Collapsible Pushdown Parity Games

Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, Andrzej S. Murawski, C. -H. Luke Ong, Olivier Serre

Collapsible Pushdown Parity Games

CHRISTOPHER H. BROADBENT, Department of Computer Science, University of Oxford, UK
ARNAUD CARAYOL, CNRS, LIGM (Université Paris Est & CNRS), France
MATTHEW HAGUE, Royal Holloway, University of London, UK
ANDRZEJ S. MURAWSKI, Department of Computer Science, University of Oxford, UK
C.-H. LUKE ONG, Department of Computer Science, University of Oxford, UK
OLIVIER SERRE, Université de Paris, IRIF, CNRS, France

This paper studies a large class of two-player perfect-information turn-based parity games on infinite graphs, namely those generated by collapsible pushdown automata. The main motivation for studying these games comes from the connections from collapsible pushdown automata and higher-order recursion schemes, both models being equi-expressive for generating infinite trees. Our main result is to establish the decidability of such games and to provide an effective representation of the winning region as well as of a winning strategy. Thus, the results obtained here provide all necessary tools for an in-depth study of logical properties of trees generated by collapsible pushdown automata/recursion schemes.

CCS Concepts: • **Theory of computation → Formal languages and automata theory**; **Verification by model checking**.

Additional Key Words and Phrases: Higher-Order (Collapsible) Pushdown Automata, Two-Player Perfect-Information Trun-Based Parity Games, Logic

ACM Reference Format:
Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. 2021. Collapsible Pushdown Parity Games. 1, 1 (April 2021), 51 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This paper studies a large class of two-player perfect-information turn-based parity games on infinite graphs, namely those generated by collapsible pushdown automata (CPDA).

Parity Games on Infinite Graphs

A two-player perfect-information turn-based parity game on a graph (or simply a parity game) is played by two players, Éloïse and Abelard, who move a pebble along edges of a graph whose vertices have been partitioned between the two players and coloured by a function assigning to every vertex a colour chosen in a finite subset of $\mathbb{N}$. The player owning the current vertex, chooses

Authors' addresses: Christopher H. Broadbent, Department of Computer Science, University of Oxford, Oxford, UK, chbroadbent@gmail.com; Arnaud Carayol, CNRS, LIGM (Université Paris Est & CNRS), 5 boulevard Descartes — Champs sur Marne, Marne-la-Vallée Cedex 2, 77454, France, Arnaud.Carayol@univ-mlv.fr; Matthew Hague, Royal Holloway, University of London, London, UK, Matthew.Hague@rhul.ac.uk; Andrzej S. Murawski, Department of Computer Science, University of Oxford, Oxford, UK, Andrzej.Murawski@cs.ox.ac.uk; C.-H. Luke Ong, Department of Computer Science, University of Oxford, Oxford, UK, Luke.Ong@cs.ox.ac.uk; Olivier Serre, Université de Paris, IRIF, CNRS, Bâtiment Sophie Germain, Case courrier 7014, 8 Place Aurélie Nemours, Paris Cedex 13, 75205, France, Olivier.Serre@cnrs.fr.

XXXX-XXXX/2021/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

where to move the pebble next and so on forever. Hence, a play is an infinite path in the graph, and the winner is determined thanks to the colouring function by declaring Éloïse to win if and only if the smallest colour appearing infinitely often is even.

Parity games have been widely studied since the 80s because of their close links to important problems arising from logic. A fundamental result of Rabin is that ω-regular tree languages, equivalently tree languages definable in monadic second-order (MSO) logic, form a Boolean algebra [31]. The difficult part of the proof is complementation, and since the publication of this result in 1969, it has been a challenging problem to simplify it. A much simpler one was obtained by Gurevich and Harrington in [21] making use of Muller games for checking membership of a tree in the language accepted by an automaton: Éloïse builds a run on the input tree while Abelard tries to exhibit a rejecting branch in the run. The proof of Gurevich and Harrington was followed by many others trying to simplify the original proof of Rabin (in particular Emerson and Jutla who introduced the connection with parity games in [19]), and beyond this historical result, the tight connection between automata and games is one of the main tools in the areas of automata theory and logic (see *e.g.* [35, 39, 40]).

The above-mentioned result of Rabin is equivalent to the fact that, given a formula from MSO logic, one can decide whether it holds in the complete infinite binary tree. Whether this result can be extended to more and more complex classes of trees is an active line of research since then. While decidability of MSO logic on the complete binary tree is equivalent to deciding whether Éloïse has a winning strategy in a parity game played on a *finite* graph, extensions to more complex trees require one to consider games played on infinite graphs (and the more general the trees, the more general the graphs to be considered).

Since the late 1990s, another important motivation for considering games played on infinite graphs emerged because of their connections with program verification. Here, there is a trade-off between richness of the graph describing the program to verify and decidability of the logic used to express the property to check. Regarding logic, most of the logics considered in program verification are captured by the μ-calculus (an extension of modal logic with fixpoint operators) and therefore the model-checking problem is reduced again to solving a parity game played on a graph that is a synchronised product between the graph describing the system to verify and a finite graph describing the dynamic of the formula. Hence, the quest here is to look for graphs that model programs using natural features in programming languages (*e.g.* recursion, higher-order arguments, rich data domains, etc.) and whose associated parity games remain decidable.

Both objectives — extending Rabin's result to richer trees and verifying programs with natural features in programming languages — games played on graphs generated by pushdown automata and their extensions, in particular collapsible pushdown automata, have proven to be fruitful. In a nutshell, collapsible pushdown automata extend usual pushdown automata by replacing the (order-1) stack by an order-n stack that is defined as a stack whose elements are order-$(n-1)$ stacks and whose base symbols are equipped with links pointing deeper in the stack and that can later be used to collapse the stack.

Main Results

Collapsible pushdown automata are equi-expressive with higher-order recursion schemes — these are essentially finite typed deterministic term rewriting systems that generate an infinite tree when one applies the rewriting rules *ad infinitum* — for generating trees [23, 24], this class of trees subsumes all known classes of trees with decidable MSO theories. Regarding programs, collapsible pushdown automata permit to capture higher-order procedure calls — a central feature in modern day programming and supported by many languages such as C++, Haskell, OCaML, Javascript, Python, or Scala.

Hence, considering parity games played on transition graphs of (collapsible) pushdown automata is a central problem for both extending Rabin's seminal result and verifying real-life programs. The study of such games raises three questions of increasing difficulty.

(1) Decide, for a given initial position, whether Éloïse has a winning strategy, *i.e.* whether she has a way to play that guarantees she wins regardless of the choices of Abelard. In the context of program verification, the counterpart of this question is the (local) model-checking problem.
(2) Finitely describe Éloïse's winning region, *i.e.* the set of all positions from which she has a winning strategy. While in the setting of games on finite graphs this is equivalent to the previous question, when considering an infinite graph it is unclear whether a finite presentation of the winning region exists and, when it does, specific tools must be used to describe such an object. In the context of program verification, the counterpart of this question is the global model-checking problem.
(3) Finitely describe, for a given initial position, a winning strategy for Éloïse. Note that a classical result (positional determinacy [19]) on parity games states that winning strategies can always be chosen to be positional, *i.e.* to depend only on the current vertex; however, when describing a winning strategy in a game played on an infinite graph, the purpose is to find a suitable machine model of implementing a winning strategy rather than focusing on capturing a special (simple) form of winning strategies. In the context of program verification, the counterpart of this question is the synthesis problem.

In this paper we positively answer those questions. More specifically, our main Theorem implies the following.

(1) One can decide, for a given initial position, whether Éloïse has a winning strategy and this is an n-ExpTime-complete problem, where n is the order of the underlying collapsible pushdown automaton.
(2) We introduce a model of finite-state automata defining regular sets of configurations of collapsible pushdown automata and prove that the winning region is always such an (effective) regular set.
(3) We introduce a model of collapsible pushdown automata tailored to describing strategies and prove that, for any game, we can compute a winning strategy described by such a machine.

Note that the above-mentioned results were presented by the authors in a series of papers in the LiCS conference [8, 15, 23] and that the current paper gives a unifying and complete presentation of their proofs.

Related Work

We briefly review the known results on collapsible pushdown parity games (and subclasses). See Table 1 for a summary.

The first paper explicitly considering pushdown games (*i.e.* order-1 CPDA games) is [37, 38]: an optimal algorithm for deciding the winner is given (ExpTime-complete) as well as a construction of a strategy realised by a synchronised pushdown automaton. However, decidability can be derived from the MSO decidability of pushdown graphs [30] in combination with the existence of positional winning strategies in parity games on infinite graphs [19]: indeed one can write an MSO formula stating the existence of a positional winning strategy for Éloïse (see *e.g.* [10] for such a formula). A construction similar to the one in [37, 38] was given by Serre in his Ph. D. [33], and we partly build upon it in the present paper. Another approach, using two-way alternating parity tree automata, was developed by Vardi in [36]. The winning region was characterised in [9, 32] and later in [22, 26] using saturation techniques.

Cachat first considered parity games played on transition graphs of higher-order pushdown automata (HOPDA, a strict subclass of collapsible pushdown automata) in [11] providing an optimal algorithm for deciding the winner (n-ExpTime-complete, where n is the order). As for pushdown games, decidability can be derived from the MSO decidability of higher-order pushdown graphs [17] in combination with the existence of positional winning strategies in parity games on infinite graphs [19]. An alternative simpler proof was given in [14] that permits moreover to characterise the winning region and to construct a synchronised order-n higher-order pushdown automaton realising a winning strategy. Also see [16] for an approach extending the techniques of [36] to higher-order, and [3, 25] for saturation techniques (for the reachability winning condition only).

Order-2 collapsible pushdown parity games were considered in [28] (under the name of panic automata), where an optimal algorithm for deciding the winner (2-ExpTime-complete) was given. The general case was later solved in [23]. Winning regions were characterised in [8] and the winning strategies in [15] (even if the results are somehow implicit in [23]). Finally, in [5], for the case of the reachability winning condition, the approach of [25] was extended, leading to an algorithm based on the saturation method to compute the winning region, and on top of this algorithm the C-SHORe tool was developed [6].

Consequences

The consequences of the results presented here, together with the equi-expressivity result [15, 23, 24] between higher-order recursion schemes and collapsible pushdown automata for generating trees, are mainly for the study of logical properties of the infinite trees generated by recursion schemes. In particular, they imply the decidability of the MSO model-checking problem, both its local [23] and global version (also known as reflection) [8], and the MSO selection problem (a synthesis-like problem) [15].

Due to space constraints, these results are discussed in full detail in a companion paper [7].

Structure of This Paper

The article is organised as follows. Section 2 introduces the main concepts and some intermediate results. In Section 3 we state our main result. Its proof is by induction and each induction step is divided into three sub-steps, which are respectively described in Section 4 (providing a normal form for CPDA), Section 5 (getting rid of the outmost links in the stack structure) and Section 6 (reducing the order of the CPDA). Section 7 summarises the proof and establishes matching upper and lower complexity bounds. Finally, Section 8 discusses some logical consequences for collapsible pushdown graphs.

2 PRELIMINARIES

2.1 Basic Objects

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 and the length of a word u is denoted by $|u|$. Let u be a finite word and v be a (possibly infinite) word. Then $u \cdot v$ (or simply uv) denotes the concatenation of u and v; the word u is a prefix of v iff there exists a word w such that $v = u \cdot w$.

A ***graph*** is a pair $G = (V, E)$, where V is a (possibly infinite) set of ***vertices*** and $E \subseteq V \times V$ is a (possibly infinite) set of ***edges***. For every vertex v we let $E(v) = \{w \mid (v, w) \in E\}$. We assume that for each vertex v of G $E(v)$ is not empty.

When τ is a (partial) mapping, we let $\mathrm{dom}(\tau)$ denote its domain.

	Pushdown	n-HOPDA	n-CPDA
Winning strategy	Realised by a synchronised pushdown automaton [33, 37]	Realised by a synchronised n-HOPDA [14, 16]	Realised by a synchronised n-CPDA [15, 23]
Winning region	Regular [9, 22, 26, 32]	Regular [14, 16] See also [3, 25] for reachability using saturation methods	Regular [8] See also [5] for reachability using saturation methods
Solving	Decidable [30] + [19] ExpTime-complete [33, 36, 37]	Decidable [17] + [19] n-ExpTime-complete [11, 14]	n-ExpTime-complete [23] See also [28] for a previous study at order-2

Table 1. Known results on collapsible pushdown parity games and subclasses.

2.2 Two-Player Perfect-Information Parity Games

An ***arena*** is a triple $\mathcal{G} = (G, V_E, V_A)$, where $G = (V, E)$ is a graph and $V = V_E \uplus V_A$ is a partition of the vertices among two players, Éloïse and Abelard.

Éloïse and Abelard play in $\mathcal{G}$ by moving a pebble along edges. A ***play*** from an initial vertex v_0 proceeds as follows: the player owning v_0 (*i.e.* Éloïse if $v_0 \in V_E$, Abelard otherwise) moves the pebble to a vertex $v_1 \in E(v_0)$. Then the player owning v_1 chooses a successor $v_2 \in E(v_1)$ and so on. As we assumed that there is no dead-end, a play is an infinite word $v_0v_1v_2 \cdots \in V^\omega$ such that for all $0 \leq i$ one has $v_{i+1} \in E(v_i)$. A ***partial play*** is a prefix of a play, *i.e.* it is a finite word $v_0v_1 \cdots v_\ell \in V^*$ such that for all $0 \leq i < \ell$ one has $v_{i+1} \in E(v_i)$.

A ***strategy*** for Éloïse is a function $\varphi_E : V^*V_E \to V$ assigning, to every partial play ending in some vertex $v \in V_E$, a vertex $v' \in E(v)$. Strategies of Abelard are defined likewise, and usually denoted φ_A. In a given play $\lambda = v_0v_1 \cdots$ we say that Éloïse (*resp.* Abelard) ***respects a strategy*** φ_E (*resp.* φ_A) if whenever $v_i \in V_E$ (*resp.* $v_i \in V_A$) one has $v_{i+1} = \varphi_E(v_0 \cdots v_i)$ (*resp.* $v_{i+1} = \varphi_A(v_0 \cdots v_i)$).

A ***winning condition*** is a subset $\Omega \subseteq V^\omega$ and a (two-player perfect information) ***game*** is a pair $\mathbb{G} = (\mathcal{G}, \Omega)$ consisting of an arena and a winning condition. A game is finite if it is played on a finite arena.

A play λ is ***won*** by Éloïse if and only if $\lambda \in \Omega$; otherwise λ is won by Abelard. A strategy φ is ***winning*** for player X in $\mathbb{G}$ from a vertex v_0 if any play starting from v_0 where X respects φ is won by X. Finally a vertex v_0 is ***winning*** for X in $\mathbb{G}$ if X has a winning strategy φ from v_0.

A ***parity winning condition*** is defined by a ***colouring function*** ρ, *i.e.* a mapping $\rho : V \to C \subset \mathbb{N}$, where C is a *finite* set of ***colours***. The parity winning condition associated with ρ is the set $\Omega_\rho = \{v_0 v_1 \cdots \in V^\omega \mid \liminf(\rho(v_i))_{i\geq 0} \text{ is even}\}$, *i.e.* a play is winning if and only if the smallest colour visited infinitely often is even. A ***parity game*** is a game of the form $\mathbb{G} = (\mathcal{G}, \Omega_\rho)$ for some colouring function.

2.3 Stacks with Links and Their Operations

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

As usual, the bottom-of-stack symbol $\perp$ cannot be popped from or pushed onto a stack. Thus we require an ***order-1 stack*** to be a non-null sequence $[\gamma_1 \cdots \gamma_l]$ of elements of Γ such that for all $1 \leq i \leq l$, $\gamma_i = \perp$ iff $i = 1$. We inductively define $\perp_k$, the ***empty k-stack***, as follows: $\perp_0 = \perp$ and $\perp_{k+1} = [\perp_k]$.

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

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

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

By abuse of notation, we set $top_{ord(s)+1}(s) = s$. Note that $pop_i(s)$ is undefined if $top_{i+1}(s)$ is a one-element i-stack. For example $pop_2([[\perp \alpha \beta]])$ and $pop_1([[\perp \alpha \beta][\perp]])$ are both undefined.

There are two kinds of *push* operations. We start with the *order*-1 *push*. Let γ be a non-$\perp$ stack symbol and $1 \leq e \leq ord(s)$, we define a new stack operation $push_1^{\gamma,e}$ that, when applied to s, first attaches a link from γ to the $(e-1)$-stack *immediately* below the top $(e-1)$-stack of s, then pushes γ (with its link) onto the top 1-stack of s. Formally, for $1 \leq e \leq ord(s)$ and $\gamma \in (\Gamma \setminus \{\perp\})$, we define

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

where

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

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

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

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

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

We also define, for any stack symbol γ, an operation on stacks that rewrites the topmost stack symbol *without modifying* its link. Formally:

$$rew_1^{\gamma}\; \underbrace{[s_1 \cdots s_{l+1}]}_{s} \quad = \quad \begin{cases} [s_1 \cdots s_l\; rew_1^{\gamma}\, s_{l+1}] & \text{if } ord(s) > 1 \\ [s_1 \cdots s_l\; \widehat{\gamma}] & \text{if } ord(s) = 1 \text{ and } l \geq 1 \end{cases}$$

where $\widehat{\gamma}$ denotes the symbol γ with a link to the same target as the link from s_{l+1}. Note that $rew_1^{\gamma}(s)$ is undefined if $top_2(s)$ is the empty 1-stack.

Finally, there is an important operation called *collapse*. We say that the n-stack s_0 is a ***prefix*** of an n-stack s, written $s_0 \leq s$, just in case s_0 can be obtained from s by a sequence of (possibly higher-order) *pop* operations. Take an n-stack s where $s_0 \leq s$, for some n-stack s_0, and $top_1\, s$ has a link to $top_e(s_0)$. Then *collapse* s is defined to be s_0.

Example 2.1. To avoid clutter, when displaying n-stacks in examples, we shall omit 1-links (indeed by construction they can only point to the symbol directly below), writing e.g. $[[\bot][\bot\alpha\,\beta]]$ instead of $[[\bot][\bot\;\alpha\;\beta]]$.

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

$$\begin{aligned} push_1^{\gamma,2}(s) &= [[[\,\bot\,\alpha]]\;[[\,\bot][\,\bot\,\alpha\,\gamma]]] \\ collapse\,(push_1^{\gamma,2}(s)) &= [[[\,\bot\,\alpha]]\;[[\,\bot]]] \\ \underbrace{push_1^{\gamma,3}(rew_1^{\beta}(push_1^{\gamma,2}(s)))}_{\theta} &= [[[\,\bot\,\alpha]]\;[[\,\bot][\,\bot\,\alpha\,\beta\,\gamma]]]. \end{aligned}$$

Then $push_2(\theta)$ and $rew_1^{\alpha}(push_3(\theta))$ are respectively

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

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

We have $collapse\,(push_2(\theta)) = collapse\,(rew_1^{\alpha}(push_3(\theta))) = collapse(\theta) = [[[\,\bot\,\alpha]]]$.

