2^{Q_{\rm rk}})^{d+1},\ \delta(p^\beta, \alpha, \overrightarrow{R}) = (p^?, push_1^{(\beta, \overrightarrow{R}),1})$. Note here that we create a 1-link as we do not permit in the definition of n-CPDA to push symbol without link: however this link being never used, one can safely forget it.
  - $\forall p \in Q_{\rm rk},\ \forall \beta \in \Gamma_{\rm rk}, \forall \overrightarrow{R} \in (2^{Q_{\rm rk}})^{d+1}$, $\delta(p^?, (\beta, \overrightarrow{R}), go) = (p, id)$.
  - $\forall p \in Q_{\rm rk},\ \forall \beta \in \Gamma_{\rm rk}, \forall \overrightarrow{R} \in (2^{Q_{\rm rk}})^{d+1}, \forall q^i \in A_{\rm lf}$, $\delta(p^?, (\beta, \overrightarrow{R}), q^i) = (q^i, id)$ provided $q \in R_i$ (otherwise δ is undefined).
  - $\forall q^i \in Q_{\rm lf},\ \forall (\beta, \overrightarrow{R}) \in \Gamma_{\rm lf},\ \delta(q^i, (\beta, \overrightarrow{R}), \varepsilon) = (q, pop_n)$.
  - $\forall q \in Q_{\rm rk},\ \forall (\alpha, \overrightarrow{R}) \in \Gamma_{\rm lf}$, if there exists some $a \in A_{\rm rk}$ such that $\delta(q, \alpha, a) = (p, collapse)$ and $p \in R_{LinkRk(\alpha)}$ then $\delta(q, (\alpha, \overrightarrow{R}), \#) = (\#, id)$ (otherwise δ is undefined).
  - $\forall q \in Q_{\rm rk},\ \forall (\alpha, \overrightarrow{R}) \in \Gamma_{\rm lf}$, if there exists some $a \in A_{\rm rk}$ such that $\delta(q, \alpha, a) = (p, collapse)$ and $p \notin R_{LinkRk(\alpha)}$ then $\delta(q, (\alpha, \overrightarrow{R}), \mathit{ff}) = (\mathit{ff}, id)$ (otherwise δ is undefined).

In order to define a game graph $\mathcal{G}_{\rm lf}$ out of $\mathcal{A}_{\rm lf}$ we let $Q_{{\rm lf}\mathbf{E}} = Q_{{\rm rk}\mathbf{E}} \cup \{q^\gamma \mid q \in Q_{\rm rk},\ \gamma \in \Gamma_{\rm rk}\} \cup \{q^i \mid q \in Q_{\rm rk},\ 0 \le i \le d\} \cup \{\mathit{ff}\}$. Finally to define a corresponding n-CPDA parity game $\mathbb{G}_{\rm lf}$ we extend ρ by letting $\rho(q^\gamma) = \rho(q^?) = d$ for any $q \in Q$ (has one cannot loop forever in such states, it means that they have no influence on the parity condition) and $\gamma \in \Gamma$ and $\rho(q_i) = i$ for every $0 \le i \le d$.

Note that $\mathcal{A}_{\rm lf}$ never create n-link, hence the game $\mathbb{G}_{\rm lf}$ is as expected.

2) Correctness of the simulation: Consider some configuration $v = (p, s)$ in $\mathbb{G}_{\rm rk}$. We explain now how to define an "equivalent" configuration $\nu_2(v)$ in $\mathbb{G}_{\rm lf}$ (here equivalent is in the sense of Theorem 6). The transformation simply replace any occurrence of a stack letter (call it γ) with an n-link in s by another letter of the form $(\gamma, \overrightarrow{R})$. Let s' be the stack obtained by popping every element above γ, and let $S = \{q \mid$ Éloïse wins in $\mathbb{G}_{\rm rk}$ from $(q, collapse(s'))\}$. Then one sets $\overrightarrow{R} = (R, \cdots, R)$.

Example 7. *Assume we are playing a two-colour parity game. Let*

$$s = [[[\,a\,]]\,[[]\overbrace{[\,a\,b\,c\,]]}\,[[]\overbrace{[\,a\,b\,c\,]]}]],$$

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

$$\nu_2(s) = [[[\,a\,]]\,[[]\overbrace{[\,a\,b\,(c, \overrightarrow{R})\,]]}\,[[]\overbrace{[\,a\,b\,(c, \overrightarrow{R})\,]]}].$$

In this section we prove the following result stating the validity of the previous construction:

Theorem 6. *Éloïse wins in* $\mathbb{G}_{\rm rk}$ *from some configuration* v *if and only if she wins in* $\mathbb{G}_{\rm lf}$ *from* $\nu_2(v)$.

Proof: In the following, we intensively work with strategy (both in $\mathbb{G}_{\rm rk}$ and $\mathbb{G}_{\rm lf}$). When defining the value of such a strategy on some partial play we may alternatively define it as a vertex (which respects the definition of a strategy as we gave it) or as a pair (q, op) formed by a control state and a stack operation. In the latter, one has to understand this as the strategy that goes to the configuration $(q, op(s))$ if s denotes the stack in the current configuration.

Assume that the configuration $v = (p_0, s)$ is winning for Éloïse in $\mathbb{G}_{\rm rk}$, and let Φ be a winning strategy for her. Using Φ, we define a strategy φ for Éloïse in $\mathbb{G}_{\rm lf}$ from $\nu_2(v)$. The strategy φ stores a partial play in $\mathbb{G}_{\rm rk}$, that is an element in $V_{\rm rk}^*$ (where $V_{\rm rk}$ denotes the set of vertices of $\mathcal{G}$). This memory will be denoted Λ. At the beginning Λ is initialized to the vertex $(p_0, \bot)$. At any moment in the play, if the

current vertex has top_1 symbol α and control state p, then the last vertex of Λ has control state p and top_1 symbol α or $(\alpha, \overrightarrow{R})$ (in this case there is an n-link from the top_1 symbol).

We first describe φ, and then we explain how Λ is updated.

Choice of the move. Assume that the play is in some vertex (p, s) with $p \in Q_{\mathbf{E}}$. The move given by Φ depends on $\varphi(\Lambda)$ (we shall later argue that Φ is well defined while proving that it is winning):

- If $\Phi(\Lambda) = (q, op)$, with op being some pop_k, $push_k$, $push_1^{\alpha,i}$ with $i < n$, or $collapse$ involving an $< n$-link then Éloïse goes to $(q, op(s))$.
- If $\Phi(\Lambda) = (q, rew_1^\alpha)$, then Éloïse goes to $(q, rew_1^\alpha(s))$ if $top_1(s) = b$ for some $b \in \Gamma$ and she goes to $(q, rew_1^{(\alpha,\overrightarrow{R})}(s))$ if $top_1(s) = (\beta, \overrightarrow{R})$.
- If $\Phi(\Lambda) = (q, collapse)$ then Éloïse goes to $(\#\!\!\#, s)$. We shall later see that this move is always valid
- If $\Phi(\Lambda) = (q, push_1^{\alpha,n})$ then Éloïse goes to (q^α, s).

In this last case, or in the case where $p \in Q_{\mathbf{A}}$ and Abelard goes to (q^α, s), we also have to explain how Éloïse behaves from (q^α, s).