The set Op_n^{Γ} of order-n CPDA ***stack operations*** over stack alphabet Γ (or simply Op_n if Γ is clear from the context) comprises six types of operations:

(1) pop_k for each $1 \leq k \leq n$,
(2) $push_j$ for each $2 \leq j \leq n$,
(3) $push_1^{\gamma,e}$ for each $1 \leq e \leq n$ and each $\gamma \in (\Gamma \setminus \{\,\bot\,\})$,
(4) rew_1^{γ} for each $\gamma \in (\Gamma \setminus \{\,\bot\,\})$,
(5) *collapse*, and
(6) *id* for the identity operation (*i.e.* $id(s) = s$ for all stack s).

Remark 2.2. One way to give a formal semantics of the stack operations is to work with appropriate numeric representations of the links as explained in [24, Section 3.2]. We believe that the informal presentation should be sufficient for this work and hence refer the reader to [24] for a formal definition of stacks.

2.4 Collapsible Pushdown Automata (CPDA) and their Transition Graphs

Collapsible pushdown automata are a generalisation (to all finite orders) of *pushdown automata with links* [1]. They are defined as automata with a finite control and a stack as memory. In this work, we are interested in CPDA as generators for infinite graphs rather than word acceptors or generators of an infinite tree (see [24] for corresponding definitions), hence we consider a non-deterministic version of them but do not equip them with an input alphabet.

An ***order-n collapsible pushdown automaton*** (***n-CPDA***) is a 4-tuple $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$, where Γ is the stack alphabet, Q is the finite set of control states, $q_0 \in Q$ is the initial state, and $\Delta : Q\times\Gamma \to 2^{Q\times Op^{\Gamma}_n\times Op^{\Gamma}_n}$ is the transition function and satisfies the following constraint. For any $q, \gamma \in Q \times \Gamma$, for any $(q', op_1, op_2) \in \Delta(q, \gamma)$ one has that $op_1 \in \{rew^{\alpha}_1 \mid \alpha \in \Gamma\} \cup \{id\}$ and $op_2 \notin \{rew^{\alpha}_1 \mid \alpha \in \Gamma\}$: hence a transition will always act on the stack by (possibly) rewriting the top symbol and then (possibly) performing another kind of operation on the stack. In the following, we will use notation $(q', op_1; op_2)$ instead of (q', op_1, op_2) (to stress that one performs op_1 followed by op_2).

Remark 2.3. Obviously allowing a top-rewriting operation followed by another stack operation does not add expressive power to the model. However, for technical reasons, this choice simplifies the presentation.

Configurations of an n-CPDA are pairs of the form (q, s) where $q \in Q$ and s is an n-stack over Γ; we call $(q_0, \perp_n)$ the ***initial configuration***.

An n-CPDA $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$ naturally defines a transition graph $\mathrm{Graph}(\mathcal{A}) := (V, E)$ whose vertices V are the configurations of $\mathcal{A}$ and whose edge relation $E \subseteq V\times V$ is given by: $((q, s), (q', s')) \in E$ iff $\exists(q', op_1; op_2) \in \Delta(q, top_1(s))$ such that $s' = op_2(op_1(s))$. Such a graph is called an ***n-CPDA graph***.

Example 2.4. Consider the following 2-CPDA (that actually does not make use of links) $\mathcal{A} = (\{\perp, \alpha\}, \{q_a, q_b, q_c, q_\sharp, \widetilde{q}_a, \widetilde{q}_b, \widetilde{q}_c\}, \Delta, \widetilde{q}_a)$ with Δ as follows (we only give those transitions that may happen):

- $\Delta(\widetilde{q}_a, \perp) = \{(q_a, id; push^{\alpha}_1)\}$
- $\Delta(q_a, \alpha) = \{(q_a, id; push^{\alpha}_1), (\widetilde{q}_b, id; push_2)\}$;
- $\Delta(\widetilde{q}_b, \alpha) = \Delta(q_b, \alpha) = \{(q_b, id; pop_1)\}$;
- $\Delta(q_b, \perp) = \{(\widetilde{q}_c, id; pop_2)\}$;
- $\Delta(\widetilde{q}_c, \alpha) = \Delta(q_c, \alpha) = \{(q_c, id; pop_1)\}$;
- $\Delta(q_c, \perp) = \{(q_\sharp, id; id)\}$;
- $\Delta(q_\sharp, \perp, \_) = \emptyset$.

Then $\mathrm{Graph}(\mathcal{A})$ is given in Figure 1.

2.5 CPDA Parity Games

We now explain how CPDA can be used to define parity games. Let $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$ be an order-n CPDA and let $\mathrm{Graph}(\mathcal{A}) = (V, E)$ be its transition graph. Let $Q_{\mathrm{E}} \uplus Q_{\mathrm{A}}$ be a partition of Q and let $\rho : Q \longrightarrow C \subset \mathbb{N}$ be a colouring function (over states). Altogether they define a partition $V_{\mathrm{E}} \uplus V_{\mathrm{A}}$ of V, whereby a vertex belongs to V_{E} iff its control state belongs to Q_{E}, and a colouring function $\rho : V \longrightarrow C$, where a vertex is assigned the colour of its control state. The structure $\mathcal{G} = (\mathrm{Graph}(\mathcal{A}), V_{\mathrm{E}}, V_{\mathrm{A}})$ defines an arena and the pair $\mathbb{G} = (\mathcal{G}, \Omega_\rho)$ defines a parity game that we call an ***n-CPDA parity game***.

Given an n-CPDA parity game, there are three main algorithmic questions:

$$\begin{tikzcd}
(\widetilde{q}_a, [[\bot]]) \arrow[r] & (q_a, [[\bot\alpha]]) \arrow[r] \arrow[d] & (q_a, [[\bot\alpha\alpha]]) \arrow[r] \arrow[d] & (q_a, [[\bot\alpha\alpha\alpha]]) \arrow[r, dotted] \arrow[d] & {} \\
& (\widetilde{q}_b, [[\bot\alpha][\bot\alpha]]) \arrow[d] & (\widetilde{q}_b, [[\bot\alpha\alpha][\bot\alpha\alpha]]) \arrow[d] & (\widetilde{q}_b, [[\bot\alpha\alpha\alpha][\bot\alpha\alpha\alpha]]) \arrow[d] & \\
& (q_b, [[\bot\alpha][\bot]]) \arrow[d] & (q_b, [[\bot\alpha\alpha][\bot\alpha]]) \arrow[d] & (q_b, [[\bot\alpha\alpha\alpha][\bot\alpha\alpha]]) \arrow[d] & \\
& (\widetilde{q}_c, [[\bot\alpha]]) \arrow[d] & (q_b, [[\bot\alpha\alpha][\bot]]) \arrow[d] & (q_b, [[\bot\alpha\alpha\alpha][\bot\alpha]]) \arrow[d] & \\
(q_\sharp, [[\bot]]) & (q_c, [[\bot]]) \arrow[l] & (\widetilde{q}_c, [[\bot\alpha\alpha]]) \arrow[d] & (q_b, [[\bot\alpha\alpha\alpha][\bot]]) \arrow[d] & \\
& & (q_c, [[\bot\alpha]]) \arrow[ul] & (\widetilde{q}_c, [[\bot\alpha\alpha\alpha]]) \arrow[d] & \\
& & & (q_c, [[\bot\alpha\alpha]]) \arrow[ul] & {} \arrow[l, dotted]
\end{tikzcd}$$

Fig. 1. Transition graph of the CPDA of Example 2.4.

(1) Decide whether $(q_0, \bot_n)$ is winning for Éloïse.
(2) Provide a description of the winning region for Éloïse.
(3) If $(q_0, \bot_n)$ is winning for Éloïse, provide a description of a winning strategy for Éloïse from $(q_0, \bot_n)$.

Remark 2.5. Note that the first question is equivalent to the following one: given a vertex $v \in V$ decide whether v is winning for Éloïse. Indeed, one can always design a new n-CPDA parity game that simulates the original one except that from the initial configuration the players are first forced to go to v, from where the simulation really starts.

To answer the second question, we will introduce the notion of *regular sets of stacks*, and to answer the third one we will consider *strategies realised by n-CPDA transducers*.

2.6 Regular Sets of Stacks with Links

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

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

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

Example 2.6. Consider the stack $s = [[[\perp\alpha]]\,[[\perp][\perp\,\alpha\,\beta\,\gamma]]]$. Then

$$\widetilde{s} = [[[\perp\alpha]]\,[[\perp][\perp\alpha\,\beta\,\gamma]]]$$

and $target(s) = \tau$ where $\tau(5) = 4$, $\tau(14) = 13$, $\tau(15) = 11$ and $\tau(16) = 7$.

We consider *deterministic* finite automata working on such representations of stacks. The automaton reads the word $\widetilde{s}$ from left to right (that is, from bottom to top). 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 of these automata as a deterministic version of Stirling's *dependency tree automata* [34] restricted to words.

Formally, an ***automaton*** is a tuple $(R, A, r_{in}, F, \delta)$ where R is a finite set of states, A is a finite input alphabet, $r_{in} \in R$ is the initial state, $F \subseteq R$ is a set of final states and $\delta : (R\times A)\cup(R\times A\times R) \to R$ is a transition function. With a pair (u, τ) 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 $r_0 \cdots r_n$ as follows:

- $r_0 = r_{in}$;
- for all $0 \le i < n$, $r_{i+1} = \delta(r_i, a_{i+1})$ if $i + 1 \notin Dom(\tau)$;
- for all $0 \le i < n$, $r_{i+1} = \delta(r_i, a_{i+1}, r_{\tau(i+1)})$ 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 recognise 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 (q, s) as the pair $(\widetilde{s} \cdot q, target(s))$ (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 L 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 $(\widetilde{s}, target(s))$ iff $s \in L$. Regular sets of configurations are defined in the same way.

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

Property 2.7. Let L_1, L_2 be regular sets of n-stacks over an alphabet Γ. Then $L_1 \cup L_2$, $L_1 \cap L_2$ and $Stacks(\Gamma) \setminus L_1$ are also regular (here $Stacks(\Gamma)$ denotes the set of all stacks over Γ). The same holds for regular sets of configurations.

Proof. Closure under complement comes from the fact that we consider *deterministic* automata. Closure under union or intersection is achieved by considering a Cartesian product, as in the case of finite automata on finite words. □

The following result shows that the notion of regular sets of n-stacks is robust with respect to the computational model of CPDA. The result is used only when discussing consequences in Section 8.1 and therefore its proof can safely be skipped by the reader.

Theorem 2.8. *Let $\mathcal{A}$ be an order-n CPDA with a state-set Q and a stack alphabet Γ, and let L be a regular set of configurations.*

Then, one can build an order-n CPDA $\mathcal{A}'$ with a state-set Q', a subset $F \subseteq Q'$ and a mapping $\chi : Q' \to Q$ such that the following holds.

(1) *Restricted to the reachable configurations from their respective initial configuration, the transition graph of $\mathcal{A}$ and $\mathcal{A}'$ are isomorphic.*

(2) *For every configuration (q, s) of $\mathcal{A}$ that is reachable from the initial configuration, the corresponding configuration (q', s') of $\mathcal{A}$ is such that $q = \chi(q')$ and belongs (q, s) belongs to L if and only if $q' \in F$.*

Proof. Fix an order-n CPDA $\mathcal{A}$ and an automaton $\mathcal{B} = (R, \Gamma \cup \{[,]\}, r_{in}, F, \delta)$ accepting L.

Let s be an order-n stack. Let $0 \leq k \leq n$ and let t be the topmost k-stack of s, *i.e.* $t = top_{k+1}(s)$. We are interested in describing how $\mathcal{B}$ behaves when reading $pop_k(t)$ (for some technical reason we do not care of the topmost $(k-1)$-stack in t as we will later compose those behaviours), with the convention that $pop_0(t) = t$. If there was no link, this behaviour could simply be described as a function from R into R. However, as we extracted t from s, there may be some "dangling link" of order greater than k.

We refer to Figure 2 for an illustration of the concepts below for the case where $n = 4$. To retrieve the states attached to the respective targets of the links (of order $n, \cdots, k+1$ respectively) in s, we will use as a parameter $n-k$ states $r_n, \cdots, r_{k+1}$ in R. For n-links, we consider the run induced by reading s starting from r_n and this gives the values for the respective targets of the n-links. For $(n-1)$-links, we consider the run induced by reading $top_n(s)$ starting from r_{n-1} (note that states in dangling n-links are known thanks to r_n from the previous step) and this gives the values for the respective targets of the $(n-1)$-links. And so on until we consider, for $(k+1)$-links, the run induced by reading $top_{k+2}(s)$ starting from r_{k+1} (note that states in dangling i-links for $i > k$ are known thanks to r_i) and this gives the values for the respectives targets of the $(k+1)$-links.

Hence, we associate with t a function $\tau_k : R^{n-k} \to (R \to R)$ such that $\tau_k(r_n, \ldots, r_{k+1})$ defines a function from R into R that maps every state $r \in R$ to the state $\tau_k(r_n, \ldots, r_{k+1})(r)$ that is reached by $\mathcal{B}$ when reading $pop_k(t)$ starting from r and where the states attached to the respective targets of the links are determined by $r_n, \cdots, r_{k+1}$ as explained above.

A stack symbol of the CPDA $\mathcal{A}'$, is a pair, consisting of a stack symbol of $\mathcal{A}$, and an $(n+1)$-tuple of the form $(\tau_n, \cdots, \tau_0)$ where the τ_is are as above.

As the function τ_k describes the behaviour of $pop_k(top_{k+1}(s))$, if we want to reconstruct the behaviour of $top_{k+1}(s)$ we need to compose, in the appropriate way, the various τ_i function for $i \leq k$ which leads the following definition. We define $\tau_0^+(r_n \cdots r_1)$ to be the same function as $\tau_0(r_n \cdots, r_1)$; and for each $1 \leq k \leq n$,

$$\tau_k^+(r_n \cdots r_{k+1}) : \begin{cases} R \to R \\ r \mapsto \tau_{k-1}^+(r_n \cdots r_k)(\tau_k(r_n \cdots r_{k+1})(r)) \end{cases}$$

.

Hence, each τ_k^+ is a function from R to R induced by reading (the segment of) s starting from $top_{k+1}(s)$. As each τ_k^+ can be obtained from the τ_is, we safely assume that we can access them directly in $\mathcal{A}'$ when reading the top_1 element of the stack. Note that, considering τ_n^+ applied to the initial state r_{in} of $\mathcal{B}$ we deduce whether the current stack is accepted by $\mathcal{B}$: hence this information will be maintained, together with a state from Q, in the control state of $\mathcal{A}'$ and is used to define F. The function χ is the one erasing all auxiliary informations used by $\mathcal{A}'$ in its control state.

We now explain how $\mathcal{A}'$ behaves. Assume that the topmost stack symbol is $(a, (\tau_n, \cdots, \tau_0))$ and that the $\mathcal{A}$-state stored is q. Then, the possible transitions of $\mathcal{A}'$ mimic the ones of $\mathcal{A}$ when being in state q with topmost stack symbol a. For each order-n stack operation op of $\mathcal{A}$, we define the corresponding stack operation of $\mathcal{A}'$:

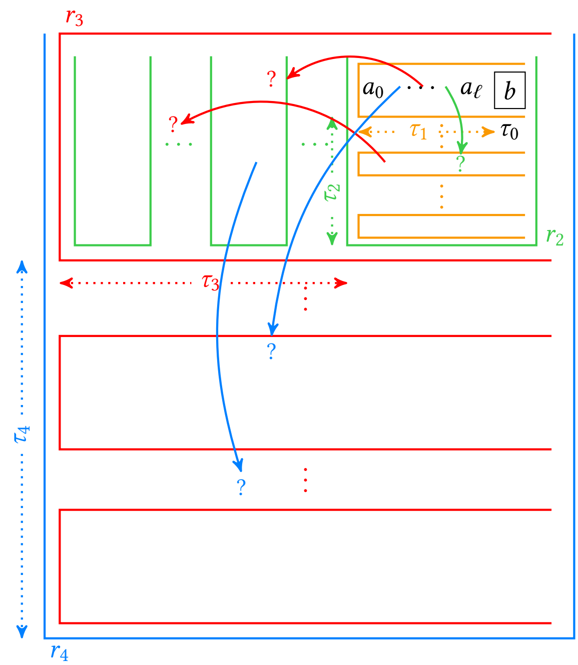

Fig. 2. Illustration for the proof of Theorem 2.8 when $n = 4$. Missing states (?) in k-link's target are retrieve by reading $top_{k+1}(s)$ from r_k. For every k, τ_k^+ is obtained by composing the τ_is for $i \leq k$.

- If $op = push_k$ then $\mathcal{A}'$ performs $push_k$ followed by $rew_1^{a,(\tau_n,\cdots,\tau_{k+1},\tau,\tau_{k-1},\cdots,\tau_0)}$, where for every $r \in R$, $\tau(r_n, \cdots, r_{k+1})(r) = \delta(\tau_{k-1}^+(r_n, \cdots, r_{k+1}, r)(r'),]_k)$ with $r' = \tau_k(r_n, \cdots, r_{k+1})(r)$. Indeed, after performing a $push_k$ operation the only top_i stack that is different from the one before, is for $i = k$. Hence, one only needs to update τ_k, which now maps a state r to the state r' obtained by first applying the previous τ_k followed by the transformation induced by the former top $k-1$-stack (with the missing k-links being retrieve starting from r) together with the missing closing parenthesis $]_k$.
- If $op = push_1^{b,k}$ then $\mathcal{A}'$ performs $push_1^{(b,(\tau_n,\cdots,\tau_2,\tau,\tau_b)),k}$ where τ and τ_b are defined as follows. The function τ is equal to τ_1^+ while the function $\tau_b(r_n, \ldots, r_1)$ maps a state r to $\delta(r, b, \tau_k(r_n, \ldots, r_{k+1})(r_k))$. Indeed, one simply has to update τ_1 and τ_0. Regarding τ_1 one needs now to take into the former topmost symbol which is exactly what does τ_1^+. For τ_0 one simulates the behaviour of $\mathcal{B}$ when reading a b and uses τ_k with the appropriate parameters to retrieve the state in the target of the newly created link.
- If $op = pop_k$ (*resp. collapse* following a k-link) then $\mathcal{A}'$ performs pop_k (*resp. collapse*), considers the new topmost stack symbol $(a', (\tau_n', \cdots, \tau_0'))$ and does a $rew_1^{(a',(\tau_n,\cdots,\tau_{k+1},\tau_k',\cdots\tau_0'))}$. Indeed, for any stack s and any $i > k$, $pop_i(top_{i+1}(s)) = pop_i(top_{i+1}(pop_k(s)))$ and therefore $\tau_n, \cdots, \tau_{k+1}$ are inherited from the previous configuration while the other components are preserved from the last time where (possibly a copy of) the topmost symbol was on top of the stack (being inductively assumed to be correct).

Correctness of the construction follows inductively from the above definition. ◻

2.7 CPDA strategies

Let $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$ be an order-n CPDA, let $\text{Graph}(\mathcal{A}) = (V, E)$ be its transition graph, let $\mathcal{G} = (\text{Graph}(\mathcal{A}), V_{\text{E}}, V_{\text{A}})$ be an arena associated with $\mathcal{A}$ and let $\mathbb{G} = (\mathcal{G}, \Omega_\rho)$ be a corresponding n-CPDA parity game.

We aim at defining a notion of n-CPDA *transducers* that provide a description for strategies in $\mathbb{G}$, that is the transducer describes a function from partial plays in $\mathbb{G}$ into V.

Consider a partial play $\lambda = v_0 v_1 \cdots v_\ell$ in $\mathbb{G}$ where $v_0 = (q_0, \perp_n)$. An alternative description of λ is by a sequence $(q_1, rew_1; op_1) \cdots (q_\ell, rew_\ell; op_\ell) \in (Q \times Op_n^\Gamma \times Op_n^\Gamma)^*$ such that $v_i = (q_i, s_i)$ for all $1 \leq i \leq \ell$ and $s_i = op_i(rew_i(s_{i-1}))$ (with the convention that $s_0 = \perp_n$). We may in the following use implicitly this representation of λ when needed. Similarly, one can represent a strategy as a (partial) function

$$\varphi : (Q \times Op_n^\Gamma \times Op_n^\Gamma)^* \to Q \times Op_n^\Gamma \times Op_n^\Gamma$$

the meaning being that in a partial play λ ending in some vertex (q, s) if $\varphi(\lambda) = (q', rew; op)$ then the player moves to $(q', op(rew(s)))$.

An ***n-CPDA transducer realising a strategy*** in $\mathbb{G}$ is a tuple $\mathcal{S} = (\Sigma, R, \delta, \tau, r_0)$ where Σ is a stack alphabet, R is a finite set of states, $r_0 \in R$ is the initial state,

$$\delta : R \times \Sigma \times (Q \times Op_n^\Gamma \times Op_n^\Gamma) \to R \times Op_n^\Sigma \times Op_n^\Sigma$$

is a *deterministic* transition function and

$$\tau : R \times \Sigma \to Q \times Op_n^\Gamma \times Op_n^\Gamma$$

is a deterministic choice function (note that we do not require τ to be total). For both δ and τ we have the same requirement as for the transition function for CPDAs, namely that the first stack operation should be a top-rewriting (or the identity) and that the second one should not be a top-rewriting.

A configuration of $\mathcal{S}$ is a pair (r, t) where r is a state and t is an n-stack over Σ; the initial configuration of $\mathcal{S}$ is $(r_0, \perp_n)$. With a configuration (r, t) is associated, when defined, a (unique) move in $\mathbb{G}$ given by $\tau(r, top_1(t))$. A partial play $\lambda = (q_1, rew_1; op_1) \cdots (q_\ell, rew_\ell; op_\ell)$ in $\mathbb{G}$ induces a (unique, when defined) ***run*** of $\mathcal{S}$ which is the sequence

$$(r_0, t_0)(r_1, t_1) \cdots (r_\ell, t_\ell)$$

where $(r_0, t_0) = (r_0, \perp_n)$ is the initial configuration of $\mathcal{S}$ and for all $0 \leq i \leq \ell - 1$ one has $\delta(r_i, top_1(t_i), (q_{i+1}, rew_{i+1}; op_{i+1})) = (r_{i+1}, rew'_{i+1}; op'_{i+1})$ with $t_{i+1} = op'_{i+1}(rew'_{i+1}(t_i))$. In other words, the control state and the stack of $\mathcal{S}$ are updated accordingly to δ.

We say that $\mathcal{S}$ is ***synchronised*** with $\mathcal{A}$ iff for all $(r, \alpha, (q, rew; op)) \in R \times \Sigma \times (Q \times Op_n^\Gamma \times Op_n^\Gamma)$ such that $\delta(r, \alpha, (q, rew; op)) = (r', rew'; op')$ is defined one has that op and op' are of the same kind, *i.e.* either they are both a pop_k (for the same k) or both a $push_k$ (for the same k) or both a $push_1^{\rightarrow e}$ (the symbol pushed being possibly different but the order of the link being the same) or both *collapse* or both *id*. In particular, if one defines the ***shape*** of a stack s as the stack obtained by replacing all symbols appearing in s by a fresh symbol $\sharp$ (but keeping the links) one has the following.

Proposition 2.9. *Assume that $\mathcal{S}$ is synchronised with $\mathcal{A}$. Then, for any partial play λ in $\mathbb{G}$ ending in a configuration with stack s, the run of $\mathcal{S}$ on λ, when exists, ends in a configuration with stack t such that s and t have the same shape.*

The ***strategy realised by $\mathcal{S}$*** is the (partial) function $\varphi_\mathcal{S}$ defined by letting $\varphi_\mathcal{S}(\lambda) = \tau((r, top_1(t)))$ where (r, t) is the last configuration of the run of $\mathcal{S}$ on λ.

We say that $\varphi_{\mathcal{S}}$ is ***well-defined*** iff for any partial play $\lambda = (q_1, rew_1; op_1) \cdots (q_\ell, rew_\ell; op_\ell)$ where Éloïse respects $\varphi_{\mathcal{S}}$ whenever the last vertex (q_ℓ, s_ℓ) in λ belongs to V_{E} one has $\varphi_{\mathcal{S}}(\lambda) \in \Delta(q, top_1(s_\ell))$, *i.e.* the move given by $\varphi_{\mathcal{S}}$ is a valid one.

3 MAIN RESULT

The following theorem is the central result of this paper.

THEOREM 3.1. *Let $\mathcal{A} = (\Gamma, Q, \delta, q_0)$ be an n-CPDA and let $\mathbb{G}$ be an n-CPDA parity game defined from $\mathcal{A}$. Then one has the following results.*

(1) *Deciding whether $(q_0, \perp_n)$ is winning for Éloïse is an n-EXPTIME-complete problem.*
(2) *The winning region for Éloïse* (resp. *for Abelard) is regular. Moreover, one can compute an automaton that recognises it.*
(3) *If $(q_0, \perp_n)$ is winning for Éloïse then one can effectively construct an n-CPDA transducer $\mathcal{S}$* synchronised *with $\mathcal{A}$ realising a well-defined* winning *strategy S for Éloïse in $\mathbb{G}$ from $(q_0, \perp_n)$.*

The proof is by induction on the order and each induction step is itself divided into three steps: the first one is a normalisation result (Section 4), the second one removes the outermost links (Section 5) while the third one lowers the order (Section 6). Finally Section 7 combines the previous constructions and provides the proof of Theorem 3.1.

4 RANK-AWARE CPDA

Intuitively, a CPDA is "rank-aware" whenever, during any run of the CPDA, one can easily determine the smallest colour seen since the creation of the link on the topmost symbol. In particular, one only needs to inspect the current control state and topmost stack symbol. This information will be crucial in the next section when we show how to remove the outermost links from a CPDA. In this section, we show that any CPDA can be transformed into an equivalent rank-aware CPDA. The notion of equivalence is formalised in the statement of Theorem 4.8.

Fix, for the whole section, an n-CPDA $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$, a partition $Q_{\mathrm{E}} \uplus Q_{\mathrm{A}}$ of Q and a colouring function $\rho : Q \to C \subset \mathbb{N}$. Denote by G its transition graph, by $\mathcal{G}$ the arena induced by G and the partition $Q_{\mathrm{E}} \uplus Q_{\mathrm{A}}$ and by $\mathbb{G}$ the parity game $(\mathcal{G}, \Omega_\rho)$.

4.1 Definitions

Our main goal in this sub-section is to define the notion of rank-awareness. To do this we will define the notion of *link-rank*. Assume that in configuration v_m the top_1-element has a link (that is possibly a copy of a link) that was created in configuration v_j: then the link-rank in v_m is defined as the smallest colour since the creation of the link, *i.e.* $\min\{\rho(v_j), \cdots \rho(v_m)\}$. Ultimately, we will show how to enrich the stack alphabet to be able to compute the link-rank. In order to maintain this information, we need to define several other concepts. First we will define indexed stacks, from which, we can then define the *collapse-rank* (for updating after performing a *collapse*) and the *pop-rank* for k (for updating after performing a pop_k).

A ***finite path*** in G is a non-empty sequence of configurations $v_0 v_1 \cdots v_m$ such that for all $0 \leq i \leq m-1$, there is an edge in G from v_i to v_{i+1}. An ***infinite path*** is an infinite sequence of configurations $v_0 v_1 \cdots$ such that for all $i \geq 0$, there is an edge in G from v_i to v_{i+1}. Note that we do not require v_0 to be the initial configuration.

We now define a generalisation of n-stacks called *indexed n-stacks*. Following the same notations as in Section 2.6, a stack s is equivalently described as a pair $(\widetilde{s}, target(s))$ (recall that $\widetilde{s}$ is a well-bracketed word description of s and that $target(s)$ gives the link structure). An ***indexed n-stack*** is described by a triple $(\widetilde{s}, target(s), ind(s))$ where $\widetilde{s} = \widetilde{s}_1 \cdots \widetilde{s}_{|\widetilde{s}|}$ and $target(s)$ are as previously and where $ind(s) : \{1, \ldots, |\widetilde{s}|\} \to \mathbb{N}$ is a partial function that is defined in any position $j < |\widetilde{s}| - n$ such that $\widetilde{s}_j \notin \{[,]\}$. The previous conditions on the domain of $ind(s)$ ensure that any stack symbol in s which is not the topmost one has a value by $ind(s)$ that we refer to as its ***index***. An ***indexed configuration*** is a pair formed by a control state and an indexed stack.

The ***erasure*** of an indexed n-stack $(\widetilde{s}, target(s), ind(s))$ is the n-stack $(\widetilde{s}, target(s))$. We extend the notion of erasure to indexed configurations in the obvious way.

The intended meaning of the index of some symbol in the stack is the following. The index is equal to the largest integer i such that since v_i the symbol no longer appears as a top_1-element. Hence, if one uses the stack to store (and maintain) some information, the index is the moment from which this information was no longer updated. Therefore when some symbol appears again as the top_1-element, one has to update the information by taking into account all that happened since v_i (included).

With any path $\lambda = v_0 v_1 \cdots$, with $v_i = (p_i, s_i)$ for all $i \geq 0$, we inductively associate a sequence of indexed configurations $\lambda' = v'_0 v'_1 \cdots$ such that the following holds.

- The erasure of λ' equals λ (the *erasure* of a sequence of indexed configurations being defined as the sequence of the respective erasures).
- For any indexed configuration $v'_m = (q_m, s'_m)$ the following holds. Let $s'_m = (\widetilde{s'_m}, target(s'_m), ind(s'_m))$, let $\widetilde{s'_m} = x_1 \cdots x_h$, and let j be in the domain of $ind(s'_m)$ and such that $x_{j+1} =]$. Then let $j' > j$ be the largest integer such that $x_k =]$ for all $j+1 \leq k \leq j'$ and let i be the unique integer such that $x_i \cdots x_{j'}$ is well-bracketed. Then, for any $i < k < j'$, if $ind(s'_m)(k)$ is defined, one has $ind(s'_m)(k) \leq ind(s'_m)(j)$, and this inequality is strict if $ind(s'_m)(j) \neq 0$. Intuitively, position j is the topmost symbol of some $(j'-j)$-stack, and any symbol in this stack has an index smaller than the topmost symbol.

The intuitive idea behind the forthcoming definition of λ' is rather simple. The indices are always preserved, so one only cares about new positions in the stack. On doing a $push_k$ the indices of the copied stack are inherited from the original copy. Then when new indices are needed (because a position is no longer the top_1 one, it gets index $m+1$ if the current configuration is v_{m+1}).

Before going to the formal definition, we start with an example.

Example 4.1. In Figure 3, we give an example (at order 3) that illustrates the previous intuitive idea as well as the formal description below (ignore the information on colours for this example). We only describe the indexed stacked (omitting the control states), and indicate the stack operation (but omit the *id* operation). Indices are written as superscripts.

Now, we formally give the construction of λ' (the previously mentioned properties easily follow from the definition). The initial configuration $v'_0 = (p_0, s'_0)$, is obtained by letting $ind(s'_0)$ be the constant (partial) function equal to 0. Assume now that $v'_1 \cdots v'_m$ has been constructed, let $v'_m = (p_m, s'_m)$ with $s'_m = (\widetilde{s}_m, target(s_m), ind(s'_m))$ and let $v_{m+1} = (p_{m+1}, s_{m+1})$ with $s_{m+1} = (\widetilde{s}_{m+1}, target(s_{m+1}))$. We let $v'_{m+1} = (p_{m+1}, s'_{m+1})$ with $s'_{m+1} = (\widetilde{s}_{m+1}, target(s_{m+1}), ind(s'_{m+1}))$ where $ind(s'_{m+1})$ is defined thanks to the following case distinction on which stack oprations have been applied to go from v_m to v_{m+1}.

- A top-rewriting operation (possibly equal to *id*) followed by a $push_1^{\gamma,k}$ operation is applied in configuration v_m. Then all previous indices are inherited and the former top_1-element gets index $m+1$. Formally, $ind(s'_{m+1})(j) = ind(s'_m)(j)$ whenever $j < |\widetilde{s}_m| - n$ and $ind(s'_{m+1})(|\widetilde{s}_m| - n) = m+1$.
- A top-rewriting operation (possibly equal to *id*) followed by a $push_k$ operation is applied.First, all existing indices are preserved, *i.e.* $ind(s'_{m+1})(j) = ind(s'_m)(j)$ whenever j belongs to the domain of $ind(s'_m)$. Then one writes $\widetilde{s}_m$ as $[\cdots[t]]^{n-k+1}$ with t being well-bracketed; hence,

$$
\begin{array}{rll}
 & s'_0 = [[[\bot^0\alpha^0]][[\bot]]] & \mathit{colour} : 3\\
\xrightarrow{push_1^{\beta,1}} & s'_1 = [[[\bot^0\alpha^0]][[\bot^1\beta]]] & \mathit{colour} : 0\\
\xrightarrow{rew_1^{\alpha};push_2} & s'_2 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^1\alpha]]] & \mathit{colour} : 1\\
\xrightarrow{pop_1} & s'_3 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot]]] & \mathit{colour} : 5\\
\xrightarrow{push_1^{\alpha,1}} & s'_4 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha]]] & \mathit{colour} : 3\\
\xrightarrow{push_1^{\beta,2}} & s'_5 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta]]] & \mathit{colour} : 2\\
\xrightarrow{push_3} & s'_6 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^6]][[\bot^1\alpha^2][\bot^4\alpha^5\beta]]] & \mathit{colour} : 4\\
\xrightarrow{push_1^{\gamma,3}} & s'_7 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^6]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^7\gamma]]] & \mathit{colour} : 6\\
\xrightarrow{push_2} & s'_8 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^6]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^7\gamma^8][\bot^4\alpha^5\beta^7\gamma]]] & \mathit{colour} : 5\\
\xrightarrow{pop_1} & s'_9 = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^6]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^7\gamma^8][\bot^4\alpha^5\beta]]] & \mathit{colour} : 6\\
\xrightarrow{collapse} & s'_{10} = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^6]][[\bot^1\alpha]]] & \mathit{colour} : 4\\
\xrightarrow{pop_3} & s'_{11} = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta]]] & \mathit{colour} : 3\\
\xrightarrow{push_1^{\gamma,1}} & s'_{12} = [[[\bot^0\alpha^0]][[\bot^1\alpha^2][\bot^4\alpha^5\beta^{12}\gamma]]] & \mathit{colour} : 2
\end{array}
$$

Fig. 3. Example of a sequence of indexed stacks.

$\widetilde{s}_{m+1} = [\cdots[t'][t']]^{n-k+1}$ where t' is obtained from t by (possibly)changing its last symbol to reflect the top-rewriting operation. Then we let $ind(s'_{m+1})(|\widetilde{s'_m}| - (n-k+1) + j) = ind(s'_m)(|\widetilde{s'_m}| - (n-k+1) - (|t|+2) + j)$ for all $j \geq 1$ such that the second member of the equality is defined: the indices are simply copied from the former top $(k-1)$-stack. Finally, the former top_1-element gets index $m+1$: $ind(s'_{m+1})(|\widetilde{s}_m| - n + k - 3) = m + 1$.

- A top-rewriting operation (possibly equal to *id*) followed by either a pop_k operation or a *collapse* or *id* is applied in configuration v_m in λ. Then all indices are inherited from the previous indexed stack. Formally, $ind(s'_{m+1})(j) = ind(s'_m)(j)$ whenever j belongs to the domain of $ind(s'_{m+1})$.

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

Proposition 4.2. *Let $\Lambda = v_0v_1\cdots$ be a path and $\Lambda' = v'_0v'_1\cdots$ be as above. Let $m \geq 0$, let $s'_m = (\widetilde{s}_m, target(s_m), ind(s'_m))$ be the indexed stack in v'_m. Let j be such that $i = ind(s'_m)(j)$ is defined. If $i > 0$, then $(i-1)$ is the largest integer such that the j-th letter of $\widetilde{s}_m$ is a copy of $top_1(s_{i-1})$. If $i = 0$, there is no i' such that the j-th letter of $\widetilde{s}_m$ is a copy of $top_1(s_{i'})$.*

Proof. Immediate by induction on m and from the definition of λ' from λ. □

Consider a finite path $\lambda = v_0 v_1 \cdots v_m$ 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[1], 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 since its link-ancestor v_j (inclusive) *i.e.* it is $\min\{\rho(v_j), \cdots \rho(v_m)\}$.

Example 4.3. Consider the sequence of indexed stacks given in Figure 3. The link-ancestor of configuration v_8 is configuration v_7 and its link-rank is 5. The link-ancestor of configuration v_{11} is configuration v_5 and its link-rank is 2.

Definition 4.4. An n-CPDA $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$ equipped with a colouring function is ***rank-aware*** from a configuration v_0 if there exists a function $LinkRk : Q \times \Gamma \to \mathbb{N}$ such that for any finite path $\lambda = v_0 v_1 \cdots v_\ell$, the link-rank (if defined) of the configuration $v_\ell = (q, s)$ is equal to $LinkRk(q, top_1(s))$. In other words, the link rank can be retrieved from the control state together with the top_1-element of the stack.

To show that any CPDA can be transformed into a rank-aware CPDA, we need to define the collapse-rank and the pop-rank. First, we introduce the notion of *ancestor*. Fix a finite path $\Lambda = v_0 v_1 \cdots v_m$, let $v_m = (q, s)$ be some configuration in Λ and let x be a symbol in s. Then the ***ancestor*** of x is the configuration v_i where i is the index of x in v'_m (the indexed version of v_m).

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

Example 4.5. Consider the sequence of indexed stacks given in Figure 3 (the colours of the corresponding configurations are indicated on the right part of the figure).

In v'_8 the collapse-ancestor is v'_6 and the collapse-rank is therefore 4. In v'_9 the collapse-ancestor is v'_2 and the collapse-rank is therefore 1.

Next, 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, for any $1 \le k \le n$, the ***pop-ancestor*** for k, when defined, is the ancestor of the top_1-element of $pop_k(s)$ and the ***pop-rank*** for k, when defined, is the smallest colour visited since the pop-ancestor for k (included). In particular, the pop-rank for n is the smallest colour visited since the stack has height at least the height of s.

Example 4.6. Again, consider the sequence of indexed stacks given in Figure 3.