She has to provide a vector $\overrightarrow{R} \in \mathcal{P}(Q)^{d+1}$ that describes which states can be reached if the n-link created by pushing α (or a copy of it) is used for collapsing the stack, depending on the smallest visited colour in the meantime. In order to define $\overrightarrow{R}$, Éloïse considers the set of all possible continuations of $\Lambda \cdot (q, push_1^{\alpha,n}(\sigma))$ (where (p, σ) denotes the last vertex of Λ) where she respects her strategy Φ. For each such play, she checks whether some configuration of the form $(r, pop_n\sigma)$ is eventually reached by collapsing using (possibly a copy of the) n-link created by $push_1^{\alpha,n}$. If it is the case, she considers the smallest colour i visited from the moment where the link was created to the moment collapse is performed (*i.e.* the link rank before collapsing). For every $i \in \{0, \dots d\}$, R_i, is defined to be the set of states $r \in Q$ such that the preceding case happens. More formally,

$$R_i = \{r \mid \exists\, \Lambda \cdot (q, push_1^{\alpha,n}(\sigma))v_0 \cdots v_k \cdot v_{k+1} \cdots \text{ play in } \mathbb{G} \text{ where Éloïse respects } \Phi \text{ and s.t. } v_{k+1} = (r, pop_n(\sigma)) \text{ is obtained by applying a collapse from } v_k, \text{ and configuration } (q, push_1^{\alpha,n}(\sigma)) \text{ is the link ancestor of } v_k\}$$

Finally, we set $\overrightarrow{R} = (R_0, \dots, R_d)$ and Éloïse moves to $(q^?, push_1^{(\alpha,\overrightarrow{R}),1}(s))$

Update of Λ. The memory Λ is updated after each visit to a configuration with a control state in Q. We have five cases depending on the transition:

- If the transition is of the form (q, op) with op being some pop_k, $push_k$, $push_1^{\alpha,i}$ with $i < n$, or $collapse$ involving an $< n$-link, then we extend Λ by applying the same transition. That is, if (p, σ) denotes the last configuration in Λ, then the updated memory is $\Lambda \cdot (q, op(\sigma))$.
- If the transition is of the form (q, rew_1^α) or $(q, rew_1^{(\alpha,\overrightarrow{R})})$ and if (p, σ) denotes the last configuration in Λ, then the updated memory is $\Lambda \cdot (q, rew_1^\alpha(\sigma))$.
- If the transition is of the form $(\#\!\!\#, id)$ or (ff, id), the play is in a dead end. Therefore Λ needs not to be updated.
- If the last transitions form a sequence of the form $(q^\alpha, id) \cdot (q^?, push_1^{(\alpha,\overrightarrow{R}),1}) \cdot (q, id)$, then the updated memory is $\Lambda \cdot (q, push_1^{\alpha,n}(\sigma))$, where (p, σ) denotes the last configuration in Λ.
- If the last transitions form a sequence of the form $(q^\alpha, id) \cdot (q^?, push_1^{(\alpha,\overrightarrow{R}),1}) \cdot (r^i, id) \cdot (r, pop_n)$, then we extend Λ by a sequence of actions (consistent with Φ) that starts by performing transition $(q, push_1^{\alpha,n})$ and ends up by collapsing (possibly a copy of) the link created at this first step and goes to state r while visiting i as a minimal colour in the meantime. By definition of $\overrightarrow{R}$ such a sequence always exists. More formally, if (p, σ) denotes the last configuration in Λ, then the updated memory is now a play in $\mathbb{G}_{\text{rk}}$, $\Lambda \cdot (q, push_1^{\alpha,n}(\sigma))v_0 \cdots v_k \cdot v_{k+1}$, where Éloïse respects Φ and such that $v_{k+1} = (r, pop_n(\sigma))$ is obtained by applying a collapse from v_k, and configuration $(q, push_1^{\alpha,n}(\sigma))$ is the link ancestor of v_k.

Therefore, with any partial play λ in $\mathbb{G}_{\text{lf}}$ in which Éloïse respects her strategy φ, is associated a partial play Λ in $\mathbb{G}_{\text{rk}}$. An immediate induction shows that Éloïse respects Φ in Λ. The same arguments works for an infinite play λ, and the corresponding play Λ is therefore infinite, starts from $\nu_2(p_0, s)$ and Éloïse respects Φ in that play. Therefore it is a winning play.

Moreover, if λ is an infinite play, it easily follows from the definitions of φ and Λ that the smallest infinitely visited colour in λ is the same as the one in Λ. Hence, any infinite play in $\mathbb{G}_{\text{lf}}$ starting from $\nu_2(p_0, \bot_n)$ where Éloïse respects φ is winning from her.

Now, considering finite plays (*i.e.* plays ending either in state $\#\!\!\#$ or ff). Reaching a dead-end is necessarily by simulating a collapse from some configuration with top_1 of the form $(\alpha, \overrightarrow{R})$. We should distinguish between those elements $(\alpha, \overrightarrow{R})$ that are "created" before (*i.e.* by the nu_2 function) or during the play (by Éloïse). For the second ones, one may note that whenever Éloïse wants to simulate a collapse, she can safely goes to state $\#\!\!\#$ (meaning φ is well defined): indeed, if this was not the case, it would contradict the way $\overrightarrow{R}$ was defined when simulating the original creation of the link. For the same reason, Abelard can never reach state ff provided Éloïse respects her strategy φ. Now consider an element $(\alpha, \overrightarrow{R})$ created by ν_2 and assume that one player wants to simulate a collapse from some configuration with such a top_1 element. Call λ the partial play just before and call Λ the associated play in $\mathbb{G}$. Then in Λ, Éloïse respects her winning strategy Φ. If she has to play next in Λ, strategy Φ indicates to collapse; if it is Abelard's turn to play it can collapse. In both case, it

means that the configuration that is reached after collapsing is winning for Éloïse (it is a configuration visited in a winning play). Hence its control state belongs to R where $\overrightarrow{R} = (R, \cdots, R)$ by definition of ν_2 and therefore, from the current vertex in $\mathbb{G}_{\text{lf}}$, there are no transition to *ff* and there is at least one to *tt*. Therefore finite plays in $\mathbb{G}_{\text{lf}}$ are won by Éloïse (provided she respects φ).

Hence, any finite play in $\mathbb{G}_{\text{lf}}$ starting from $\nu_1(p_0, s)$ where Éloïse respects φ is winning from her.

Altogether, it proves that φ is a winning strategy for Éloïse in $\mathbb{G}_{\text{lf}}$ from $(p_{in}, \bot_n)$.

Let us now prove the converse implication Assume that the configuration $\nu_2((p_0, s))$ is winning for Éloïse in $\mathbb{G}_{\text{lf}}$, and let φ be a winning strategy for her. Using φ, we define a strategy Φ for Éloïse in $\mathbb{G}_{\text{rk}}$ from (p_0, s). Recall how $\nu_2((p_0, s))$ is defined: it replaces every symbol α in s with an n-link by a pair $(\alpha, (R, \ldots, R))$ where R is the collection of states r such that Éloïse wins from (r, s') where s' is the stack obtained by removing every symbol (and stacks) above α and then performing collapse. We can therefore assume that we have a collection of winning strategies for all those configurations (r, s'). Then, during a play where Éloïse respects Φ, if one eventually visits such a configuration (r, s'), the strategy Φ will mimic the winning strategy from that point and therefore the resulting play will be winning for Éloïse. Then in the rest of this description we mainly deal with the case of plays where this phenomenon is not happening.

The strategy Φ stores a partial play in $\mathbb{G}_{\text{lf}}$. This memory will be denoted λ. At the beginning λ is initialized to the vertex $\nu_2((p_0, s))$. At any moment in the play, if the current vertex has top_1 symbol α and control state p, then the last vertex of Λ has control state p and top_1 symbol α. At any moment in the play, if the current vertex has top_1 symbol $(\alpha, \overrightarrow{R}))$ and control state p, then the last vertex of Λ has control state p and top_1 symbol α with an n-link; moreover if (possibly a copy of) this link is eventually used in a collapse, then the state that will be reached after collapsing will belongs to R_i where i would be the link rank just before collapsing.