In configuration v'_9 the pop-ancestor (*resp.* pop-rank) for 3 is v'_6 (*resp.* 4), the pop-ancestor (*resp.* pop-rank) for 2 is v'_8 (*resp.* 5) and the pop-ancestor (*resp.* pop-rank) for 1 is v'_5 (*resp.* 2).

In configuration v'_{12} the pop-ancestor (*resp.* pop-rank) for 3 is v'_0 (*resp.* 0), the pop-ancestor (*resp.* pop-rank) for 2 is v'_2 (*resp.* 1) and the pop-ancestor (*resp.* pop-rank) for 1 is $v'_1 2$ (*resp.* 2).

Remark 4.7. To permit that the construction remains uniform if the ancestor of the pointed stack (*resp* the ancestor of the top_1-element of $pop_k(s)$ / the link-ancestor) is v_0, the collapse-rank (*resp* the pop-rank / the link-rank) is simply the smallest colour seen since the beginning of the play.

[1]Formally, one could index links as well: whenever performing, in configuration v_j, a $push_1^{\gamma,e}$, one attaches to the newly created link the index $j + 1$. Later, if the link is copied (by doing a $push_k$ operation) then the index is copied as well.

4.2 Main Result

The next theorem shows that we can restrict our attention to CPDA games where the underlying CPDA is rank-aware.

Theorem 4.8. *For any n-CPDA $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$ and any associated parity game $\mathbb{G}$, one can construct an n-CPDA $\mathcal{A}_{\text{rk}}$ and an associated parity game $\mathbb{G}_{\text{rk}}$ such that the following holds.*

- *There exists a mapping ν from the configurations of $\mathcal{A}$ to that of $\mathcal{A}_{\text{rk}}$ such that:*
  - *for any configuration v_0 of $\mathcal{A}$, $\mathcal{A}_{\text{rk}}$ is rank-aware from $\nu(v_0)$;*
  - *Éloïse has a winning strategy in $\mathbb{G}$ from a configuration v_0 iff she has a winning strategy in $\mathbb{G}_{\text{rk}}$ from $\nu(v_0)$;*
  - *both ν and ν^{-1} preserve regularity of sets of configurations.*
- *If there is an n-CPDA transducer $\mathcal{S}_{\text{rk}}$ synchronised with $\mathcal{A}_{\text{rk}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\text{rk}}$ from $\nu(q_0, \perp_n)$, then one can effectively construct an n-CPDA transducer $\mathcal{S}$ synchronised with $\mathcal{A}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}$ from the initial configuration $(q_0, \perp_n)$.*

4.3 Proof of Theorem 4.8

The proof of Theorem 4.8 is a non-trivial generalisation of [28, Lemma 6.3] (which concerns 2-CPDA) to the general setting of n-CPDA and starting from an arbitrary configuration.

Fix an n-CPDA $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$, a partition $Q_{\text{E}} \uplus Q_{\text{A}}$ 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 $\mathcal{A}_{\text{rk}} = (\Gamma_{\text{rk}}, Q_{\text{rk}}, \Delta_{\text{rk}}, q_{0,\text{rk}})$ such that $Q_{\text{rk}} = Q \times C$ and

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

We define a map ν that associates with any configuration of $\mathcal{A}$ a configuration of $\mathcal{A}_{\text{rk}}$. Let (q, s) be a configuration in $\mathcal{A}$. Then $\nu(q, s) = ((q, \rho(q)), s')$ where s' is obtained by:

- Replacing every internal (*i.e.* that is not the top_1-element) symbol γ 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 $(\gamma, \rho(q), \rho(q), \tau_{\rho(q)}$ if it has an n-link and otherwise by $(\gamma, \rho(q), \dagger, \tau_{\rho(q)})$, where $\tau_{\rho(q)}$ is the constant function assigning to any $1 \leq i \leq n$ the value $\rho(q)$.

We equip $\mathcal{A}_{\text{rk}}$ with a colouring function ρ_{rk} by letting $\rho_{\text{rk}}(q, \theta) = \rho(q)$. Our construction will satisfy the following invariant. Let Λ be a finite path in Graph$(\mathcal{A}_{\text{rk}})$ starting in some configuration $\nu(q, s)$ ending in some configuration $((p, \theta), s)$ then the following holds. First, θ is the minimal colour visited from the beginning of the path. Second, if $top_1(s) = (\alpha, m_c, m_l, \tau)$ then

- m_c is the collapse-rank;
- m_l is the link-rank if it makes sense (*i.e.* if there is an n-link in the current top_1-symbol) or is $\dagger$ otherwise;
- τ is the *pop-rank*: $\tau(i)$ is the pop-rank for i for every $1 \leq i \leq n$.

Trivially, from the definition of ν, the invariant holds at the beginning of the path.

The transition function of $\mathcal{A}_{\text{rk}}$ 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 been created using a $push_1^{\cdot,k}$ or a $rew_1^{\circlearrowleft}$ 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 yet been settled. However, when a $\circlearrowleft$ symbol appears in the top_1-element the various ranks can be easily retrieved as they necessarily equal the smallest colour visited so far (as noted in Remark 4.7): this is why we will compute the minimal colour visited so far in the control state of $\mathcal{A}_{\mathrm{rk}}$.

In order to make the construction more readable, we do not formally describe Δ_{rk} but rather explain how $\mathcal{A}_{\mathrm{rk}}$ behaves. 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 as we will allow to do several stack operation per transition); technical issues about this construction are discussed in Remark 4.9. Note that the description below 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 $i \in \mathbb{N}$.

The intuitive idea is the following. One stores in the stack information on the various ranks, and after performing a pop_k or a $collapse$, one needs to update the information stored in the new top_1-element. Indeed this information has no longer been updated since the ancestor configuration (this was the last time it was on top of the stack). To update it, one uses either the collapse-rank / pop-rank in the previous configuration, which is exactly what is needed for this update.

Assume $\mathcal{A}_{\mathrm{rk}}$ is in configuration $v_\ell = ((q, \theta), s)$ with $top_1(s) = (\alpha, m_c, m_l, \tau)$ and let $v_0 v_1 \cdots v_\ell$ be the beginning of the path of Graph($\mathcal{A}_{\mathrm{rk}}$) where we denote $v_i = ((q_i, \theta_i), s_i)$ (hence $q_\ell = q$ and $s_\ell = s$). For any $(q', rew_1^{\gamma}; op) \in \Delta(q, \alpha)$ (note that the case where no rew_1 is performed corresponds to the case where $\gamma = \alpha$) the following behaviours are those allowed in $((q, \theta), s)$.

(1) Assume $op = pop_k$ for some $1 \le k \le n$, let $pop_k(s) = s'$ and let $top_1(s') = (\alpha', m'_c, m'_l, \tau')$. Then $\mathcal{A}_{\mathrm{rk}}$ can go 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 τ' $\circlearrowleft$;
  (b) $(\alpha', \theta', \dagger, (\theta', \ldots, \theta'))$ if $m'_c = \circlearrowleft$, $m'_l = \dagger$ and τ' $\circlearrowleft$;
  (c) $(\alpha', \min(m'_c, \tau(k), \rho(q')), \min(m'_l, \tau(k), \rho(q')), \tau'')$ otherwise, 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 reaches (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 ancestor of $top_1(pop_k(s))$. Then $x > 0$ as otherwise we would be in case (a) or (b). By Proposition 4.2, it follows that $top_1(pop_k(s)) = top_1(s_{x-1})$, 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 top_1-element of the pointed stack in s': y is also the top_1-element of the pointed stack in s_{x-1}, and moreover $y < x$. Hence, the collapse-rank in $v_{\ell+1}$ is

$$\begin{aligned} &\min\{\rho(q_y), \ldots, \rho(q_{x-1}), \rho(q_x), \ldots, \rho(q_\ell), \rho(q')\} \\ =& \min\{\min\{\rho(q_y), \ldots, \rho(q_{x-1})\}, \min\{\rho(q_x), \ldots, \rho(q_\ell)\}, \rho(q')\} \\ =& \min\{m'_c, \tau(k), \rho(q')\} \end{aligned}$$

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 $i \le k$, $top_1(pop_i(s')) = top_1(s_{x-1})$ 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 $top_1(pop_i(s')) = top_1(pop_i(s))$. 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) Assume $op = collapse$, let k be the order of the link in $top_1(s)$, let $collapse(s) = s'$ and let $top_1(s') = (\alpha', m'_c, m'_l, \tau')$. Then $\mathcal{A}_{\mathrm{rk}}$ can go 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 $\tau' = \circlearrowleft$;
   (b) $(\alpha', \theta', \dagger, (\theta', \ldots, \theta'))$ if $m'_c = \circlearrowleft$, $m'_l = \dagger$ and $\tau' = \circlearrowleft$;
   (c) $(\alpha', \min(m'_c, m_c, \rho(q')), \min(m'_l, m_c, \rho(q')), \tau'')$ otherwise with

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

The proof follows the same line as for the previous case. Cases (a) and (b) correspond to the case where one reaches (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 collapse-ancestor of v_ℓ. Then $x > 0$ as otherwise we would be in case (a) or (b). By induction hypothesis, m'_c, m'_l and τ' give the collapse-rank / link-rank / pop-ranks in v_{x-1}. Moreover the ancestor of the top_1-element 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 collapse-rank in v_{x-1} with $\min\{\rho(q_x), \ldots \rho(q_\ell), \rho(q')\} = \min\{m_c, \rho(q')\}$. 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 $\min\{\rho(q_x), \ldots, \rho(q_\ell), \rho(q')\} = \min\{m_c, \rho(q')\}$.

Let $i \leq k$. The ancestor of $top_1(pop_i(s'))$ is the same as the ancestor of $top_1(pop_i(s_{x-1}))$. Therefore the pop-rank for i in $v_{\ell+1}$ is obtained by taking the minimum of the one in v_{x-1} with $\min\{\rho(q_x), \ldots \rho(q_\ell), \rho(q')\} = \min\{m_c, \rho(q')\}$.

Let $i > k$. Then the ancestor of $top_1(pop_i(s'))$ is the same as the ancestor of $top_1(pop_i(s_\ell))$: indeed the collapse only modifies the top_k stack, in other words $pop_i(collapse(s)) = pop_i(s)$. 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 $\rho(q')$.

(3) Assume $op = push_j$ for some $2 \leq j \leq n$, let $push_j(rew_1^{(\gamma, m_c, m_l, \tau)}(s)) = s'$ and let $top_1(s') = (\gamma, m_c, m_l, \tau)$ (note that $\circlearrowleft$ does not appear in $top_1(s')$). Then, $\mathcal{A}_{\mathrm{rk}}$ 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 $(\gamma, \min(m_c, \rho(q')), \min(m_l, \rho(q')), \tau')$ 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 collapse-ancestor 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 $i \neq j$, the ancestor of $top_1(pop_i(s)')$ and the ancestor of $top_1(pop_i(s'))$ 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 ancestor of $top_1(pop_j(s'))$ 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.

(4) Assume $op = push_1^{\beta,k}$ with $1 \leq k \leq n$, and $\beta \in (\Gamma \setminus \{\perp\})$. Then $\mathcal{A}_{\mathrm{rk}}$ can go to (q', θ'), where $\theta' = \min(\theta, \rho'(q'))$, and apply successively $rew_1^{(\gamma, m_c, m_l, \tau)}$ and $push_1^{(\beta, m'_c, m'_l, \tau'), k}$ 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 that $top_1(pop_i(s')) = top_1(pop_i(s))$ 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 ancestor of $pop_1(s')$ is $v_{\ell+1}$ then the pop-rank for 1 is the current colour, namely $\rho(q')$.

From the previous description (and the included inductive proof) we conclude that, for any configuration v_0 of $\mathcal{A}$, $\mathcal{A}_{\mathrm{rk}}$ is rank-aware from $\nu(v_0)$, where we let $LinkRk((q, (\gamma, m_c, m_l, \tau))) = m_l$.

Remark 4.9. One may object that $\mathcal{A}_{\mathrm{rk}}$ does not fit the definition of n-CPDA. Indeed, in a single transition it can do a top-rewriting followed by another stack operation and followed again by a top-rewriting (which itself depends on the new top_1-element). One could add intermediate states and simply decompose such a transition into two transitions, but this would be problematic later when defining an n-CPDA transducer realising a winning strategy.

Fortunately, one can define a variant $\mathcal{A}'_{\mathrm{rk}}$ of $\mathcal{A}_{\mathrm{rk}}$ that has the same properties as $\mathcal{A}_{\mathrm{rk}}$ and additionally fits the definition of n-CPDA. The idea is simply to postpone the final top-rewriting to the next transition. Indeed, it suffices to add a new component on the control state where one encodes the top-rewriting that should be performed next: this top-rewriting is then performed in the next transition (note that this fits the definition as performing two top-rewriting is the same as doing only the last one). However, there is still an issue as the top-rewriting was actually depending on the top_1-symbol (one updates the various ranks) hence, one cannot save the next top-rewriting in the control state without first observing the symbol to be rewritten. Again this is not a real problem, as it suffices to remember which kind of update should be done (one concerning a pop_k or one concerning a *collapse*) and to store in the control state the various objects needed for this update (for this, one can simply store the former top_1-element).

One also needs to slightly modify the *LinkRk* function so that it returns the link-rank of the top_1-symbol after it is rewritten. This can easily be done as the domain of *LinkRk* is $Q_{\mathrm{rk}} \times \Gamma_{\mathrm{rk}}$.

Note that $\mathcal{A}'_{\mathrm{rk}}$ and $\mathcal{A}_{\mathrm{rk}}$ use the same stack alphabet, but that the state space of $\mathcal{A}'_{\mathrm{rk}}$ uses an extra component of size linear in the one of the stack alphabet.

In conclusion building a rank-aware (valid) n-CPDA from a non-aware one increases (by a multiplicative factor) the stack alphabet by $|C|^{n+3}$ and the state set by $O(|C|^{n+3})$.

For now on, we uses $\mathcal{A}_{\mathrm{rk}}$ to mean $\mathcal{A}'_{\mathrm{rk}}$.

We are now ready to conclude the proof of Theorem 4.8. First recall that we defined ρ_{rk} by letting $\rho_{\mathrm{rk}}(q, \theta) = \rho(q)$. Then, we define a partition $Q_{\mathrm{rk,E}} \uplus Q_{\mathrm{rk,A}}$ of Q_{rk} by letting the states in $Q_{\mathrm{rk,E}}$ be those states with their first component in Q_{E}, and those states in $Q_{\mathrm{rk,A}}$ be those states with their first component in Q_{A}. Let $\mathcal{G}_{\mathrm{rk}}$ be the corresponding arena and let $\mathbb{G}_{\mathrm{rk}} = (\mathcal{G}_{\mathrm{rk}}, \Omega_{\rho_{\mathrm{rk}}})$ be the corresponding n-CPDA parity game.

Consider the projection ζ defined from configurations of $\mathcal{A}_{\mathrm{rk}}$ into configurations of $\mathcal{A}$ by only keeping the first component of the control state, and by only keeping the Γ part of the symbols in the stack. Note that, on the domain of ν^{-1}, ζ and ν^{-1} coincide. Also note that ζ preserves the shape of stacks[2], *i.e.* for any configuration v_{rk}, the stack in v_{rk} has the same shape as the stack in $\nu(v_{\mathrm{rk}})$.

We extend ζ as a function from (possibly partial) plays in $\mathbb{G}_{\mathrm{rk}}$ into (possibly partial) plays in $\mathbb{G}$ by letting $\zeta(v'_0 v'_1 \cdots) = \zeta(v'_0)\zeta(v'_1)\cdots$. It is obvious that for any play λ' in $\mathbb{G}_{\mathrm{rk}}$ starting from $\nu(v_0)$, its image $\zeta(\lambda')$ is a play in $\mathbb{G}$ starting from v_0; moreover these two plays induce the same sequence of colours and at any round the player that controls the current configuration is the same in both plays. Conversely, from the definition of $\mathcal{A}_{\mathrm{rk}}$ it is also clear that there is, for any play λ in $\mathbb{G}$ starting from v_0, a *unique* play λ' in $\mathbb{G}_{\mathrm{rk}}$ starting from $\nu(v_0)$ such that $\zeta(\lambda') = \lambda$.

In particular, ζ can be used to construct a strategy in $\mathbb{G}$ from a strategy in $\mathbb{G}_{\mathrm{rk}}$. Indeed, let φ_{rk} be a strategy for Éloïse from $\nu(v_0)$ in $\mathbb{G}_{\mathrm{rk}}$. We define a strategy φ in $\mathbb{G}$ from $\nu(v_0)$. This strategy maintains as a memory a partial play λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$ such that, if Éloïse respects φ, in $\mathbb{G}$ starting from v_0 after having played λ one has $\zeta(\lambda_{\mathrm{rk}}) = \lambda$ and moreover λ_{rk} is a play in $\mathbb{G}_{\mathrm{rk}}$ starting from $\nu(v_0)$ where Éloïse respects φ_{rk}. Initially, we let $\lambda_{\mathrm{rk}} = \nu(v_0)$. Assume that we have been playing λ and that Éloïse has to play next. Then she considers $v_{\mathrm{rk}} = \varphi_{\mathrm{rk}}(\lambda_{\mathrm{rk}})$ and she plays to v where $v = \zeta(v_{\mathrm{rk}})$. Finally one updates λ_{rk} to be $\lambda_{\mathrm{rk}} \cdot v_{\mathrm{rk}}$. If it is Abelard that has to play next and if he moves to some v, then Éloïse updates λ_{rk} to be $\lambda_{\mathrm{rk}} \cdot v_{\mathrm{rk}}$ where v_{rk} is the unique configuration such that $\lambda_{\mathrm{rk}} \cdot v_{\mathrm{rk}}$ is a valid play and such that $\zeta(v_{\mathrm{rk}}) = v$. A similar construction can be done to build a strategy of Abelard in $\mathbb{G}$ from one in $\mathbb{G}_{\mathrm{rk}}$.

Now, assume that $\nu(v_0)$ is winning for Éloïse (*resp.* Abelard) and call φ_{rk} an associated winning strategy. Let φ be the strategy in $\mathbb{G}$ obtained as explained above. Then φ is winning for Éloïse (*resp.* Abelard) in $\mathbb{G}$ from v_0 (this follows directly from the fact that φ_{rk} is winning and that we have the property that $\zeta(\lambda_{\mathrm{rk}}) = \lambda$ for any partial play λ in $\mathbb{G}$ consistent with φ). Hence this proves that Éloïse has a winning strategy in $\mathbb{G}$ from v_0 iff she has a winning strategy in $\mathbb{G}_{\mathrm{rk}}$ from $\nu(v_0)$.

The fact that both ν and ν^{-1} preserve regular sets of configurations is obvious: for this one basically needs to simulate an automaton on the image by ν (or ν^{-1}) that can be computed on-the-fly (except for the very last steps of ν where one needs to know the control state before deducing the top_1 stack element as it has 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).