We first describe Φ and then we explain how λ is updated. Recall that we switch to a known winning strategy in case we apply a collapse from (possibly a copy of) an n-link that was already in s_0.

Choice of the move. Assume that the play is in some vertex (p, s) with $p \in Q_{\mathbf{E}}$. The move given by Φ depends on $\varphi(\Lambda)$ (we shall later argue that φ is well defined while proving that it is winning):

- If $\varphi(\Lambda) = (q, op)$, with op being some pop_k, $push_k$, $push_1^{\alpha,i}$ with $i < n$, or $collapse$ (necessarily involving an $< n$-link) then Éloïse goes to $(q, op(s))$.
- If $\varphi(\Lambda) = (q, rew_1^{\alpha})$ or $\varphi(\Lambda) = (q, rew_1^{(\alpha, \overrightarrow{R})})$ then Éloïse goes to $(q, rew_1^{\alpha}(s))$.
- If $\varphi(\Lambda) = (\#, id)$ then Éloïse goes to $(r, collapse)$ for some $r \in R_i$ where we let σ be the stack in the last configuration of Λ, $top_1(\sigma) = (\alpha, \overrightarrow{R})$ and $i = LinkRk(\alpha)$. Note that in this case, the collapse involves an n-link.
- If $\varphi(\Lambda) = (q^{\alpha}, id)$ then Éloïse goes to $(q, push_1^{\alpha,n}(s))$.

Update of λ. The memory λ is updated after each move (by any player). We have four cases depending on the last transition:

- If the transition is of the form (q, op) with op being some pop_k, $push_k$, $push_1^{\alpha,i}$ with $i < n$, or $collapse$ involving an $< n$-link, then we extend λ by applying the same transition. That is, if (p, σ) denotes the last configuration in λ, then the updated memory is $\lambda \cdot (q, op(\sigma))$.
- If the transition is of the form (q, rew_1^{α}) then we extend λ by mimicking the same transition. That is, if (p, σ) denotes the last configuration in λ, then the updated memory is $\lambda \cdot (q, rew_1^{\alpha}(\sigma))$ if $top_1(s) = b\beta$ for some $\beta \in \Gamma$ and it is $\lambda \cdot (q, rew_1^{(\alpha, \overrightarrow{R})}(\sigma))$ if $top_1(s) = (\beta, \overrightarrow{R})$.
- If the transition is of the form $(q, push_1^{\alpha,n})$ then, if (p, σ) denotes the last configuration in λ, the updated memory is $\lambda \cdot (q^{\alpha}, \sigma) \cdot (q^{?}, push_1^{(\alpha, \overrightarrow{R}),1}(\sigma)) \cdot (q, push_1^{(\alpha, \overrightarrow{R}),1}(\sigma))$ where $\varphi(\lambda \cdot (q^{\alpha}, \sigma)) = (q^{?}, push_1^{(\alpha, \overrightarrow{R}),1}(\sigma))$.
- If the last transition is of the form $(r, collapse)$ involving an n-link, then we have two cases. Either the collapse was following (possibly a copy of) an n-link that was already in s_0 in case we claim (and prove later) that we end up in a winning configuration and switch to a corresponding winning strategy as already explained. Either we follow an n-link that was created during the play and we let $\lambda = v_0 \cdots v_m$ and let v_i be the link ancestor of v_m [6]. Then the updated memory is obtained by backtracking inside λ until reaching the configuration where the (simulation of the) collapsed n-link was created (this configuration being the link ancestor) and then extend it by a choice of Abelard consistent with the collapse. That is, if $\lambda = v_0 \cdots v_m$, and if $v_i = (p^{?}, \sigma)$ is the link ancestor of v_m, then the updated memory is $v_0 \cdots v_i \cdot (r^i, pop_n(\sigma)) \cdot (r, pop_n(\sigma))$ where i denotes the link rank in the configuration Λ was just before collapsing.

Therefore, with any partial play Λ in $\mathbb{G}_{\text{rk}}$ in which Éloïse respects her strategy φ, is associated a partial play λ in $\mathbb{G}_{\text{lf}}$. Note that if we end up in a configuration that is known to be winning, λ is no longer extended. This also

[6] Here we implicitly extends the notion of link ancestor as follows. In $\mathbb{G}_{\text{lf}}$ instead of creating n-link one pushes symbol of the form $(\alpha, \overrightarrow{R})$: hence whenever doing a $push_1^{(\alpha, \overrightarrow{R}),1}$ one attached to the vector $\overrightarrow{R}$ the index of the current configuration. Then if the top_1 element of v_n is some $(\alpha, \overrightarrow{R})$ then the link ancestor of v_m is defined to be v_i where i is the indexed attached with $\overrightarrow{R}$. Note in particular that the control state in the link ancestor is of the form $p^{?}$.

implies that when collapsing an n-link that was already in s_0 one necessarily ends up in a winning configuration. Indeed assume the contrary and let λ be the constructed play before collapsing: then either Éloïse has to play and therefore moves to $\#$ (and therefore the configuration in Λ after collapsing is winning by definition of ν_2, leading a contradiction) of Abelard could move to *ff* (leading a contradiction with φ being winning). Therefore from now on we restrict our attention to the case where the n-links (and their copies) in s_0 are never used to collapse.

An immediate induction shows that Éloïse respects φ in λ. The same arguments works for an infinite play Λ, and the corresponding play λ is therefore infinite, starts from $(p_{in}, \bot_n)$ and Éloïse respects φ in that play. Therefore it is a winning play.

Now, in order to conclude that any play Λ in $\mathbb{G}_{\mathrm{rk}}$ in which Éloïse respects her strategy φ is winning for her, one needs to relate the sequence of colours in Λ and in λ. For this, we introduce a notion of factorisation of a partial play in $\Lambda = v_0 v_1 \cdots v_m$ (we should later note that it trivially extend to infinite plays). A factor will be a nonempty sequence of vertices of the following kind:

(1) it is a sequence $v_h \cdots v_k$ such that the transition from v_{h-1} to v_h is a $push_1^{n,\alpha}$, the transition from v_{k-1} to v_k is collapse involving an n-link, and v_h is the link ancestor of v_k.
(2) or it is a single vertex;

Then the factorisation of Λ denoted $Fact(\Lambda)$ is a sequence of factors inductively defined as follows (we bracket factors to make them explicit):

$Fact(\Lambda) = [v_0 \cdots v_k], Fact(v_{n+1} \cdots v_n)$ if there exists some k such that $v_0 \cdots v_k$ is a in (1) above, and $Fact(\Lambda) = [v_0], Fact(v_1 \cdots v_n)$ otherwise.

In the following, we refer to the *colour of a factor* as the minimal colour of its elements.

Note that the previous definition is also valid for infinite plays. Now we easily get the following proposition (the result is obtained by reasoning on partial play using a simple induction combined with a case analysis. Then it directly extends to infinite plays):

Proposition 7. *Let Λ be some infinite play in $\mathbb{G}_{\mathrm{rk}}$ starting from (p_0, s) where Éloïse respects Φ and assume that it never collapses (possibly a copy of) an n-link in s. Let λ the associated infinite play in $\mathbb{G}_{\mathrm{lf}}$ constructed from Φ. Let $\Lambda_0, \Lambda_1, \cdots$ be the factorisation of Λ and, for every $i \geq 0$, let c_i be the colour of Λ_i.*

Then the sequence $(c_i)_{i \geq 0}$ and the sequence of colours visited in λ (ignoring the dummy colours of states of the form q^α $q_?$) are equals.

The previous proposition directly implies that Φ is a winning strategy for Éloïse from (p_0, s) in $\mathbb{G}_{\mathrm{rk}}$. ∎

3) Regularity of sets of winning positions is preserved: We established in Theorem 6 that Éloïse wins in $\mathbb{G}_{\mathrm{rk}}$ from some configuration v if and only if she wins in $\mathbb{G}_{\mathrm{lf}}$ from $\nu_2(v)$. It remains now to prove that regular sets of winning positions are preserved by inverse image by ν_2.

Proposition 8. *Assume that we have an automaton $\mathcal{B}_{\mathrm{lf}}$ that recognises the set of winning configurations in $\mathbb{G}_{\mathrm{lf}}$. Then, one can compute an automaton $\mathcal{B}_{\mathrm{rk}}$ that recognises the set of winning configurations in $\mathbb{G}_{\mathrm{rk}}$.*

Proof: We can safely assume that any control state of $\mathcal{B}_{\mathrm{lf}}$ is of the form (p, R) with $R \subseteq Q_{\mathrm{lf}}$ and such that, after reading some input stack s (possibly with some pending parenthesis) $\mathcal{B}_{\mathrm{lf}}$ is in a state of the form (p, R) with $R = \{r \mid \mathcal{B}_{\mathrm{lf}} \text{ accepts } (r, s')\}$ where s' is the stack obtained from s by closing the pending parenthesis (*i.e.* $s' = s]^k$ for some $k \leq n$).

On input (p, s) the automaton $\mathcal{B}_{\mathrm{rk}}$ simultaneously computes on-the-fly the image by $\nu_2((p, s))$ and simulates $\mathcal{B}_{\mathrm{lf}}$ on it. In order to compute $\nu_2((p, s))$, $\mathcal{B}_{\mathrm{rk}}$ needs to retrieve the states that are winning for the stack obtained by collapsing the link.This is simple as it is given by the $2^{Q_{\mathrm{lf}}}$ component of $\mathcal{B}_{\mathrm{lf}}$ (recall that $\mathcal{B}_{\mathrm{rk}}$ simulates $\mathcal{B}_{\mathrm{lf}}$) and hence the automaton can access it by definition of the model of automata. Indeed, the information is correct before reading the first letter with an n-link, and by induction on the number of n links, if it is correct after reading the k first n-links, on reading the $(k+1)$ n-link, the information is still correct as it was correct for the prefix read so far and therefore $\mathcal{B}_{\mathrm{rk}}$ correctly simulated $\mathcal{B}_{\mathrm{lf}}$ on this prefix. ∎

E. Third step: from $\mathbb{G}_{\mathrm{lf}}$ to $\mathbb{G}'$. Reducing the order

The contain of Section E reuse many of the ideas and results developed in [8]: abstract pushdown automata, automata with oracles, description of the winning region by mean of automata with oracles. Our only contribution in this section is to show that the approach can be used for our purpose of describing the winning region in $\mathbb{G}_{\mathrm{lf}}$. A minor change is that we use deterministic CPDA instead of collapsible pushdown processes (which are CPDAs without input, hence somehow non deterministic which is not an issue when dealing with game graphs).
Therefore, the main result, Theorem ?? comes with no proof and we refer the reader to the long version of [8] for this.

1) Abstract pushdown automata: We situate the techniques developed here in a general and abstract framework of (order-1) pushdown processes whose stack alphabet is a *possibly infinite* set.

An *abstract pushdown automaton* is a tuple $\mathcal{A} = \langle A, Q, \Gamma, \delta \rangle$ where A is a finite input alphabet, Q is a finite set of states, Γ is a (possibly infinite) set called an *abstract pushdown alphabet* and containing a bottom-of-stack symbol denoted $\bot \in \Gamma$, and

$$\delta : Q \times \Gamma \times A \to Q \times \{rew(\gamma), pop, push(\gamma) \mid \gamma \in \Gamma\}$$

is the transition function. We additionally require that $\forall q \in Q$, $\forall \gamma \neq \bot$, $\forall a \in A$, $\delta(q,\gamma,a) \notin \{(q', push(\bot)) \mid q' \in Q\} \cup \{(q', rew(\bot)) \mid q' \in Q\}$ and $\delta(q,\bot,a) \notin \{(q', pop) \mid q' \in Q\} \cup \{(q', rew(\gamma)) \mid q' \in Q \text{ and } \gamma \neq \bot\}$, *i.e.* the bottom-of-stack symbol can only occur at the bottom of the stack, and is never popped or rewritten.

An *abstract pushdown content* is a word in $St = \bot(\Gamma \setminus \{\bot\})^*$. A configuration of $\mathcal{A}$ is a pair (q,σ) with $q \in Q$ and $\sigma \in St$. Note that the top stack symbol in some configuration (q,σ) is the rightmost symbol of σ.

Remark 12. *In general an abstract pushdown automaton is not finitely describable, as the domain of δ is infinite and no further assumption is made on δ.*

Example 8. *A pushdown automaton is an abstract pushdown automaton whose stack alphabet is finite.*

An abstract pushdown automaton $\mathcal{A} = \langle A, Q, \Gamma, \delta \rangle$ induces a (possibly infinite) graph, called an *abstract pushdown graph*, denoted $G(\mathcal{A}) = (V, E)$, whose vertices are the configurations of $\mathcal{A}$ (*i.e.* pairs from $Q \times St$), and edges $E \subseteq V \times A \times V$ are induced by the transition function δ, *i.e.*, from a vertex $(p, \sigma\gamma)$ one has, provided $\delta(p,\gamma,a)$ is defined, an a-labeled edge to:

- $(q, \sigma\gamma')$ whenever $\delta(p,\gamma,a) = (q, rew(\gamma'))$.
- (q,σ) whenever $\delta(p,\gamma,a) = (q, pop)$.
- $(q, \sigma\gamma\gamma')$ whenever $\delta(p,\gamma,a) = (q, push(\gamma'))$.

Example 9. *n-CPDAs that does not create n-links (as $\mathcal{A}_{\text{lf}}$) are special cases of abstract pushdown automata. Let $n > 1$ and consider such an n-CPDA $\mathcal{A} = \langle A, Q, \Sigma, \delta \rangle$. Set Γ to be the set of all order-$(n-1)$ stacks with links over Σ, and for every $p \in Q$, $\gamma \in \Gamma$ with $\sigma = top_1\gamma$ and $a \in A$, we define $\delta(p,\gamma,a)$ to be*

- *(q, pop) if $\delta(q,\sigma,a) = (q, pop_n)$;*
- *$(q, push(\gamma))$ if $\delta(q,\sigma,a) = (q, push_n)$;*
- *$(q, rew(op(\gamma)))$ if $\delta(q,\sigma,a) = (q, op)$ where $k < n$ and op is an order-k action.*
- *undefined otherwise.*

It follows that the abstract pushdown automaton $\langle A, Q, \Gamma, \delta' \rangle$ and $\mathcal{A}$ have isomorphic transition graphs.

Consider now a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q between Éloïse and Abelard. It induces a natural partition $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ of V by setting $V_{\mathbf{E}} = Q_{\mathbf{E}} \times St$ and $V_{\mathbf{A}} = Q_{\mathbf{A}} \times St$. The resulting game graph $\mathcal{G} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$ is called an *abstract pushdown game graph*. Let ρ be a colouring function from Q to a finite set of colours $C \subset \mathbb{N}$. This function is easily extended to a function from V to C by setting $\rho((q,\sigma)) = \rho(q)$. Finally, an *abstract pushdown parity game* is a parity game played on such an abstract pushdown game graph where the colouring function is defined as above.