Finally, from the previous construction of a strategy φ from a strategy φ_{rk} we prove that if there is an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$ synchronised with $\mathcal{A}_{\mathrm{rk}}$ realising a well-defined winning strategy φ_{rk} for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ from $\nu(q_0, \perp_n)$, then one can effectively construct an n-CPDA transducer $\mathcal{S}$ synchronised with $\mathcal{A}$ realising a well-defined winning strategy φ for Éloïse in $\mathbb{G}$ from the initial configuration $(q_0, \perp_n)$. Indeed, in our previous construction of φ, we maintained a partial play λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$ and used the value of $\varphi_{\mathrm{rk}}(\lambda_{\mathrm{rk}})$ to define $\varphi(\lambda)$. But if φ_{rk} is realised by an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$, it suffices to remember the configuration of this transducer after playing λ_{rk} (as this suffices to compute $\varphi_{\mathrm{rk}}(\lambda_{\mathrm{rk}})$). Hence, to obtain $\mathcal{S}$ from $\mathcal{S}_{\mathrm{rk}}$ one needs to "embed" the transition function of $\mathcal{A}_{\mathrm{rk}}$ into it, so that $\mathcal{S}$ can read/output elements in $Q \times Op_n^{\Gamma} \times Op_n^{\Gamma}$ instead of $Q_{\mathrm{rk}} \times Op_n^{\Gamma_{\mathrm{rk}}} \times Op_n^{\Gamma_{\mathrm{rk}}}$. This can easily (but writing the formal construction would be quite heavy) be achieved by noting that the shape of stacks is preserved by ζ: hence if $\mathcal{S}_{\mathrm{rk}}$ is synchronised with $\mathcal{A}_{\mathrm{rk}}$ then $\mathcal{S}$ is synchronised with $\mathcal{A}$ (as $\mathcal{A}_{\mathrm{rk}}$ and $\mathcal{A}$ are "synchronised", and $\mathcal{S}_{\mathrm{rk}}$ and $\mathcal{S}$ are "synchronised" as well).

4.4 Complexity

If we summarise, the overall blowup in the transformation from $\mathbb{G}$ to $\mathbb{G}_{\mathrm{rk}}$ given by Theorem 4.8 is as follows.

[2] Recall that the *shape* of a stack is the stack obtained by replacing all non-$\perp$ symbols appearing in s by a fresh dummy symbol $\sharp$ (but keeping the links).

Proposition 4.10. *Let $\mathcal{A}$ and $\mathcal{A}_{\mathrm{rk}}$ be as in Theorem 4.8. Then the set of states of $\mathcal{A}_{\mathrm{rk}}$ has size $O(|Q|(|C|+1)^{n+3})$ and the stack alphabet of $\mathcal{A}_{\mathrm{rk}}$ has size $O(|\Gamma|(|C|+1)^{2n+5})$. Moreover the set of colours used in $\mathbb{G}$ and $\mathbb{G}_{\mathrm{rk}}$ are the same.*

Proof. By construction together with Remark 4.9. □

5 REMOVING THE n-LINKS

5.1 Main Result

In this section, we show how one can remove the outmost (*i.e.* order-n) links. In the following lf intended to mean *link-free*.

Theorem 5.1. *For any* rank-aware *n-CPDA $\mathcal{A}_{\mathrm{rk}} = (\Gamma_{\mathrm{rk}}, Q_{\mathrm{rk}}, \Delta_{\mathrm{rk}}, q_{0,\mathrm{rk}})$ and any associated parity game $\mathbb{G}_{\mathrm{rk}}$, one can construct an n-CPDA $\mathcal{A}_{\mathrm{lf}}$ and an associated parity game $\mathbb{G}_{\mathrm{lf}}$ such that the following holds.*

- *$\mathcal{A}_{\mathrm{lf}}$ does not create n-links.*
- *There exists a mapping ν from the configurations of $\mathcal{A}_{\mathrm{rk}}$ to that of $\mathcal{A}_{\mathrm{lf}}$ such that:*
  - *Éloïse has a winning strategy in $\mathbb{G}_{\mathrm{rk}}$ from a configuration v_0 iff she has a winning strategy in $\mathbb{G}_{\mathrm{lf}}$ from $\nu(v_0)$;*
  - *If the set of winning configurations for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ is regular, then the set of winning configurations for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ is regular as well.*
- *If there is an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ synchronised with $\mathcal{A}_{\mathrm{lf}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ from $\nu(q_{0,\mathrm{rk}}, \perp_n)$, then one can effectively construct an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$ synchronised with $\mathcal{A}_{\mathrm{rk}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ from the initial configuration $(q_{0,\mathrm{rk}}, \perp_n)$.*

The whole section is devoted to the proof of Theorem 5.1 and we thus fix from now on, a *rank-aware* n-CPDA $\mathcal{A}_{\mathrm{rk}} = (\Gamma_{\mathrm{rk}}, Q_{\mathrm{rk}}, \Delta_{\mathrm{rk}}, q_{0,\mathrm{rk}})$ (together with a function *LinkRk*), a partition $Q_{\mathrm{rk,E}} \uplus Q_{\mathrm{rk,A}}$ of Q_{rk}, a colouring function $\rho : Q_{\mathrm{rk}} \to C \subset \mathbb{N}$ and we let $C = \{0, \ldots, d\}$. Denote by G_{rk} the transition graph of $\mathcal{A}_{\mathrm{rk}}$, by $\mathcal{G}_{\mathrm{rk}}$ the arena induced by G_{rk} and the partition $Q_{\mathrm{rk,E}} \uplus Q_{\mathrm{rk,A}}$, and by $\mathbb{G}_{\mathrm{rk}}$ the parity game $(\mathcal{G}_{\mathrm{rk}}, \Omega_\rho)$.

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 $\mathbb{G}_{\mathrm{rk}}$ from some vertex v iff she wins in the $\mathbb{G}_{\mathrm{lf}}$ from the configuration $\nu(v)$ for some mapping ν — to be defined — transforming vertices of the first game into vertices of the second one). The first task (see Section 5.2) is simple as the initial n-CPDA defining $\mathbb{G}_{\mathrm{rk}}$ is rank aware and therefore comes with a function *LinkRk* as in Lemma 4.8. The second task (see Section 5.3) is more involved because we have to define ν and to prove that it preserves (arbitrary) winning configurations.

5.2 The Simulation Game: $\mathbb{G}_{\mathrm{lf}}$

We now define $\mathcal{A}_{\mathrm{lf}}$ and the associated game $\mathbb{G}_{\mathrm{lf}}$. We start with an informal description of $\mathcal{A}_{\mathrm{lf}}$ and then formally describe its structure.

The n-CPDA $\mathcal{A}_{\mathrm{lf}}$ *simulates* $\mathcal{A}_{\mathrm{rk}}$ as follows. Assume that the play is in some configuration (q, s) and that the player that controls it wants to simulate a transition $(p, rew_1^\alpha; op) \in \Delta_{\mathrm{rk}}(q, top_1(s))$. In case op is neither of the form $push_1^{\beta,n}$ nor of the form *collapse* with $top_1(s)$ having an n-link then the same transition $(p, rew_1^\alpha; op)$ is available in $\mathcal{A}_{\mathrm{rk}}$ and is performed. The interesting case is when $op = push_1^{\beta,n}$, and it is simulated by $\mathcal{A}_{\mathrm{lf}}$ as follows.

- The control state of $\mathcal{A}_{\mathrm{lf}}$ is updated to be p^β and one performs rew_1^α.

- From p^β, Éloïse has to move to a new control state $p^?$ and can push any symbol of the form $(\alpha, \overrightarrow{R})$ where $\overrightarrow{R} = (R_0, \cdots R_d) \in (2^Q)^{d+1}$. A dummy 1-link is attached (and will never be used for a *collapse*).
- From $p^?$, Abelard has to play and choose between one of the following two options:
  - either go to state p and perform no action on the stack,
  - or pick a state r in some R_i, go to an intermediate new state r^i (of colour i) without changing the stack and from this new configuration go to state r and perform a pop_n action.

The intended meaning of such a decomposition of the $push_1^{\beta,n}$ 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 (or a later copy of it) created by pushing β 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 $\mathcal{A}_{\mathrm{rk}}$ — just before collapsing). Note that the R_i are arbitrary sets because Éloïse does not have full control over the play (and in general cannot force R_i to be a singleton). Then Abelard can either choose to simulate the *collapse* (here state r^i is only used for going through a state of colour i). If he does not want to simulate a *collapse* then one stores $\overrightarrow{R}$ since its truth may be checked later in the play.

Assume that later, in configuration (p', t) one of the two players wants to simulate a transition $(r, rew_1^\beta; collapse)$ involving an n-link. By construction, $top_1(t)$ is necessarily of the form $(\gamma, \overrightarrow{R})$. Then the simulation is done by going to a sink configuration that is winning for Éloïse iff $r \in R_{LinkRk(p,\gamma)}$, *i.e.* Éloïse wins iff her former claim on $\overrightarrow{R}$ was correct.

Formally we let $\mathcal{A}_{\mathrm{lf}} = (\Gamma_{\mathrm{lf}}, Q_{\mathrm{lf}}, \Delta_{\mathrm{lf}}, q_{0,\mathrm{lf}})$ with

- $\Gamma_{\mathrm{lf}} = \Gamma_{\mathrm{rk}} \cup \Gamma_{\mathrm{rk}} \times (2^{Q_{\mathrm{rk}}})^{d+1}$
- $Q_{\mathrm{lf}} = Q_{\mathrm{rk}} \cup \{p^\beta \mid p \in Q_{\mathrm{rk}},\ \beta \in \Gamma_{\mathrm{rk}}\} \cup \{p^? \mid p \in Q_{\mathrm{rk}}\} \cup \{r^i \mid r \in Q_{\mathrm{rk}},\ 0 \leq i \leq d\} \cup \{q_{\mathfrak{t}}, q_{\mathfrak{f}}\}$
- Δ_{lf} is defined as follows, where p, q, r range over Q_{rk}, α, β, γ range over Γ_{rk} and $\overrightarrow{R} = (R_0, \ldots, R_d)$ ranges over $(2^{Q_{\mathrm{rk}}})^{d+1}$.
  - If $(p, rew_1^\alpha; op) \in \Delta_{\mathrm{rk}}(q, \gamma)$ and if op is neither of the form $push_1^{\beta,n}$ nor *collapse*, then $(p, rew_1^\alpha; op) \in \Delta_{\mathrm{lf}}(q, \gamma)$ and $(p, rew_1^{(\alpha,\overrightarrow{R})}; op) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$.
  - If $(p, rew_1^\alpha; push_1^{\beta,n}) \in \Delta_{\mathrm{rk}}(q, \gamma)$, then $(p^\beta, rew_1^\alpha; id) \in \Delta_{\mathrm{lf}}(q, \gamma)$ and $(p^\beta, rew_1^{(\alpha,\overrightarrow{R})}; id) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$.
  - For all $p^\beta \in Q_{\mathrm{lf}}$, $\Delta(p^\beta, \gamma) = \Delta(p^\beta, (\gamma, \overrightarrow{R})) = \{(p^?, push_1^{(\beta,\overrightarrow{S}),1}) \mid \overrightarrow{S} \in (2^{Q_{\mathrm{rk}}})^{d+1})\}$.
  - For all $p^? \in Q_{\mathrm{lf}}$, $\Delta(p^?, (\gamma, \overrightarrow{R})) = \{(p, id)\} \cup \{(r^i, id) \mid 0 \leq i \leq d \text{ and } r \in R_i\}$.
  - For all $r^i \in Q_{\mathrm{lf}}$, $\Delta(r^i, (\gamma, \overrightarrow{R})) = \{(r, pop_n)\}$.
  - If $(p, rew_1^\alpha; collapse) \in \Delta_{\mathrm{rk}}(q, \gamma)$, then $(p, rew_1^\alpha; collapse) \in \Delta_{\mathrm{lf}}(q, \gamma)$.
  - If $(r, rew_1^\alpha; collapse) \in \Delta_{\mathrm{rk}}(q, \gamma)$, then $(q_{\mathfrak{t}}, id) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$ if $r \in R_{LinkRk(q,\gamma)}$ and $(q_{\mathfrak{f}}, id) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$ if $r \notin R_{LinkRk(q,\gamma)}$.
  - $\Delta_{\mathrm{lf}}(q_{\mathfrak{t}}, (\gamma, \overrightarrow{R})) = \{(q_{\mathfrak{t}}, id)\}$ and $\Delta_{\mathrm{lf}}(q_{\mathfrak{f}}, (\gamma, \overrightarrow{R})) = \{(q_{\mathfrak{f}}, id)\}$.

We let G_{lf} be the transition graph of $\mathcal{A}_{\mathrm{lf}}$. Now, in order to define a game graph $\mathcal{G}_{\mathrm{lf}}$ out of G_{lf} we let $Q_{\mathrm{lf,E}} = Q_{\mathrm{rk,E}} \cup \{p^\beta \mid p \in Q_{\mathrm{rk}},\ \beta \in \Gamma_{\mathrm{rk}}\}$. Finally to define a corresponding n-CPDA parity game $\mathbb{G}_{\mathrm{lf}}$ we extend ρ by letting, $\forall p, r \in Q_{\mathrm{rk}}$ and $\beta \in \Gamma_{\mathrm{rk}}$, $\rho(p^\beta) = \rho(p^?) = d$ (as one cannot loop forever in such states, it means that they have no influence on whether a play will be winning or not), $\rho(r^i) = i$ for every $0 \leq i \leq d$, $\rho(q_{\mathfrak{t}}) = 0$ and $\rho(q_{\mathfrak{f}}) = 1$ (hence a play that visits $q_{\mathfrak{t}}$ is winning for Éloïse and a play that visits $q_{\mathfrak{f}}$ is winning for Abelard, as these states are sinks).

Note that $\mathcal{A}_{\text{lf}}$ never creates an n-link.

5.3 Correctness of the Simulation

Consider some configuration $v_0 = (p_0, s_0)$ in $\mathbb{G}_{\text{rk}}$. We explain now how to define an "equivalent" configuration $\nu(v_0)$ in $\mathbb{G}_{\text{lf}}$ (here equivalent is in the sense of Lemma 5.3 below). The transformation consists in replacing any occurrence of a stack letter (call it γ) with an n-link in s_0 by another letter of the form $(\gamma, \overrightarrow{R})$ and replacing the n-link by a 1-link. The vector $\overrightarrow{R}$ is defined as follows. Let s' be the stack obtained by popping every symbol and stack above γ, and let $R = \{q \mid$ Éloïse wins in $\mathbb{G}_{\text{rk}}$ from $(q, collapse(s'))\}$. Then one sets $\overrightarrow{R} = (R, \cdots, R)$.

Example 5.2. Assume we are playing a two-colour parity game and let

$$s_0 = [[[\,a]]\,[[][\,a\,b\,c]]\,[[][\,a\,b\,c\,d]]],$$

$$R_1 = \{r \mid (r, [[[a]]]) \text{ is winning for Éloïse in } \mathbb{G}_{\text{rk}}\}$$

$$R_2 = \{r \mid (r, [[[\,a]]\,[[][\,a\,b\,c]]]) \text{ is winning for Éloïse in } \mathbb{G}_{\text{rk}}\}$$

Then

$$\nu(s_0) = [[[\,a]]\,[[][\,a\,b\,(c,(R_1,R_1))]]\,[[][\,a\,b\,(c,(R_1,R_1))\,(d,(R_2,R_2))]]].$$

The rest of this section is devoted to the proof of the following result.

Lemma 5.3. *Éloïse wins in $\mathbb{G}_{\text{rk}}$ from some configuration v_0 if and only if she wins in $\mathbb{G}_{\text{lf}}$ from $\nu(v_0)$.*

Assume that the configuration $v_0 = (p_0, s_0)$ is winning for Éloïse in $\mathbb{G}_{\text{rk}}$, and let φ_{rk} be a winning strategy for her. Using φ_{rk}, we define a strategy φ_{lf} for Éloïse in $\mathbb{G}_{\text{lf}}$ from $\nu(v_0)$. The strategy φ_{lf} maintains as a memory a partial play λ_{rk} in $\mathbb{G}_{\text{rk}}$, that is an element in V_{rk}^* (where V_{rk} denotes the set of vertices of G_{rk}). At the beginning λ_{rk} is initialised to be (p_0, s_0). The play λ_{rk} will satisfy the following invariant: assume that the play ends in a configuration (q, s), then the last configuration in λ_{rk} has control state q and its top_1-element is either $top_1(s)$ or $(top_1(s), \overrightarrow{R})$ for some $\overrightarrow{R}$ (and in this case there is an n-link from the top_1-symbol of s).

We first describe φ_{lf}, and then we explain how λ_{rk} is updated.

Choice of the move. Assume that the play is in some vertex (q, s) with $q \in Q_{\text{lf,E}} \setminus \{p^\beta \mid q \in Q_{\text{rk}},\ \beta \in \Gamma_{\text{rk}}\}$. The move given by φ_{lf} depends on $\varphi_{\text{rk}}(\lambda_{\text{rk}}) = (p, rew_1^\alpha; op)$ (we shall later argue that φ_{lf} is well defined whilst proving that it is winning).

- If op is neither of the form $push_1^{\beta,n}$ nor $collapse$ then Éloïse plays $(p, rew_1^\alpha; op)$ if $top_1(s) = \gamma$ and she plays $(p, rew_1^{(\alpha,\overrightarrow{R})}; op)$ if $top_1(s) = (\gamma, \overrightarrow{R})$.
- If $op = collapse$ and $top_1(s) = \gamma \in \Gamma_{\text{rk}}$ then Éloïse plays $(p, rew_1^\alpha; collapse)$.
- If $op = collapse$ and $top_1(s) = (\gamma, \overrightarrow{R})$ then Éloïse plays (q_t, id). We shall later see that this move is always valid.
- If $op = push_1^{\beta,n}$ then Éloïse plays $(p^\beta, rew_1^\alpha; id)$ if $top_1(s) = \gamma$ and she plays $(p^\beta, rew_1^{(\alpha,\overrightarrow{R})}; id)$ if $top_1(s) = (\gamma, \overrightarrow{R})$.

In this last case, or in the case where $q \in Q_{\text{A}}$ and Abelard plays some $(p^\beta, rew_1^\alpha; id)$ (*resp.* some $(p^\beta, rew_1^{(\alpha,\overrightarrow{R})}; id)$), we also have to explain how Éloïse behaves from $(p^\beta, rew_1^\alpha(s))$ (*resp.* $(p^\beta, rew_1^{(\alpha,\overrightarrow{R})}(s))$).

Éloïse has to play $(p^?, push_1^{(\beta,\overrightarrow{S}),1})$ where $\overrightarrow{S} \in (2^{Q_{\text{rk}}})^{d+1}$ describes which states can be reached if the n-link created by pushing β (or a copy of it) is used for collapsing the stack, depending on the smallest visited colour in the meantime. In order to define $\overrightarrow{S}$, she considers the set of all possible continuations of $\lambda_{\mathrm{rk}} \cdot (p, push_1^{\beta,n}(t))$ (where (q,t) denotes the last vertex of λ_{rk}) where she respects her strategy φ_{rk}. For each such play, she checks whether some configuration of the form $(r, pop_n(t))$ is eventually reached by collapsing (possibly a copy of the) n-link created by $push_1^{\beta,n}$. If such an r exists, she considers the smallest colour i visited from the moment where the link was created to the moment a *collapse* is performed (*i.e.* the link rank just before collapsing). For every $i \in \{0, \dots d\}$, the set S_i is defined to be the set of states $r \in Q$ such that the preceding case happens. Formally,

$$
\begin{aligned}
S_i = \{r \mid \exists\, \lambda_{\mathrm{rk}} \cdot v_0 \cdots v_k \cdot v_{k+1} \cdots \ & \text{play in } \mathbb{G}_{\mathrm{rk}} \text{ where Éloïse respects } \varphi_{\mathrm{rk}} \text{ and s.t.} \\
v_0 = (p, push_1^{\beta,n}(t)),\ v_{k+1} = (r, pop_n(t)) & \text{ is obtained by applying } collapse \text{ from } v_k, \\
& v_0 \text{ is the link ancestor of } v_k \text{ and } i \text{ is the link rank in } v_k\}
\end{aligned}
$$

Finally, we set $\overrightarrow{S} = (S_0, \dots, S_d)$ and Éloïse plays $(p^?, push_1^{(\beta,\overrightarrow{S}),1})$.

Update of λ_{rk}. The memory λ_{rk} is updated after each visit to a configuration with a control state in $Q_{\mathrm{rk}} \cup \{q_{\mathfrak{t}}, q_{\mathfrak{f}}\}$. We have several cases depending on the transition.

- If the last transition is of the form $(p, rew_1^{\alpha}; op)$ or $(p, rew_1^{(\alpha,\overrightarrow{R})}; op)$ with op being neither of the form $push_1^{\beta,n}$ nor *collapse*, then we extend λ_{rk} by applying transition $(p, rew_1^{\alpha}; op)$, *i.e.* if (q,t) denotes the last configuration in λ_{rk}, then the updated memory is $\lambda_{\mathrm{rk}} \cdot (p, op(rew_1^{\alpha}(t)))$.
- If the last transition is of the form $(q_{\mathfrak{t}}, id)$ or $(q_{\mathfrak{f}}, id)$, the play is in a sink configuration. Therefore we do not update λ_{rk} as the play will loop forever.
- If the last transitions form a sequence of the form $(p^{\beta}, rew_1^{\alpha}; id) \cdot (p^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (p, id)$ or of the form $(p^{\beta}, rew_1^{(\alpha,\overrightarrow{R})}; id) \cdot (p^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (p, id)$, then the updated memory is $\lambda_{\mathrm{rk}} \cdot (p, push_1^{\beta,n}(t))$, where (q,t) denotes the last configuration in λ_{rk}.
- If the last transitions form a sequence of the form $(p^{\beta}, rew_1^{\alpha}; id) \cdot (p^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (r^i, id) \cdot (r, pop_n)$ or of the form $(p^{\beta}, rew_1^{(\alpha,\overrightarrow{R})}; id) \cdot (p^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (r^i, id) \cdot (r, pop_n)$, then we extend λ_{rk} by a sequence of actions (consistent with φ_{rk}) that starts by performing transition $(p, push_1^{\beta,n})$ and ends up by collapsing (possibly a copy of) the link created at this first step and goes to state r whilst visiting i as a minimal colour in the meantime. By definition of $\overrightarrow{S}$ such a sequence always exists. More formally, if (q,t) denotes the last configuration in λ_{rk}, then the updated memory is a play in $\mathbb{G}_{\mathrm{rk}}$, $\lambda_{\mathrm{rk}} \cdot v_0 \cdots v_k \cdot v_{k+1}$, where Éloïse respects φ_{rk} and such that $v_0 = (p, push_1^{\beta,n}(t))$, $v_{k+1} = (r, pop_n(t))$ is obtained by applying *collapse* from v_k, v_0 is the link ancestor of v_k and i is the link rank in v_k.

Therefore, with any partial play λ_{lf} in $\mathbb{G}_{\mathrm{lf}}$ starting from v_0 in which Éloïse respects her strategy φ_{lf}, is associated a partial play λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$. An immediate induction shows that λ_{rk} is a play where Éloïse respects φ_{rk}. The same argument works for any infinite play λ_{lf} that does not contain a state in $\{q_{\mathfrak{t}}, q_{\mathfrak{f}}\}$, and the corresponding play λ_{rk} is therefore infinite, starts from $\nu(v_0)$ and Éloïse respects φ_{rk} in that play. Therefore it is a winning play.

Moreover, if λ_{lf} is an infinite play that does not contain a state in $\{q_{\mathfrak{t}}, q_{\mathfrak{f}}\}$, it easily follows from the definitions of φ_{lf} and λ_{rk} that the smallest infinitely visited colour in λ_{lf} is the same as the one in λ_{rk}. Hence, any infinite play in $\mathbb{G}_{\mathrm{lf}}$ starting from $\nu(v_0)$ where Éloïse respects φ_{lf} and that does not contain a state in $\{q_{\mathfrak{t}}, q_{\mathfrak{f}}\}$ is won by Éloïse.

Now, consider a play that contains a state in $\{q_{\mathrm{t}}, q_{\mathrm{f}}\}$ (hence loops on it forever). Reaching a configuration with state in $\{q_{\mathrm{t}}, q_{\mathrm{f}}\}$ is necessarily by simulating a *collapse* from some configuration with a top_1-element of the form $(\alpha, \overrightarrow{R})$. We should distinguish between those elements $(\alpha, \overrightarrow{R})$ that are "created" before (*i.e.* by the ν function) or during the play (by Éloïse). For the second ones, note that whenever Éloïse wants to simulate a collapse, she can safely go to state q_{t} (meaning φ_{lf} is well defined): indeed, if this was not the case, it would contradict the way $\overrightarrow{S}$ was defined when simulating the original creation of the link. For the same reason, Abelard can never reach state q_{f} provided Éloïse respects her strategy φ_{lf}. Now consider an element $(\alpha, \overrightarrow{R})$ created by ν and assume that one player wants to simulate a collapse from some configuration with such a top_1-element. Call λ_{lf} the partial play just before and call λ_{rk} the associated play in $\mathbb{G}_{\mathrm{rk}}$. Then in λ_{rk}, Éloïse respects her winning strategy φ_{rk}. If she has to play next in λ_{rk}, strategy φ_{rk} indicates to play *collapse*; if it is Abelard's turn to move he can play *collapse*. In both cases, the configuration that is reached after collapsing is winning for Éloïse (it is a configuration visited in a winning play). Hence, by definition of ν, its control state belongs to R where $\overrightarrow{R} = (R, \cdots, R)$, and therefore from the current vertex in $\mathbb{G}_{\mathrm{lf}}$ there is no transition to q_{f} and there is at least one to q_{t}. Therefore plays where Éloïse respects φ_{lf} and that contain a state in $\{q_{\mathrm{t}}, q_{\mathrm{f}}\}$ necessarily contain state q_{t} hence are won by Éloïse.

Altogether, it proves that φ_{lf} is a winning strategy for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ from $\nu(v_0)$.

Let us now prove the converse implication. Assume that the configuration $\nu(v_0)$ is winning for Éloïse in $\mathbb{G}_{\mathrm{lf}}$, and let φ_{lf} be a winning strategy for her. Using φ_{lf}, we define a strategy φ_{rk} for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ from $v_0 = (p_0, s_0)$. First, recall how $\nu(v_0)$ is defined: every symbol γ in s_0 with an n-link is replaced by a pair $(\gamma, (R, \dots, R))$ where R is the set of states r such that Éloïse wins from (r, s') where s' is the stack obtained by first removing every symbol (and stack) above γ and then performing a *collapse*. We can therefore assume that we have a collection of winning strategies, one for each such configuration (r, s'); call such a strategy $\varphi_{\mathrm{rk}}^{r,s'}$. Then, during a play where Éloïse respects φ_{rk}, if one eventually visits such a configuration (r, s'), the strategy φ_{rk} will mimic the winning strategy $\varphi_{\mathrm{rk}}^{r,s'}$ from that point and therefore the resulting play will be winning for Éloïse. Then in the rest of this description we mostly focus on the case of plays where this situation does not occur.

The strategy φ_{rk} maintains as a memory a partial play λ_{lf} in $\mathbb{G}_{\mathrm{lf}}$, that is an element in V_{lf}^* (where V_{lf} denotes the set of vertices of G_{lf}). At the beginning λ_{lf} is initialised to the configuration $\nu(v_0)$. After having played λ_{rk}, the play λ_{lf} will satisfy the following invariant. Assume that the play λ_{lf} ends in a configuration (q, s) then the following holds.

- If $top_1(s) = \alpha$, the last configuration of λ_{rk} has control state q and its top_1-element is α and it has a k-link for some $k < n$.
- If $top_1(s) = (\alpha, \overrightarrow{R})$, the last configuration of λ_{rk} has control state q, its top_1-element is α and it has an n-link. Moreover, if Éloïse keeps respecting φ_{rk} in the rest of the play, if (possibly a copy of) this link is eventually used in a *collapse*, then the state that will be reached just after doing the *collapse* will belong to R_i where i will be the link rank just before collapsing.

We first describe φ_{rk} and we then explain how λ_{lf} is updated. Recall that we switch to a known winning strategy in case we do 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 (q, s) with $q \in Q_{\mathrm{rk,E}}$. The move given by φ_{rk} depends on $\varphi_{\mathrm{lf}}(\lambda_{\mathrm{lf}}) = (q', rew; op)$ (we shall later argue that φ_{rk} is well defined whilst proving that it is winning).

- If $q' \in Q_{\mathrm{rk}}$ then Éloïse plays $(q', rew_1^{\alpha}; op)$ where α is such that either $rew = rew_1^{\alpha}$ or $rew = rew_1^{(\alpha,\overrightarrow{R})}$. Note that in this case, op is neither a *collapse* involving an n-link nor of the form $push_1^{\beta,n}$.
- If $q' = p^{\beta}$ then Éloïse plays to $(p, rew_1^{\alpha}; push_1^{\beta,n})$ where α is such that either $rew = rew_1^{\alpha}$ or $rew = rew_1^{(\alpha,\overrightarrow{R})}$.
- If $q' = q_{\mathrm{t}}$ then Éloïse plays $(r, collapse)$ for some arbitrary $r \in R_{LinkRk(p,top_1(s))}$ where $(\alpha, \overrightarrow{R})$ denotes the top_1-element of the last vertex of λ_{lf}. Note that in this case, the collapse involves an n-link.

Update of λ_{lf}. The memory λ_{lf} is updated after each move (played by any of the two players). We have several cases depending on the last transition.

- If the last transition is of the form $(q', rew_1^{\alpha}; op)$ and op is neither a *collapse* involving an n-link nor of the form $push_1^{\beta,n}$, then λ_{lf} is extended by mimicking the same transition, *i.e.* if (q, t) denotes the last configuration in λ_{lf}, then the updated memory is $\lambda_{\mathrm{lf}} \cdot (q', op(rew_1^{\alpha}(t))$ if $top_1(t) = \gamma$ for some $\gamma \in \Gamma_{\mathrm{rk}}$, and is $\lambda_{\mathrm{lf}} \cdot (q', op(rew_1^{(\alpha,\overrightarrow{R})}(t))$ if $top_1(t) = (\gamma, \overrightarrow{R})$ for some $(\gamma, \overrightarrow{R}) \in \Gamma_{\mathrm{lf}}$.
- If the last transition is of the form $(p, rew_1^{\alpha}; push_1^{\beta,n})$ then, we let (q, t) denote the last configuration in λ_{lf}. If $top_1(t) = \gamma$ for some $\gamma \in \Gamma_{\mathrm{rk}}$ then the updated memory is $\lambda_{\mathrm{lf}} \cdot (p^{\beta}, rew_1^{\alpha}(t)) \cdot (p^{?}, push_1^{(\beta,\overrightarrow{R}),1}(rew_1^{\alpha}(t))) \cdot (p, id)$ where $\varphi_{\mathrm{lf}}(\lambda_{\mathrm{lf}} \cdot (p^{\beta}, rew_1^{\alpha}(t))) = (p^{?}, push_1^{(\beta,\overrightarrow{R}),1}(rew_1^{\alpha}(t)))$. If $top_1(t) = (\gamma, \overrightarrow{S})$ for some $(\gamma, \overrightarrow{S}) \in \Gamma_{\mathrm{lf}}$ then the updated memory is $\lambda_{\mathrm{lf}} \cdot (p^{\beta}, rew_1^{(\alpha,\overrightarrow{S})}(t)) \cdot (p^{?}, push_1^{(\beta,\overrightarrow{R}),1}(rew_1^{(\alpha,\overrightarrow{S})}(t))) \cdot (p, id)$ where $\varphi_{\mathrm{lf}}(\lambda_{\mathrm{lf}} \cdot (p^{\beta}, rew_1^{(\alpha,\overrightarrow{S})}(t))) = (p^{?}, push_1^{(\beta,\overrightarrow{R}),1}(rew_1^{(\alpha,\overrightarrow{S})}(t)))$.
- If the last transition is of the form $(r, collapse)$ and the *collapse* follows an n-link, then we have two cases. In the first case, the *collapse* follows (possibly a copy of) an n-link that was already in s_0 and we claim (and prove later) that one ends up in a winning configuration and thus one switches to a corresponding winning strategy as already explained. In the other case, it follows an n-link that was created during the play, in which case we let $\lambda_{\mathrm{lf}} = v_0 \cdots v_m$ and denote by v_i the link ancestor of v_m[3]. Then the updated memory is obtained by backtracking inside λ_{lf} until reaching the configuration where the (simulation of the) collapsed n-link was created (this configuration is v_i, the link ancestor) and then extending it by a choice of Abelard consistent with the *collapse*. That is the updated memory is $v_0 \cdots v_i \cdot (r^{\ell}, t) \cdot (r, pop_n(t))$ where $v_i = (p^{?}, t)$ and ℓ denotes the link rank in the configuration λ_{rk} was just before doing the *collapse*.

Therefore, with any partial play λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$ in which Éloïse respects her strategy φ_{rk}, is associated a partial play λ_{lf} in $\mathbb{G}_{\mathrm{lf}}$. Note that if we end up in a configuration that is known to be winning, λ_{lf} becomes useless and is no longer extended. This also 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_{lf} be the constructed play before collapsing: then either Éloïse has to play and therefore moves to q_{t} (and therefore the configuration in λ_{rk} after collapsing is winning by definition of ν, leading a contradiction) or Abelard could move to q_{ff} (leading a contradiction with φ_{lf} being

[3]Here we implicitly extend the notion of link ancestor as follows. In $\mathbb{G}_{\mathrm{lf}}$ instead of creating n-link one pushes symbol of the form $(\beta, \overrightarrow{R})$: hence whenever doing a $push_1^{(\beta,\overrightarrow{R}),1}$ one attaches to the vector $\overrightarrow{R}$ the index of the current configuration. Then if the top_1 element of v_n is some $(\beta, \overrightarrow{R})$ 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 necessarily of the form $p^{?}$.

winning). Therefore, from now on, we restrict our attention to the case where the n-links (and their copies) originally in s_0 are never used to do a *collapse*.

An easy induction shows that Éloïse respects φ_{lf} in λ_{lf}. The same argument works for an infinite play λ_{rk}, and the corresponding play λ_{lf} is therefore infinite (one simply considers the limit of the λ_{lf} in the usual way[4]), starts from $\nu(v_0)$, never visits a state in $\{q_{\text{t}}, q_{\text{ff}}\}$ and Éloïse respects φ_{lf} in that play. Therefore it is a winning play.

Now, in order to conclude that any play λ_{rk} in $\mathbb{G}_{\text{rk}}$ in which Éloïse respects strategy φ_{rk} is winning for her, one needs to relate the sequence of colours in λ_{rk} with the one in λ_{lf}. For this, we introduce a notion of factorisation of a partial play $\lambda_{\text{rk}} = v_0 v_1 \cdots v_m$ in $\mathbb{G}_{\text{rk}}$ (we should later note that it directly extends to infinite plays). A factor is a nonempty sequence of vertices of the following kind:

(1) it is a sequence $v_h \cdots v_k$ such that the stack operation from v_{h-1} to v_h is of the form $rew_1^{\alpha}; push_1^{n,\beta}$, the stack operation from v_{k-1} to v_k is a *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_{rk} denoted $Fact(\lambda_{\text{rk}})$ is a sequence of factors inductively defined as follows (we underline factors to make them explicit): $Fact(\lambda_{\text{rk}}) = \underline{v_0 \cdots v_k}, Fact(v_{k+1} \cdots v_n)$ if there exists some k such that $v_0 \cdots v_k$ is as in (1) above, and $Fact(\lambda_{\text{rk}}) = \underline{v_0}, Fact(v_1 \cdots v_n)$ 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 5.4. *Let λ_{rk} be some infinite play in $\mathbb{G}_{\text{rk}}$ starting from v_0 where Éloïse respects φ_{rk} and assume that there is no collapse that follows (possibly a copy of) an n-link already in s_0. Let λ_{lf} be the associated infinite play in $\mathbb{G}_{\text{lf}}$ constructed from φ_{rk}. Let $\lambda_{\text{rk},0}, \lambda_{\text{rk},1}, \cdots$ be the factorisation of λ_{rk} and, for every $i \geq 0$, let c_i be the colour of $\lambda_{\text{rk},i}$.*

Then the sequence $(c_i)_{i\geq 0}$ and the sequence of colours visited in λ_{lf} have the same $\liminf$.

The previous proposition directly implies that φ_{rk} is a winning strategy for Éloïse from v_0 in $\mathbb{G}_{\text{rk}}$.

5.4 Regularity of the Winning Region is Preserved

We established in Lemma 5.3 that Éloïse wins in $\mathbb{G}_{\text{rk}}$ from some configuration v_0 if and only if she wins in $\mathbb{G}_{\text{lf}}$ from $\nu(v_0)$. We now prove that regular sets of winning positions are preserved by inverse image by ν.

Proposition 5.5. *Assume that we have an automaton $\mathcal{B}_{\text{lf}}$ that recognises the set of winning configurations in $\mathbb{G}_{\text{lf}}$. Then, one can compute an automaton $\mathcal{B}_{\text{rk}}$ that recognises the set of winning configurations in $\mathbb{G}_{\text{rk}}$.*

Proof. We can safely assume that any control state of $\mathcal{B}_{\text{lf}}$ is of the form (ξ, R) with $R \subseteq Q_{\text{lf}}$ and such that, after reading some input stack s (possibly with some pending open brackets) $\mathcal{B}_{\text{lf}}$ is in a

[4] Let $(u_m)_{m\geq 0}$ be a sequence of finite words. For any $m \geq 0$ let $u_m = u_{m,0} \cdots u_{m,k_m}$. Then the limit of the sequence $(u_m)_{m\geq 0}$ is the (possibly infinite) word $\alpha = \alpha_0\alpha_1 \cdots$ such that α is maximal for the prefix ordering and for all $0 \leq i < |\alpha|$ there is some N_i such that $u_{m,i} = \alpha_i$ for all $m \geq N_i$.