2) Automata with oracles.: We now define a class of automata to accept the winning positions in an abstract pushdown game. An *automaton with oracles* is a tuple $\mathcal{B} = (\mathcal{S}, Q, \Gamma, \delta, s_{\text{in}}, \mathcal{O}_1 \cdots \mathcal{O}_n, Acc)$ where $\mathcal{S}$ is a finite set of control states, Q is a set of input states, Γ is a (possibly infinite) input alphabet, $s_{\text{in}} \in S$ is the initial state, $\mathcal{O}_i$ are subsets of Γ (called *oracles*) and $\delta : \mathcal{S} \times \{0,1\}^n \to \mathcal{S}$ is the transition function. Finally Acc is a function from $\mathcal{S}$ to 2^Q. Such an automaton is designed to accept in a *deterministic* way configurations of an abstract pushdown automaton whose abstract pushdown content alphabet is Γ and whose control states are Q.

Let $\mathcal{B} = (\mathcal{S}, Q, \Gamma, \delta, s_{\text{in}}, \mathcal{O}_1 \cdots \mathcal{O}_n, Acc)$ be such an automaton. With every $\gamma \in \Gamma$ we associate a Boolean vector $\pi(\gamma) = (b_1, \cdots b_n)$ where

$$b_i = \begin{cases} 1 & \text{if } \gamma \in \mathcal{O}_i \\ 0 & \text{otherwise.} \end{cases}$$

The automaton reads a configuration $C = (q, \gamma_1\gamma_2\cdots\gamma_\ell)$ from left to right. A *run* over C is the sequence $s_0, \cdots, s_{\ell+1}$ such that $s_0 = s_{\text{in}}$ and $s_{i+1} = \delta(s_i, \pi(\gamma_i))$ for every $i = 0, \cdots, \ell$. Finally the run is *accepting* if and only if $q \in Acc(s_{\ell+1})$.

Remark 13. When the input alphabet is finite, it is easily seen that automata with oracles behave as (standard) deterministic finite automata.

We are going to use automata with oracles to accept sets of configurations of n-CPDA that does not have n-links. As seen in Example 9 for an order-n CPDA that does not have n-links, we take Γ to be the set of all order-$(n-1)$ stacks with links. The sets of configurations of an order-n CPDA without n-links accepted by automata using, as oracles, regular sets of order-$(n-1)$ stacks are easily seen to be regular.

Proposition 9. *Fix an order-n CPDA $\mathcal{A}$ and consider an automaton $\mathcal{B}$ with oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$ respectively accepted by automata $\mathcal{B}_1, \ldots, \mathcal{B}_n$ (hence working on order-$(n-1)$ stacks with links). Let C be the set of configurations of $\mathcal{A}$ accepted by $\mathcal{B}$. Then we can construct an automaton (hence working on order-n stacks with links), of size $\mathcal{O}(|\mathcal{B}||\mathcal{B}_1|\cdots|\mathcal{B}_n|)$, accepting the set C.*

Proof: It mainly suffices to mimic the behaviour of $\mathcal{B}$ and to run in parallel the $\mathcal{B}_i$ to compute the value of the oracles. ∎

3) Conditional games and winning regions of abstract pushdown game: From now on, let us fix an abstract pushdown automaton $\mathcal{A} = \langle A, Q, \Gamma, \delta \rangle$ together with a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q and a colouring function ρ using a finite set of colours C. Denote respectively by $\mathcal{G}_{\text{abs}} = (V, E)$ and $\mathbb{G}_{\text{abs}}$ the associated abstract pushdown game graph and abstract pushdown parity game.

We can define an automaton with oracles that accepts Éloïse's winning region of the game $\mathbb{G}_{\text{abs}}$. The oracles of this automaton are defined using conditional games. For

every subset R of Q the game $\mathbb{G}_{\rm abs}(R)$ played over $\mathcal{G}$ is the *conditional game induced by R over $\mathcal{G}$*. A play Λ in $\mathbb{G}_{\rm abs}(R)$ is winning for Éloïse iff one of the following happens:

- In Λ no configuration with an empty stack (*i.e.* of the form $(q, \bot)$) is visited, and Λ satisfies the parity condition.
- In Λ a configuration with an empty stack is visited and the control state in the first such configuration belongs to R.

More formally, the winning condition in $\mathbb{G}_{\rm abs}(R)$ is

$$[\Omega_{par} \setminus V^*(Q \times \{\bot\})V^\omega] \cup V^*(R \times \{\bot\})V^\omega$$

For any state q, any stack letter $\gamma \neq \bot$, and any subset $R \subseteq Q$ it follows from Martin's Determinacy theorem that either Éloïse or Abelard has a winning strategy from $(q, \bot\gamma)$ in $\mathbb{G}_{\rm abs}(R)$. We denote by $\mathcal{R}(q, \gamma)$ the set of subsets R for which Éloïse wins in $\mathbb{G}_{\rm abs}(R)$ from $(q, \bot\gamma)$:

$$\mathcal{R}(q, \gamma) = \{R \subseteq Q \mid (q, \bot\gamma) \text{ is winning for Éloïse in } \mathbb{G}_{\rm abs}(R)\}$$

Then one has the following characterisation of the set of winning positions in $\mathbb{G}_{\rm abs}$ in terms of automaton with oracles.

Theorem 7. *[8] Let $\mathbb{G}_{\rm abs}$ be an abstract pushdown parity game induced by an abstract pushdown automaton $\mathcal{A} = \langle A, Q, \Gamma, \delta\rangle$. Then the set of winning positions in $\mathbb{G}_{\rm abs}$ for Éloïse (respectively for Abelard) is accepted by an automaton with oracles $\mathcal{A} = (\mathcal{S}, Q, \Gamma, \delta, s_i, \mathcal{O}_1 \cdots \mathcal{O}_n, Acc)$ such that*

- *$\mathcal{S} = 2^Q$;*
- *$s_i = \emptyset$.*
- *There is an oracle $\mathcal{O}_{p,R}$ for every $p \in Q$ and $R \subseteq Q$, and $\gamma \in \mathcal{O}_{p,R}$ iff $R \in \mathcal{R}(p, \gamma)$ and $\gamma \neq \bot$.*
- *There is an oracle $\mathcal{O}_\bot$ and $\gamma \in \mathcal{O}_\bot$ iff $\gamma = \bot$*
- *Using the oracles, δ is designed such that:*
  - *From state $\emptyset$ on reading $\bot$, $\mathcal{A}$ goes to $\{p \mid (p, \bot)$ is winning for Éloïse in $\mathbb{G}_{\rm abs}\}$.*
  - *From state S on reading γ, $\mathcal{A}$ goes to $\{p \mid S \in \mathcal{R}(p, \gamma)\}$.*
- *Acc is the identity function.*

4) Solving the abstract pushdown game: In [8] one proved the following result.

Theorem 8. *[8] Let $\mathbb{G}_{\rm abs}$ be an abstract pushdown parity game induced by an abstract pushdown automaton $\mathcal{A} = \langle A, Q, \Gamma, \delta\rangle$. Then one can effectively construct a new game $\mathbb{G}'$ such that the following holds:*

1) *A configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}$ if and only if $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \rho(p_{in}))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*
2) *For every $q \in Q$, $\gamma \in \Gamma$ and $R \subseteq Q$, $R \in \mathcal{R}(q, \gamma)$ if and only if $(q, \gamma, (R, \ldots, R), \rho(q))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*

Moreover in case $\mathbb{G}_{\rm abs}$ aims at coding an order-n CPDA that does not use n-links, the resulting game $\mathbb{G}'$ is an order-$(n-1)$ CPDA game.