In our setting, the play λ_{lf} associated with an infinite play λ_{rk} is defined as the limit of the sequence of partial plays $(\lambda_{\text{lf}}^m)_{m\geq 0}$ where λ_{lf}^m is the partial play associated with λ_{rk} truncated to its $m+1$ first vertices. From the definitions of the λ_{lf}^m it is easily verified that the limit λ_{lf} is infinite.

state of the form (ξ, R) with $R = \{r \mid \mathcal{B}_{\mathrm{lf}} \text{ accepts } (r, s')\}$ where s' is the stack obtained from s by closing all the pending open brackets (*i.e.* $s' = s]^k$ for some well chosen $k \leq n$).

On an input (p_0, s_0) the automaton $\mathcal{B}_{\mathrm{rk}}$ computes *on-the-fly* the image of (p_0, s_0) by ν and simulates $\mathcal{B}_{\mathrm{lf}}$ on it. In order to compute $\nu((p_0, s_0))$, $\mathcal{B}_{\mathrm{rk}}$ needs to retrieve, when reading a stack symbol with an n-link, the states that are winning for the stack obtained by collapsing the n-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 $\mathcal{B}_{\mathrm{lf}}$, hence keeps track of this information) and hence the automaton can access it by definition of the model of automata. Indeed, the information (*i.e.* the states winning when doing a collapse) is correct before reading the first stack symbol coming with an n-link, and by induction on the number of n-links, if it is correct after processing the k first symbols with an n-link, on reading the $(k+1)$-th symbol with an 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.

We do not formally describe $\mathcal{B}_{\mathrm{rk}}$ as it is rather straightforward but we note that the size of $\mathcal{B}_{\mathrm{rk}}$ is linear in the size of $\mathcal{B}_{\mathrm{lf}}$. □

5.5 Strategies

In order to complete the proof of Theorem 5.1 it remains to establish the following proposition.

Proposition 5.6. *If there is an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ synchronised with $\mathcal{A}_{\mathrm{lf}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ from $\nu((q_{0,\mathrm{rk}}, \perp_n))$, then one can effectively construct an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$ synchronised with $\mathcal{A}_{\mathrm{rk}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ from the initial configuration $(q_{0,\mathrm{rk}}, \perp_n)$.*

Proof. The result follows from a carefully analysis of how we defined φ_{rk} from φ_{lf} in the proof of Lemma 5.3. As we now only focus on the initial configuration $(q_{0,\mathrm{rk}}, \perp_n)$ we will not have to deal with the special case of doing a *collapse* following (possibly a copy of) an n-link originally in the initial configuration. Also note that $\nu((q_{0,\mathrm{rk}}, \perp_n)) = (q_{0,\mathrm{rk}}, \perp_n)$.

Recall that φ_{rk} uses as a memory a partial play λ_{lf} in $\mathbb{G}_{\mathrm{lf}}$ and considers the value of $\varphi_{\mathrm{lf}}(\lambda_{\mathrm{lf}})$ to determine the next move to play. Now assume that φ_{lf} is realised by an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ synchronised with $\mathcal{A}_{\mathrm{lf}}$. Hence, instead of storing λ_{lf} it suffices to store the configuration $\mathcal{S}_{\mathrm{lf}}$ is in after reading λ_{lf}.

One can also notice that the stack s_{rk} in the last configuration of some partial play λ_{rk} and the stack s_{lf} in the last configuration of the associated λ_{lf} have the same shapes *provided* one replaces in s_{lf} every 1-link from a symbol in $\Gamma_{\mathrm{rk}} \times (2^{Q_{\mathrm{rk}}})^{d+1}$ by an n-link. Recall that these 1-links are never used to perform a *collapse*: hence replacing those 1-links by n-links does not change the issue of the game, and if one does a similar transformation on $\mathcal{S}_{\mathrm{lf}}$ it still realises a winning strategy, and it is synchronised with the transformed version of λ_{lf}.

Now, it follows from the way one defined φ_{rk} (both the choice of the move and the memory update) that one can design an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$ synchronised with $\mathcal{A}_{\mathrm{rk}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ from the initial configuration $(q_{0,\mathrm{rk}}, \perp_n)$. In all cases but one $\mathcal{S}_{\mathrm{rk}}$ simulates $\mathcal{S}_{\mathrm{lf}}$. The only problematic case is when the move to play is some $(r, \mathit{collapse})$ involving an n-link. Indeed, one needs to backtrack in λ_{lf} (namely retrieve the configuration of $\mathcal{S}_{\mathrm{lf}}$ right after the link ancestor) and extend it by doing (r^ℓ, id) (where ℓ is the link rank) and then (r, pop_n); one needs to retrieve the configuration of $\mathcal{S}_{\mathrm{lf}}$ right after this. If one performs a *collapse* in $\mathcal{S}_{\mathrm{rk}}$, one directly retrieves the stack content, but the control state of $\mathcal{S}_{\mathrm{lf}}$ is still missing. However, one can modify $\mathcal{S}_{\mathrm{lf}}$ so that after the simulation of the creation of an n-link, *i.e.* after a symbol of the form $(\beta, \overrightarrow{R})$ is pushed, it stores in its top_1-element the control state it will be in after doing the transitions $(r^\ell, id)(r, pop_n)$, for each $0 \leq \ell \leq d$ and each $r \in R_\ell$ (this can easily be computed). As this information is then propagated when copying the symbol/link, it is available in the top_1-element before doing a *collapse* involving an n-link, hence $\mathcal{S}_{\text{rk}}$ can also correctly retrieve the control state of $\mathcal{S}_{\text{lf}}$.

From this (somehow informal) description of $\mathcal{S}_{\text{rk}}$ the reader should be convinced that $\mathcal{S}_{\text{rk}}$ correctly simulates $\mathcal{S}_{\text{lf}}$ on λ_{lf} and hence, realises a winning strategy in $\mathbb{G}_{\text{rk}}$. The fact that $\mathcal{S}_{\text{rk}}$ is synchronised with $\mathcal{A}_{\text{rk}}$ follows from the fact that it is synchronised with the variant of $\mathcal{S}_{\text{lf}}$ that itself is synchronised with the variant of λ_{lf} which is synchronised with λ_{rk}. □

5.6 Optimising the Construction

The set Q_{lf} has size $O(|Q_{\text{rk}}|(|\Gamma_{\text{rk}}| + |C| + 3))$, which is not very satisfactory for complexity reasons. Actually, one would prefer a variant of the construction where $|\Gamma_{\text{rk}}|$ does not appear in the blowup concerning states. This factor actually comes from states $\{q^\gamma \mid q \in Q_{\text{rk}},\ \gamma \in \Gamma_{\text{rk}}\}$, and one can easily get rid of them by doing the following modification on $\mathcal{A}_{\text{lf}}$. When simulating a $push_1^{\beta,n}$, instead of going to q^β, one stores the information on β (thanks to a rew_1 operation) in the top_1 element of the stack (hence, the stack alphabet increases by a linear factor in $|\Gamma_{\text{rk}}|$) and goes to a special state $q^!$. State $q^!$ is controlled by Éloïse and the transition function is the same as from q^β where β is the symbol stored on the top_1-element of the stack.

It is straightforward that this modification does not change the validity of Proposition 5.5 nor Proposition 5.6.

5.7 Complexity

If we summarise, the overall blowup in the transformation from $\mathbb{G}_{\text{rk}}$ to $\mathbb{G}_{\text{lf}}$ given by Theorem 5.1 is as follows.

Proposition 5.7. *Let $\mathcal{A}_{\text{rk}}$ and $\mathcal{A}_{\text{lf}}$ be as in Theorem 5.1. Then the set of states of $\mathcal{A}_{\text{lf}}$ has size $O(|Q_{\text{rk}}|(|C| + 3))$ and the stack alphabet of $\mathcal{A}_{\text{lf}}$ has size $O(|\Gamma_{\text{rk}}|^2 \cdot 2^{|Q_{\text{rk}}||C|})$. Finally, the set of colours used in $\mathbb{G}_{\text{rk}}$ and $\mathbb{G}_{\text{lf}}$ are the same.*

Proof. By construction together with the optimisation discussed in Section 5.6. □

6 REDUCING THE ORDER

In the previous section, given a game played on a *rank-aware* n-CPDA, we have constructed another game played on an n-CPDA that does not create n-links. The winning region (*resp.* a winning strategy realised by an n-CPDA transducer) in the original game can then be recovered from the winning region (*resp.* a winning strategy realised by n-CPDA transducer) in the latter game.

In this section, we prove a result of a similar flavour. Namely, starting from a game played on an n-CPDA that does not create n-links, we construct a game played on an $(n-1)$-CPDA, and we show that the winning region (*resp.* a winning strategy realised by an n-CPDA transducer) in the original game can be recovered from the winning region (*resp.* a winning strategy realised by an $(n-1)$-CPDA transducer) in the latter game.

We situate the techniques developed here in a general and abstract framework of (order-1) pushdown automata whose stack alphabet is a *possibly infinite* set: *abstract pushdown automata.* We start by introducing this concept and show how n-CPDA that do not create n-links fit into it. Then, we introduce a model of automata, *automata with oracles*, that accept configurations of abstract pushdown automata and we relate this model with automata accepting configurations of n-CPDA as defined in Section 2.6. Then, we introduce the notion of *conditional games* and show that it is the notion that captures the winning region in the original game. Finally, we show how such games can be solved by reduction to an $(n-1)$-CPDA parity game, and from the proof we also get the expected result on the regularity of the winning region and on the existence of a winning strategy realised by a CPDA transducer.

6.1 Abstract Pushdown Automata

An ***abstract pushdown automaton*** is a tuple $\mathcal{A} = (A, Q, \Delta, q_0)$ where A is a (possibly infinite) set called an ***abstract pushdown alphabet*** and containing a bottom-of-stack symbol denoted $\perp \in A$, Q is a finite set of states, $q_0 \in Q$ is an initial state and

$$\Delta : Q \times A \to 2^{Q \times A^{\leq 2}}$$

is the transition relation (here $A^{\leq 2} = \{\varepsilon\} \cup A \cup A \cdot A$ are the words over A of length at most 2). We additionally require that for all $a \neq \perp$, $\Delta(q, a) \subseteq Q \times (A \setminus \{\perp\})^{\leq 2}$ and that $\Delta(q, \perp) \subseteq Q \times (\{\perp\} \cup \{\perp b \mid b \neq \perp\})$, *i.e.* the bottom-of-stack symbol can only occur at the bottom of the stack, and is never popped nor rewritten.

An ***abstract pushdown content*** is a word in $St = \perp(A \setminus \{\perp\})^*$. A configuration of $\mathcal{A}$ is a pair (q, s) with $q \in Q$ and $s \in St$.

Remark 6.1. 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.

A abstract pushdown automaton $\mathcal{A}$ induces a possibly infinite graph $G = (V, E)$, called an ***abstract pushdown graph,*** whose vertices are the configurations of $\mathcal{A}$ and edges are defined by the transition relation Δ, *i.e.* , from a vertex $(q, s \cdot a)$ one has an edge to $(q', s \cdot u)$ whenever $(q', u) \in \Delta(q, a)$.

Example 6.2. An order-1 pushdown automaton is an abstract pushdown automaton whose stack alphabet is finite.

Example 6.3. Order-n CPDA that do not create n-links are special cases of abstract pushdown automata. Indeed, let $n > 1$ and consider such an order-n CPDA $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$. Let A be the set of all order-$(n-1)$ stacks over Γ, and for every $p \in Q$ and $a \in A$ with $\gamma = top_1(a)$, we define $\Delta'(p, a)$ by

- $(q, \varepsilon) \in \Delta'(p, a)$ iff $(q, rew_1^{\alpha}; pop_n) \in \Delta(q, \gamma)$;
- $(q, a' \cdot a') \in \Delta'(p, a)$ with $a' = rew_1^{\alpha}(a)$ iff $(q, rew_1^{\alpha}; push_n) \in \Delta(q, \gamma)$;
- $(q, a') \in \Delta'(p, a)$ with $a' = op(rew_1^{\alpha}(a))$ iff $(q, rew_1^{\alpha}; op) \in \Delta(q, \gamma)$ and $op \notin \{pop_n, push_n\}$.

It follows from the definitions that $\mathcal{A}$ and the abstract pushdown automaton (A, Q, Δ', q_0) have isomorphic transition graphs.

Consider now a partition $Q_E \cup Q_A$ of Q between Éloïse and Abelard. It induces a natural partition $V_E \cup V_A$ of V by setting $V_E = Q_E \times St$ and $V_A = Q_A \times St$. The resulting arena $\mathcal{G}_{abs} = (V_E, V_A, E)$ is called an ***abstract pushdown arena***. 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 $\rho((q, t)) = \rho(q)$. Finally, an ***abstract pushdown parity game*** is a parity game played on such an abstract pushdown arena where the colouring function is defined as above.

6.2 Automata with Oracles

An ***automaton with oracles*** is a tuple $\mathcal{B} = (P, Q, A, \delta, p_0, O_1 \cdots O_k, Acc)$ where P is a finite set of control states, Q is a set of input states, A is a (possibly infinite) input alphabet, $p_0 \in P$ is the initial state, O_i are subsets of A (called ***oracles***) and $\delta : P \times \{0, 1\}^k \to S$ is the transition function. Finally Acc is a function from P 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 A and whose set of control states is Q.

Let $\mathcal{B} = (P, Q, A, \delta, p_0, O_1 \cdots O_k, Acc)$ be such an automaton. With every $a \in A$ we associate a Boolean vector $\pi(a) = (b_1, \cdots b_k)$ where

$$b_i = \begin{cases} 1 & \text{if } a \in O_i \\ 0 & \text{otherwise.} \end{cases}$$

The automaton reads a configuration $C = (q, a_1 a_2 \cdots a_\ell)$ from left to right. A ***run*** over C is the sequence $r_0, \cdots, r_{\ell+1}$ such that $r_0 = p_0$ and $r_{i+1} = \delta(r_i, \pi(a_i))$ for every $i = 0, \cdots, \ell$. Finally the run is ***accepting*** if and only if $q \in Acc(r_{\ell+1})$.

Remark 6.4. When the input alphabet is finite, it is easily seen that automata with oracles have the same expressive power as usual deterministic finite automata.

We are going to use automata with oracles to accept sets of configurations of n-CPDA that do not have n-links. As seen in Example 6.3 for an order-n CPDA that does not have n-links, we take A to be the set of all order-$(n-1)$ stacks. The sets of configurations of an order-n CPDA without n-links accepted by automata that use as oracles regular sets of order-$(n-1)$ stacks are easily seen to be regular.

Proposition 6.5. *Let $\mathcal{A}$ be an order-n CPDA $\mathcal{A}$ that never creates n-links. Let $\mathcal{B}$ be an automaton with oracles $O_1, \ldots, O_k$ and assume that each O_i is a regular set of $(n-1)$-stacks (and denote by C_i an associated automaton). Let C be the set of configurations of $\mathcal{A}$ accepted by $\mathcal{B}$. Then C is regular and we can construct an automaton C (now working on order-n stacks without n-links) of size $O(n|\mathcal{B}||C_1|\cdots|C_k|)$ accepting it.*

Proof. It suffices to mimic the behaviour of $\mathcal{B}$ and to run in parallel the C_is to compute the value of the oracles. Hence, the automaton C is obtained by taking a synchronised product of $\mathcal{B}$ together with the automata $C_1, \cdots, C_k$. An extra component, coding a counter taking its values in $\{0, 1, \ldots, n\}$, is needed to keep track of the bracketing depth (initially the counter equals 0; on reading an opening bracket [the counter is incremented, on reading a closing bracket] it is decremented). When the counter is equal to 0 or 1 one simulates $\mathcal{B}$. When the counter goes to 2 (and as long as it differs from 1) one simulates in parallel the C_is. When the counter returns to 1 the components corresponding to the C_is give the value of the oracles on the last $(n-1)$-stack (*i.e.* $b_i = 1$ if and only if the control state of the C_is component is final). Hence the $\mathcal{B}$ component can be updated. Then the control states of the C_is are put back to the initial state and the next $(n-1)$-stack is processed. Finally, when the counter is again equal to 0 (*i.e.* the last closing bracket has been read), the control state q of the input configuration is read and C goes to a final state if and only if the current state p in the $\mathcal{B}$ component is such that $q \in Acc(p)$. □

6.3 Conditional Games and Winning Regions of Abstract Pushdown Parity Games

We fix an abstract pushdown automaton $\mathcal{A} = (A, Q, \Delta, q_0)$ together with a partition $Q_E \cup Q_A$ of Q and a colouring function ρ using a finite set of colours C. We denote respectively by $\mathcal{G}_{\text{abs}} = (V, E)$ and $\mathbb{G}_{\text{abs}}$ the associated abstract pushdown arena and abstract pushdown parity game.

We show in Lemma 6.6 below how to define an automaton with oracles that accepts Éloïse's winning region in the game $\mathbb{G}_{\text{abs}}$. The oracles of this automaton are defined using the concept of conditional game. For every subset $R \subseteq Q$ we define the ***conditional game induced by R over $\mathcal{G}_{\text{abs}}$***, denoted $\mathbb{G}_{\text{abs}}(R)$, as the game played over $\mathcal{G}_{\text{abs}}$ where a play λ is winning for Éloïse iff one of the following happens:

- In λ no configuration with an empty stack, *i.e.* of the form $(q, \perp)$, 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 set of winning plays $\Omega(R)$ in $\mathbb{G}_{\text{abs}}(R)$ is defined as follows:

$$\Omega(R) = [\Omega_p \setminus V^*(Q \times \{\perp\})V^\omega] \ \cup \ V^*(R \times \{\perp\})V^\omega$$

For any state q, any stack letter $a \neq \perp$, and any subset $R \subseteq Q$ it follows from Martin's Determinacy theorem [29] that either Éloïse or Abelard has a winning strategy from $(q, \perp a)$ in $\mathbb{G}_{\text{abs}}(R)$. We denote by $\mathcal{R}(q, a)$ the set of subsets R for which Éloïse wins in $\mathbb{G}_{\text{abs}}(R)$ from $(q, \perp a)$:

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

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

Lemma 6.6. *Let $\mathbb{G}_{\text{abs}}$ be an abstract pushdown parity game induced by an abstract pushdown automaton $\mathcal{A} = (A, Q, \Delta, q_0)$. Then the set of winning positions in $\mathbb{G}_{\text{abs}}$ for Éloïse is accepted by an automaton with oracles $\mathcal{A} = (P, Q, A, \delta, p_0, O_1 \cdots O_k, Acc)$ such that*

- *$P = 2^Q$*
- *$p_0 = \emptyset$*
- *There is an oracle $O_{q,R}$ for every $q \in Q$ and $R \subseteq Q$, and $a \in O_{q,R}$ iff $R \in \mathcal{R}(q, a)$ and $a \neq \perp$*
- *There is an oracle $O_\perp$ and $a \in O_\perp$ iff $a = \perp$*
- *Using the oracles, δ is designed so that:*
  - *From state $\emptyset$ on reading $\perp$, $\mathcal{A}$ goes to $\{q \mid (q, \perp)$ is winning for Éloïse in $\mathbb{G}_{\text{abs}}\}$*
  - *From state R on reading a, $\mathcal{A}$ goes to $\{q \mid R \in \mathcal{R}(q, a)\}$*
- *Acc is the identity function*

The proof of Lemma 6.6 is a direct consequence of the following proposition.