Proof: We refer the reader to [8] for the first part of the statement. The fact that in case $\mathbb{G}_{\rm abs}$ aims at coding an order-n CPDA that does not use n-links, the resulting game $\mathbb{G}'$ is an order-$(n-1)$ CPDA game, is simply by observing the shape of $\mathbb{G}'$. ∎

F. Conclusion

We are ready to conclude. For this we reason by induction on the order-n of $\mathbb{G}$. If $n = 1$ (*i.e.* $\mathbb{G}$ is a pushdown game), it is a well known result that the winning region is regular (see *e.g.* [15]).

Assume the result holds for some order $n-1$. Now consider an order-n CDPA game $\mathbb{G}$. Following the transformations $\mathbb{G} \to \mathbb{G}_{\rm rk} \to \mathbb{G}_{\rm lf} \simeq \mathbb{G}_{\rm abs} \to \mathbb{G}'$ one ends up with an order-$(n-1)$ game $\mathbb{G}'$. Hence the winning regions in $\mathbb{G}'$ is recognised by an automaton by induction hypothesis. Then, using Proposition 9 one gets an automaton recognising the winning region in $\mathbb{G}_{\rm lf}$, using Proposition 8 one gets an automaton recognising the winning region in $\mathbb{G}_{\rm rk}$, and finally using Lemma 8 one gets an automaton recognising the winning region in $\mathbb{G}$.

Concerning complexity, going from order n to order $(n-1)$ cost an exponential blow up in the size of automata. Hence constructing an automaton recognising the winning regions is n-ExpTime-complete (completeness come from the fact that deciding whether the initial configuration is winning is already complete for n-ExpTime [6]) and the resulting automaton is n times exponential in the size of the n-CPDA describing $\mathbb{G}$.

We start with the full proof of Corollary 2.

Corollary 2. *Let t be a Σ-labelled tree gererated by an order-n recursion scheme $\mathcal{S}$ and let $\varphi(x)$ 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: We only concentrate on (2) as it implies (1).

We denote by $t, u \models \varphi(x)$ the fact that a node u of t satisfies $\varphi(x)$. For any node u, we let t_u be the tree obtained from t by marking the node u (and no other node). Consider now the MSO formula $\psi(y) = \mathrm{root}(y) \wedge \exists x,\ \mathrm{marked}(x) \wedge \varphi(x)$ (here root and marked are predicates respectively true at the root and at a marked node). Then for any node u, $t, u \models \varphi(x)$ iff $t_u, \varepsilon \models \psi(y)$. Using the well-known equivalence between MSO logic and automata (see *e.g.* [17]), one can construct a parity tree automaton $\mathcal{B}$ that accepts t_u iff $t_u, \varepsilon \models \psi(y)$ iff $t, u \models \varphi(x)$.

In order to construct t_φ, we first annotate t with informations 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 $t[ud]$ starting from q (here $t[u]$ is the subtree of t rooted at u). Existence of $\psi_{q,d}$ is due to the strong relations between μ-calculus and tree automata (see *e.g.* [12]). By (successive applications of) Theorem 2, $\bar{t}$ can be generated by a order-n collapsible automaton.

Let Σ' be the alphabet of $\bar{t}$. Using the annotations on $\bar{t}$, for any node u one can decide, only considering the path from the root to u, whether $\mathcal{B}$ accepts t_u. More precisely, there exists a regular words language L over $\Sigma' \cup \mathsf{Dir}(\Sigma')$ such that a node u of t satisfies φ if and only if the word obtained by reading in $\bar{t}$ the labels and directions from the root to the node u belongs to L. Indeed given a node n of $\bar{t}$, a state q of $\mathcal{B}$ and a direction d one can compute the set of states p of $\mathcal{B}$ that may appear in a run of $\mathcal{B}$ over t_u where:

- $\mathcal{B}$ is in state q in node n.
- the continuations of the run from every node nd' with $d' \neq d$ are accepting provided no marked node appear in the subtrees.
- p is the state in nd.

Indeed it suffices to know whether there is a transition of $\mathcal{B}$ that, from q on reading $t(n)$, goes to q_{d_i} for direction d_i and such that $t,n \models \psi_{q_{d_i},d_i}$ for all $d_i \neq d$ and $q_d = p$. Doing a subset construction, one gets a deterministic finite automaton reading the path from the root to u in $\bar{t}$, and computing the set of possible states of $\mathcal{B}$ in u on the prefix of some accepting run over t_u. Finally to decide whether this prefix of run can be prolongated into an accepting run in case u is marked, it suffices to check whether there is a transition of $\mathcal{B}$ that from q on reading $\underline{t(n)}$ goes to q_{d_i} for direction d_i and such that $t,n \models \psi_{q_{d_i},d_i}$ for all d_i.

Finally an order-n collapsible automaton accepting t_φ is obtained by taking a synchronisation product between an order-n collapsible automaton accepting $\bar{t}$ and a finite *deterministic* automaton recognising L: a note is marked iff the associated control state in the automaton recognising L is finite.

Now to construct $\mathcal{A}$ as in (1), it suffices to consider an n-CPDA $\mathcal{A}$ generating t_φ: a node n belongs to $\|t\|_\varphi$ if the configuration reached on reading n by $\mathcal{A}$ is marked. Hence it suffices to take as final states for $\mathcal{A}$ the one that are marked. ■

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

Proof:

Let t be a Σ-labelled tree given by some order-n recursion scheme $\mathcal{S}$ and let $\mathcal{I}$ be a well-formed MSO-interpretation given by formulas $\varphi_\delta(x)$, $\varphi_\sigma(x)$ for each $\sigma \in \Sigma$ and $\varphi_\ell(x,y)$ for each direction $\ell \in \mathsf{Dir}(\Sigma)$. Let u be a node of t and let t' be the tree obtained by unfolding $\mathcal{I}(t)$ from u[7]. We want to show that t' is the solution of some order-$(n+1)$ scheme. By Theorem 1, it is enough to show that $\mathcal{I}(t)$ restricted to the vertices reachable from u is isomorphic to the ε-closure of the transition graph of some order-$(n+1)$ CPDA $\mathcal{A}$ restricted to the reachable configurations.

Before proceeding with the construction of such a CPDA, we need to fix some notations on MSO logic and on tree automata.

We denote by $t,u,v \models \varphi(x_1,x_2)$ the fact t satisfies $\varphi(x_1,x_2)$ when x_1 is interpreted as u and x_2 as v. For all Σ-labelled tree t, all nodes u and v of t, let $t_{u,v}$ be the tree obtained from t by marking the pair (u,v). Formally, $t_{u,v}$ is the $(\Sigma \times 2^{\{1,2\}})$-labelled tree such that $\mathsf{Dom}(t_{u,v}) = \mathsf{Dom}(t)$ and for $w \in \mathsf{Dom}(t_{u,v})$, $t_{u,v}(w) = (t(w), X)$ where $1 \in X$ iff $w = u$ and $2 \in X$ iff $w = v$.

Similarly we define $t_{\bullet,v}$ (*resp.* $t_{u,\bullet}$, *resp.* $t_{\bullet,\bullet}$) for $\bullet \notin \mathsf{Dom}(t)$ as the tree obtained by marking v by 2 (*resp* u by 2, *resp* no node). ***I.e*** $\bullet$ means here that no node is marked with the corresponding index (1 and / or 2).

Using the well-known equivalence between MSO logic and automata (see *e.g.* [17]), one can construct for any formula $\varphi(x_1,x_2)$ a parity tree automaton $\mathcal{B}_\varphi$ that accepts $t_{u,v}$ iff $t,u,v \models \varphi(x_1,x_2)$.

For all $\ell \in \mathsf{Dir}(\Sigma)$, we write $\mathcal{B}_\ell$ the parity tree automaton corresponding to the formula $\varphi_\ell(x,y)$ of $\mathcal{I}$. Let Q_ℓ be its fine set of states and let Δ_ℓ be its set of transitions.

Annotation of t by MSO-definable sets: The first step of the construction of $\mathcal{A}$ is to annotate t with information concerning essentially the behaviour of the automata $\mathcal{B}_\ell$ on the subtrees of t. The resulting annotated version of t is denoted $\bar{t}$. More precisely the annotated tree $\bar{t}$ has for each node u satisfying $\varphi_\delta(x)$ on t, the following finite information:

- the unique $\sigma \in \Sigma$ such that $t,u \models \varphi_\sigma(x)$. Unicity is by definition of a well-formed interpretation.
- $d_\uparrow \in \mathsf{Dir}(\Sigma) \cup \{\bullet\}$ which is the direction from the father of the curent node to the current node (***i.e.*** u is of the form $u'd_\uparrow$ for some $u' \in \mathsf{Dom}(t)$) and $\bullet$ if the current node is the root.
- for each $\ell \in \mathsf{Dir}(\Sigma)$, we have:
  - $i_\ell \in \{\uparrow, \downarrow, \circlearrowleft, \bot\}$ such that
    - $i_\ell = \bot$ iff there are no v such that $t,u,v \models \varphi_\ell(x,y)$,
    - $i_\ell = \uparrow$ iff there is a unique v such that $t,u,v \models \varphi_\ell(x,y)$ which does not lay below u,
    - $i_\ell = \downarrow$ iff there is a unique v such that $t,u,v \models \varphi_\ell(x,y)$ which is below u,
    - $i_\ell = \circlearrowleft$ iff $t,u,u \models \varphi_\ell(x,y)$.
  - the set R_ℓ of states $q \in Q_\ell$ such that there exists a partial accepting run of $\mathcal{B}_\ell$ on t deprived of the nodes below u which assigns the state q to u,

[7] We assume that $t,u \models \varphi_\delta(x)$.

– the set S_ℓ of pairs $(d, q) \in \mathsf{Dir}(\Sigma) \times Q_\ell$ such that there exists an accepting run of $\mathcal{B}_\ell$ starting from q on the subtree of $t_{\bullet,\bullet}$ rooted at ud,
– the set T_ℓ of pairs $(d, q) \in \mathsf{Dir}(\Sigma) \times Q_\ell$ such that there exists a node v' below ud s.t. $\mathcal{B}_\ell$ has an accepting run on the subtree of $t_{\bullet,v'}$ rooted at ud starting in state q.

Let Σ' be the resulting labelling alphabet of $\bar{t}$. As the information annotated on $\bar{t}$ is MSO-definable in t, we know from Corollary 2 that $\bar{t}$ is the solution of some order-n scheme.

Replacing MSO-formulas on t by tree-walking automata working on $\bar{t}$.: Fix a direction $\ell \in \mathsf{Dir}(\Sigma)$. Thanks to the extra information available on $\bar{t}$, it is possible to decide if a pair of nodes (u, v) satisfies the formula $\varphi_\ell(x, y)$ on t using a deterministic tree-walking automaton running on $\bar{t}$. Intuitively a tree-walking automaton is a finite state automaton that can navigate through the tree.

Formally, a *deterministic tree-walking automaton* working on Σ-labelled trees is a tuple $W = (Q, q_0, F, \delta)$ where Q is the finite set of states, $q_0 \in Q$ is the initial state, F is the set of final states and δ is the transition function. The transition function associates to a pair $(p, \sigma) \in Q \times \Sigma$ – corresponding respectively to the current state and node label – a pair $(q, a) \in Q \times (\{\uparrow, \varepsilon\} \cup \mathsf{Dir}(\Sigma))$ where q is the new state and a is action to perform. Intuitively ε corresponds to "staying in the current node", $\uparrow$ to "going to the parent node" and $d \in \mathsf{Dir}(\Sigma)$ corresponds to "going to the d-son".

We say that W accepts a pair of nodes (u, v) if it can reach v in a final state starting from u in the initial state.

We claim that there exists a deterministic tree-walking automaton W_ℓ such for any pair (u, v) of nodes of t, we have:

$$W_\ell \text{ accepts } (u, v) \text{ in } \bar{t} \text{ iff } t, u, v \models \varphi_\ell(x, y).$$

The automaton W_ℓ works in two phases: during the first phase the automaton only goes up in the tree (or stay in the current node) and during the second phase it only goes down in the tree (or stay in the current node). Both phases can potentially be empty. In fact, to accept a pair (u, v) the automaton will first go up to the greatest common ancestor of u and v and down to the v.

Assume that W_ℓ started at a node u and denote by v the unique node (if it exists) such that $t, u, v \models \varphi_\ell(x, y)$.

Initialisation.: The automaton is in its initial state q_0 at node u and reads a label $(\sigma, d_\uparrow, (i_k, R_k, S_k, T_k)_{k \in \mathsf{Dir}(\Sigma)})$. The automaton checks in which of the following cases, we are:

The node v does not exists (i.e. $i_\ell = \perp$): No transition is defined.

The node v is equal to u (i.e. $i_\ell = \circlearrowleft$: The automaton goes to the accepting state.

The node v is not below u (i.e. $i_\ell = \uparrow$): The automaton begins the first phase while memorising the set X of state $q \in Q_\ell$ such that $\mathcal{B}_\ell$ admits an accepting run on the subtree of $t_{u,\bullet}$ rooted at u. This set can be computed from Δ_ℓ and S_ℓ.

The node v is below u (i.e. $i_\ell = \downarrow$): The automaton begins the second phase. It computes the unique direction d and the set Y of states $q \in Q_\ell$ such that:
– $\mathcal{B}_\ell$ admits an accepting run on the tree $t_{u,\bullet}$ deprived of the nodes below ud and assigning states q to ud,
– $(q, d) \in S'_\ell$.

First phase.: The automaton is at some node w and stores the set X of states $q \in Q_\ell$ such that $\mathcal{B}_\ell$ admits an accepting run on the subtree of $t_{u,\bullet}$ rooted at w. The label of the node w is $(\sigma, d_\uparrow, (i_k, R_k, S_k, T_k)_{k \in \mathsf{Dir}(\Sigma)})$. The automaton goes up in the tree (while remembering $d_\uparrow$ and X) to a node w' whose label is $(\sigma', d'_\uparrow, (i'_k, R'_k, S'_k, T'_k)_{k \in \mathsf{Dir}(\Sigma)})$. Let $d_1, \ldots, d_m$ be the set of directions of σ'. Let j be the index of $d_\uparrow$ in this enumeration.

There are now there cases:

The node v is the current node. This is the case iff there exists a state $q \in R'_\ell$ and a transition in Δ_ℓ starting in state q with $\sigma \times \{2\}$ as label and associating states q_i to the d_i-son such that:
– q_j belongs to X ,
– for all $i \neq j$, $(d_i, q_i) \in S'_\ell$.

In this case, the automaton goes to the accepting state.

The node v is below d-son of w for some $d \in \mathsf{Dir}(\Sigma)$.
This is the case iff there exists a state $q \in R'_\ell$ and a transition in Δ_ℓ starting in state q with $\sigma \times \emptyset$ as label and associating states q_i to the d_i-son such that there exists $j' \neq j \in [m]$ s.t.
– q_j belongs to X,
– for all $i \neq j$, $(d_i, q_i) \in S'_\ell$,
– $(d_{j'}, q_{j'}) \in T'_\ell$.

In this case, the automaton begins the second phase while memorising the sets of all Y of all state $q_{j'}$ matching this definition together with the direction $d_{j'}$.[8] the accepting state.

The node v is not below w. This is the case when the two previous cases do not apply. The automaton update the new set X using $d_\uparrow$, Δ_ℓ and the old value of X and goes to the beginning of the first phase.

The second phase: The automaton is at some node w and stores a direction d and the set Y of states $q \in Q_\ell$ such:

- $\mathcal{B}_\ell$ admits an accepting run on the tree $t_{u,\bullet}$ deprived of the nodes below wd and assigning states q to wd
- there exists a node v' below wd such that $\mathcal{B}_\ell$ admits an accepting run on the subtree of $t_{\bullet,v'}$ rooted at wd and starting in state q.

[8] Due to the restriction imposed on $\varphi_\ell(x, y)$ by the fact that $\mathcal{I}$ is a well-formed MSO-interpretation, there cannot be two different directions. Otherwise, we would have $v \neq v'$ such that $t, u, v \models \varphi(x, y)$ and $t, u, v' \models \varphi(x, y)$.

The automaton goes down in direction d (while remembering Y) to a node w' whose label is $(\sigma', d'_\uparrow, (i'_k, R'_k, S'_k, T'_k)_{k\in\mathsf{Dir}(\Sigma)})$. Let $d_1, \ldots, d_m$ be the set of directions of σ'.

There are two cases:

The node v is below the d'-son of w' for some $d' \in \mathsf{Dir}(\Sigma)$.
This is the case iff there exists a state $q \in Y$ and a transition in Δ_ℓ starting in state q with $\sigma \times \emptyset$ as label and associating states q_i to the d_i-son such that there exists $j \in [m]$ s.t.
- for all $i \neq j$, $(d_i, q_i) \in S'_\ell$,
- $(d_{j'}, q_{j'}) \in T'_\ell$.

In this case, the automaton return at the beginning of the second phase and update the set Y with all state q_j matching this condition. It also stores the direction d_j.

The node v is the current node. This is precisely when the previous case does not hold. The automaton moves to an accepting state.

Construction of the order-$(n+1)$ CPDA $\mathcal{B}$.: By Theorem 1, there exists an order-n CPDA $\mathcal{C} = \langle\, \mathsf{Dir}(\Sigma) \cup \{\varepsilon\}, \Gamma, Q, \delta, q_0, F \,\rangle$, and a mapping $\rho : Q \to \Sigma'$ such that $\bar{t}$ is the tree generated by $\mathcal{C}$ and ρ.

Hence for every node $u = d_1 \cdots d_m \in \mathsf{Dom}(t)$, there exists a unique sequence of configuration $(q_0, s_0), \ldots, (q_m, s_m)$ of $\mathcal{C}$ such that:

- there exists a path in $G(\mathcal{C})$ labelled by ε^* from the initial configuration to (q_0, s_0),
- for all $i \in [0, m]$, (q_i, s_i) does not have out-going ε-labelled arcs in $G(\mathcal{C})$,
- for all $i \in [0, m-1]$, there exists a path labelled in $d_{i+1}\varepsilon^*$ from (q_i, s_i) to (q_{i+1}, s_{i+1}) in $G(\mathcal{C})$.

Such a sequence can be coded as order-$(n+1)$ stack s_u – recall that the s_i are order-n – in the following way:

$$s_u = [\tilde{s_0}, \tilde{s_1}, \ldots, \tilde{s_{m-1}}, push_1^{q_m,1}(s_m)]$$

where for all $i \in [0, m-1]$, $\tilde{s_i} = push_1^{d_{i+1},1}(push_1^{q_i,1}(s_i))$. The stack alphabet contains both the stacks alphabet of $\mathcal{C}$ and its set of states.

The automaton $\mathcal{B}$ works on stacks corresponding to some s_u for some $u \in \mathsf{Dom}(t)$. The states of $\mathcal{C}$ include a distinguished state $q_\star$ and the states of all the tree-walking automata $(W_\ell)_{\ell\in\mathsf{Dir}(\Sigma)}$ which we assumed to be disjoint. The configurations of $\mathcal{B}$ that are source of non-ε-labelled arcs will be of the form $(q_\star, s_u)$ for some $u \in \mathsf{Dom}(t)$. The intended behaviour of $\mathcal{B}$ is that for some $\ell \in \mathsf{Dir}(\Sigma)$, $t, u, v \models \varphi_\ell(x, y)$ then $\mathcal{B}$ can go from the configuration $(q_\star, s_u)$ to the configuration $(q_\star, s_v)$ by a path labelled by $\ell\varepsilon^*$.

First $\mathcal{B}$ moves by ℓ-labelled transition to the configuration (q_0^ℓ, s_u) where q_0^ℓ is the initial state of the tree-walking automaton W_ℓ. Recall that the existence of the vertex v is annotated in $\rho(top_1(s_u))$.

In a configuration of the form (p, s_u) with p a state of W_ℓ, $\mathcal{B}$ simulates the behaviour of W_ℓ on $\bar{t}$ at node u in state p by a sequence of ε-transitions. As $\bar{t}(u) = \rho(top_1(s_u))$, $\mathcal{B}$ can compute the transition taken by the automaton W_ℓ on $\bar{t}$ at node u in state p. The behaviour of $\mathcal{B}$ will be such that if W_ℓ goes from (p, u) to (q, u') in one step then $\mathcal{B}$ will go through a series of ε-transitions from (p, s_u) to $(q, s_{u'})$.

We distinguish several cases depending on the action performed by W_ℓ.

- W_ℓ stays in the current node in state q. Then $\mathcal{B}$ changes its state to q by an ε-transition.
- W_ℓ goes to its parent node in state q (i.e. $u = u'd$ and W_ℓ ends up in u' in state q). Then $\mathcal{B}$ performs pop_{n+1} followed by a pop_1 and moves to state q. The configuration of $\mathcal{B}$ is now $(q, pop_1(pop_{n+1}(s_u))) = s_{u'}$.
- W_ℓ goes to its d-son in state q (i.e. $u = u'd$ and W_ℓ ends up in ud in state q). Assume that s_u is equal to:

  $$[\tilde{s_0}, \tilde{s_1}, \ldots, \tilde{s_{m-1}}, push_1^{q_m,1}(s_m)]$$

  and that s_{ud} is of the from:

  $$\tilde{s_0}, \tilde{s_1}, \ldots, push_1^{d,1}(push_1^{q_m,1}(s_m)), push_1^{q_{m+1},1}(s_{m+1})$$

  By definition, there exists a path π in $G(\mathcal{C})$ from (q_m, s_m) and (q_{m+1}, s_m) labelled by $d\varepsilon^*$.
  Then $\mathcal{B}$ starts by performing a $push_1^{d,1}$ followed by $push_{n+1}$ and pop_1. At this point the stack is:

  $$[\tilde{s_0}, \tilde{s_1}, \ldots, push_1^{d,1}(push_1^{q_m,1}(s_m)), push_1^{q_m,1}(s_m).]$$

  $\mathcal{B}$ pops the state q_m and simulates the order-n operations of $\mathcal{C}$ along the path π using ε-transitions. When no ε-transition of $\mathcal{C}$ can be applied, $\mathcal{B}$ goes to state q_{m+1}.

Eventually $\mathcal{B}$ will reach a configuration of the form (q_f^ℓ, s_v) where q_f^ℓ is the accepting state of W_ℓ. It then goes to the state $q_\star$.

From its initial configuration, $\mathcal{B}$ deterministically build the stack s_{u_0} (which correspond to the vertex u_0 from which $\mathcal{I}(t)$ is unfolded) by using sequence of ε-transitions and goes to the state $q_\star$.

By construction, we have that the ε-closure of $\mathcal{B}$ restricted to the vertices reachable from its initial configuration is isomorphic to $\mathcal{I}(t)$ restricted to the vertices reachable from u_0. The isomorphism simply maps a configuration $(q_\star, s_u) \in G(\mathcal{B})$ to $u \in \mathsf{Dom}(t)$.

■