Proposition 6.7. *Let $s \in (A \setminus \{\perp\})^*$, $q \in Q$ and $a \in A \setminus \{\perp\}$. Then Éloïse has a winning strategy in $\mathbb{G}_{\text{abs}}$ from $(q, \perp sa)$ if and only if there exists some $R \in \mathcal{R}(q, a)$ such that $(r, \perp s)$ is winning for Éloïse in $\mathbb{G}_{\text{abs}}$ for every $r \in R$.*

Proof. Assume Éloïse has a winning strategy from $(q, \perp sa)$ in $\mathbb{G}_{\text{abs}}$ and call it φ. Consider the set $\mathcal{L}$ of all plays in $\mathbb{G}_{\text{abs}}$ that start from $(q, \perp sa)$ and where Éloïse respects φ. Define R to be the (possibly empty) set that consists of all $r \in Q$ such that there is a play in $\mathcal{L}$ of the form $v_0 \cdots v_k (r, \perp s) v_{k+1} \cdots$ where each v_i for $0 \leq i \leq k$ is of the form $(p_i, \perp st_i)$ for some $t_i \neq \varepsilon$. In other words, R consists of all states that can be reached on popping (possibly a rewriting of) a for the first time in a play where Éloïse respects φ. Define a (partial) function $\tau : V \to V$ by letting $\tau(p, \perp st) = (p, \perp t)$ for every $p \in Q$. Define a function $\tau^{-1} : V \to V$ by letting $\tau^{-1}(p, \perp t) = (p, \perp st)$ for all $t \in A^*$. We extend τ^{-1} as a morphism over V^*.

It is easily shown that $R \in \mathcal{R}(q, a)$. Indeed a winning strategy for Éloïse in $\mathbb{G}_{\text{abs}}(R)$ is defined as follows:

- if some empty stack configuration has already been visited, play any legal move,
- otherwise go to $\tau(\varphi(\tau^{-1}(\lambda)))$, where λ is the partial play seen so far.

By definition of $\mathcal{L}$ and R, it easily follows that the previous strategy is winning for Éloïse in $\mathbb{G}_{\text{abs}}(R)$, and therefore $R \in \mathcal{R}(q, a)$.

Finally, for every $r \in R$ there is, by definition of $\mathcal{L}$, a partial play λ_r that starts from $(q, \perp sa)$, where Éloïse respects φ and that ends in $(r, \perp s)$. A winning strategy for Éloïse in $\mathbb{G}_{\text{abs}}$ from $(r, \perp s)$ is given by $\psi(\lambda) = \varphi(\lambda'_r \cdot \lambda)$, where λ'_r denotes the partial play obtained from λ_r by removing its last vertex $(r, \perp s)$.

Conversely, let us assume that there is some $R \in \mathcal{R}(q,a)$ such that $(r, \bot s)$ is winning for Éloïse in $\mathbb{G}_{\text{abs}}$ for every $r \in R$. and denote by φ_r a winning strategy for Éloïse from $(r, \bot s)$ in $\mathbb{G}_{\text{abs}}$. Let φ_R be a winning strategy for Éloïse in $\mathbb{G}_{\text{abs}}(R)$ from $(q, \bot a)$. We define τ and τ^{-1} as in the direct implication and extend them as (partial) morphism over V^*. We now define a strategy φ for Éloïse in $\mathbb{G}_{\text{abs}}$ for plays starting from $(q, \bot sa)$. For any partial play λ,

- if λ does not contain a configuration of the form $(p, \bot s)$ then $\varphi(\lambda) = \tau^{-1}(\varphi_R(\tau(\lambda)))$;
- otherwise let $\lambda = \lambda' \cdot (r, \bot s) \cdot \lambda''$ where λ' does not contain any configuration of the form $(p, \bot s)$. From how φ is defined in the previous case, it is follows that $r \in R$. One finally sets $\varphi(\lambda) = \varphi_r((r, \bot s) \cdot \lambda'')$.

It is then easy to check that φ is a winning strategy for Éloïse in $\mathbb{G}_{\text{abs}}$ from $(q, \bot sa)$. □

6.4 Reducing the Conditional Game

The main purpose of this section is to build a new parity game $\widetilde{\mathbb{G}}$ whose winning region provides all the information needed to compute the sets $\mathcal{R}(q,a)$. Moreover, in the underlying arena the vertices no longer encode stacks.

To help readability, we will use standard letters, *e.g.* λ or φ, to denote objects (plays, strategies...) in $\mathbb{G}_{\text{abs}}$, and letters with tilde, *e.g.* $\widetilde{\lambda}$ or $\widetilde{\varphi}$, to denote objects in $\widetilde{\mathbb{G}}$.

For an infinite play $\lambda = v_0 v_1 \cdots$ in $\mathbb{G}_{\text{abs}}$, let $Steps_\lambda$ be the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. More formally, $Steps_\lambda = \{i \in \mathbb{N} \mid \forall j \geq i\ sh(v_j) \geq sh(v_i)\}$, where $sh((q, \bot a_1 \cdots a_n)) = n+1$ is the stack height. Note that $Steps_\lambda$ is always infinite and hence induces a decomposition of the play λ into infinitely many finite pieces.

In the decomposition induced by $Steps_\lambda$, a factor $v_i \cdots v_j$ is called a ***bump*** if $sh(v_j) = sh(v_i)$, called a ***Stair*** otherwise (that is, if $sh(v_j) = sh(v_i)+1$ and $j = i+1$).

For any play λ with $Steps_\lambda = \{n_0 < n_1 < \cdots\}$, we can define the sequence $(mcol_i^\lambda)_{i \geq 0} \in \mathbb{N}^\mathbb{N}$ by letting $mcol_i^\lambda = \min\{\rho(v_k) \mid n_i \leq k \leq n_{i+1}\}$. Obviously, this sequence fully characterises the parity condition.

Proposition 6.8. *For every play λ, one has $\lambda \in \Omega_\rho$ iff* $\liminf((mcol_i^\lambda)_{i\geq 0})$ *is even.*

In the sequel, we build a new parity game $\widetilde{\mathbb{G}}$ over a new arena $\widetilde{\mathcal{G}} = (\widetilde{V}, \widetilde{E})$. This game *simulates* the abstract pushdown game, in the sense that the sequence of visited colours during a *correct* simulation of a play λ in $\mathbb{G}_{\text{abs}}$ is exactly the sequence $(mcol_i^\lambda)_{i\geq 0}$. Moreover, a play in which a player does not correctly simulate the abstract pushdown game is losing for that player. We will then show how the winning region in $\widetilde{\mathbb{G}}$ permits to compute the sets $\{a \in A \mid R \in \mathcal{R}(q,a)\}$.

Before providing a description of the arena $\widetilde{\mathcal{G}}$, let us consider the following informal description of this simulation game. We aim at simulating a play in the abstract pushdown game from its initial configuration $(q_0, \bot)$. In $\widetilde{\mathcal{G}}$ we keep track of only the control state and the top stack symbol of the simulated configuration.

The interesting case is when the simulated play is in a configuration with control state p and top stack symbol a, and the player owning p wants to perform transition $(q, a'b)$, *i.e.* go to state q, rewrite a into a' and push b on top of it. For every strategy of Éloïse, there is a certain set of possible (finite) prolongations of the play (consistent with her strategy) that will end with popping b (or actually a symbol into which b was rewritten in the meantime) from the stack. We require Éloïse to declare a vector $\overrightarrow{R} = (R_0, \ldots, R_d)$ of $(d+1)$ subsets of Q, where R_i is the set of all states the game can be in after popping (possibly a rewriting of) b along those plays where in addition the smallest visited colour whilst (possibly a rewriting of) b was on the stack is i.

Abelard has two choices. He can continue the game by pushing b onto the stack and updating the state; we call this a ***pursue move***. Otherwise, he can select a set R_i and pick a state $r \in R_i$, and continue the simulation from that state r; we call this a ***jump move***. If he does a pursue move, then he remembers the vector $\overrightarrow{R}$ claimed by Éloïse; if later on, a transition of the form (r, ε) is simulated, the play goes into a sink state (either q_{t} or $q_{\dagger}$) that is winning for Éloïse if and only if the resulting state is in R_θ where θ is the smallest colour seen in the current level (this information will be encoded in the control state, reseted after each pursue move and updated after each jump move). If Abelard does a jump move to a state r in R_i, the currently stored value for θ is updated to $\min(\theta, i, \rho(r))$, which is the smallest colour seen since the current stack level was reached.

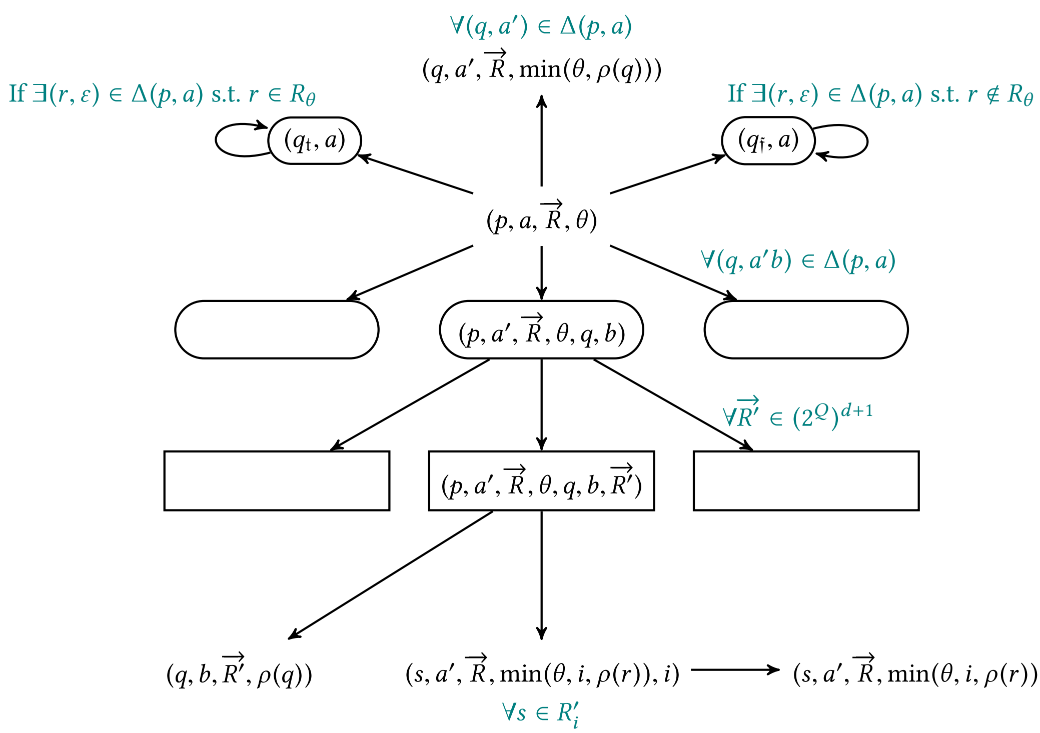

Fig. 4. Local structure of $\widetilde{\mathcal{G}}$.

Let us now precisely describe the arena $\widetilde{\mathcal{G}}$. We refer the reader to Figure 4.

- The main vertices of $\widetilde{\mathcal{G}}$ are those of the form $(p, a, \overrightarrow{R}, \theta)$, where $p \in Q$, $a \in A$, $\overrightarrow{R} = (R_0, \ldots, R_d) \in (2^Q)^{d+1}$ and $\theta \in \{0, \ldots, d\}$. A vertex $(p, a, \overrightarrow{R}, \theta)$ is reached when simulating a partial play λ in $\mathbb{G}_{\text{abs}}$ such that:
  - The last vertex in λ is (p, sa) for some $s \in A^*$.
  - Éloïse claims that she has a strategy to continue λ in such a way that if a (or a rewriting of it) is eventually popped, the control state reached after popping belongs to R_i, where i is the smallest colour visited since the stack height was at least $|sa|$.
  - The colour θ is the smallest one since the current stack level was reached from a lower stack level.

  A vertex $(p, a, \overrightarrow{R}, \theta)$ is controlled by Éloïse if and only if $p \in Q_{\mathrm{E}}$.

- The vertices (q_{t}, a) and (q_{f}, a) are here to ensure that the vectors $\overrightarrow{R}$ encoded in the main vertices are correct. Both are sink vertices and are controlled by Éloïse. Vertex (q_{t}, a) gets colour 0 and vertex (q_{f}, a) gets colour 1. As these vertices are sinks, a play reaching (q_{t}, a) is won by Éloïse whereas a play reaching (q_{f}, a) is won by Abelard.
  There is a transition from some vertex $(p, a, \overrightarrow{R}, \theta)$ to (q_{t}, a), if and only if there exists a transition rule $(r, \varepsilon) \in \Delta(p, a)$, such that $r \in R_\theta$ (this means that $\overrightarrow{R}$ is correct with respect to this transition rule). Dually, there is a transition from a vertex $(p, a, \overrightarrow{R}, \theta)$ to (q_{f}, a) if and only if there exists a transition rule $(r, \varepsilon) \in \Delta(p, a)$ such that $r \notin R_\theta$ (this means that $\overrightarrow{R}$ is not correct with respect to this transition rule).
- To simulate a transition rule $(q, a') \in \Delta(p, a)$, the player that controls $(p, a, \overrightarrow{R}, \theta)$ moves to $(q, a', \overrightarrow{R}, \min(\theta, \rho(q)))$. Note that the last component has to be updated as the smallest colour seen since the current stack level was reached is now $\min(\theta, \rho(q))$.
- To simulate a transition rule $(q, a'b) \in \Delta(p, a)$, the player that controls $(p, a, \overrightarrow{R}, \theta)$ moves to $(p, a', \overrightarrow{R}, \theta, q, b)$. This vertex is controlled by Éloïse who has to give a vector $\overrightarrow{R'} = (R'_0, \ldots, R'_d) \in (2^Q)^{d+1}$ that describes the control states that can be reached if b (or a symbol that rewrites it later) is eventually popped. To describe this vector, she goes to the corresponding vertex $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'})$.
  Any vertex $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'})$ is controlled by Abelard who chooses either to simulate a bump or a stair. In the first case, he additionally has to pick the minimal colour of the bump. To simulate a bump with minimal colour i, he goes to a vertex $(r', a', \overrightarrow{R}, \min(\theta, i, \rho(s)))$, for some $r' \in R'_i$, through an intermediate vertex $(r', a', \overrightarrow{R}, \min(\theta, i, \rho(s)), i)$ coloured by i.
  To simulate a stair, Abelard goes to the vertex $(q, b, \overrightarrow{R'}, \rho(q))$.
  The last component of the vertex (that stores the smallest colour seen since the currently simulated stack level was reached) has to be updated in all those cases. After simulating a bump of minimal colour i, the minimal colour is $\min(\theta, i, \rho(r'))$. After simulating a stair, this colour has to be initialised (since a new stack level is simulated). Its value, is therefore $\rho(q)$, which is the unique colour since the (new) stack level was reached.

The vertices of the form $(p, a, \overrightarrow{R}, \theta)$ get colour $\rho(p)$. Intermediate vertices of the form $(p, a', \overrightarrow{R}, \theta, q, b)$ or $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'})$ get colour d and hence, will be neutral with respect to the parity condition.

The following lemma relates the winning region in $\widetilde{\mathbb{G}}$ with $\mathbb{G}_{\mathrm{abs}}$ and the conditional games induced over $\mathcal{G}_{\mathrm{abs}}$.

Lemma 6.9. *For every $p_0, q \in Q$ and $a \in A$ the following holds.*

(1) *Configuration $(p_0, \perp)$ is winning for Éloïse in $\mathbb{G}_{\mathrm{abs}}$ if and only if $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*
(2) *For every $R \subseteq Q$, $R \in \mathcal{R}(q, a)$ if and only if $(q, a, (R, \ldots, R), \rho(q))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*

Remark 6.10. Note that the above lemma is proved in [33, Theorem 5.1] in the case of usual pushdown automata, *i.e.* when A is finite as remarked in Example 6.2. A careful analysis of that proof shows that it does not make use of the fact that A is finite and therefore the proof of Lemma 6.9 could be skipped. Nevertheless, we give it below for completeness and also because we need a careful analysis later when dealing with the regularity of the winning configuration and when constructing a $(n-1)$-transducer realising a winning strategy (in Theorem 6.15 below).

The rest of the section is devoted to the proof of Lemma 6.9. We mainly focus on the proof of the first item, the proof of the second one being a subpart of it. We start by introducing some useful concept and then prove both implications.

6.4.1 *Factorisation of plays in $\mathbb{G}_{\text{abs}}$ and in $\widetilde{\mathbb{G}}$.*

Recall that for an infinite play $\lambda = v_0 v_1 \cdots$ in $\mathbb{G}_{\text{abs}}$, $Steps_\lambda$ denotes the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. Recall that for any play λ with $Steps_\lambda = \{n_0 < n_1 < \cdots\}$, we define the sequence $(mcol_i^\lambda)_{i\geq 0} \in \mathbb{N}^\mathbb{N}$ by letting $mcol_i^\lambda = \min\{\rho(v_k) \mid n_i \leq k \leq n_{i+1}\}$.

Indeed, for any play λ with $Steps_\lambda = \{n_0 < n_1 < \cdots\}$, one can define the sequence $(\lambda_i)_{i\geq 0}$ by letting $\lambda_i = v_{n_i} \cdots v_{n_{i+1}}$. Note that each of the λ_i is either a bump or a stair. In the later we designate $(\lambda_i)_{i\geq 0}$ as the ***rounds factorisation*** of λ.

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

- The round is of the form $(p, a, \overrightarrow{R}, \theta)(q, a', \overrightarrow{R}, \theta)$ and corresponds therefore to the simulation of a transition (q, a'). We designate it as a ***trivial bump***.
- The round is of the form $(p, a, \overrightarrow{R}, \theta)(p, a', \overrightarrow{R}, \theta, q, b)(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'})(s, a', \overrightarrow{R}, \min(\theta, i, \rho(s)), i)(s, a', \overrightarrow{R}, \min(\theta, i, \rho(s)))$ and corresponds therefore to the simulation of a transition $(q, a'b)$ pushing b followed by a sequence of moves that ends by popping b (or a rewriting of it). Moreover, i is the smallest colour encountered whilst b (or other stack symbol obtained by successively rewriting it) was on the stack. We designate it as a ***(non-trivial) bump***.
- The round is of the form $(p, a, \overrightarrow{R}, \theta)(p, a', \overrightarrow{R}, \theta, q, b)(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'})(q, b, \overrightarrow{R'}, \rho(q))$ and corresponds therefore to the simulation of a transition $(q, a'b)$ pushing a symbol b leading to a new stack level below which the play will never go. We designate it as a ***stair***.

We define the ***colour*** of a round as the smallest colour of the vertices in the round.

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

$$Rounds_{\widetilde{\lambda}} = \{n \mid v_n = (p, a, \overrightarrow{R}, \theta),\ p \in Q,\ a \in A, \overrightarrow{R} \in (2^Q)^{d+1},\ 0 \leq \theta \leq d\}$$

The set $Rounds_{\widetilde{\lambda}}$ induces a natural factorisation of $\widetilde{\lambda}$ into rounds. Indeed, let $Rounds_{\widetilde{\lambda}} = \{n_0 < n_1 < n_2 < \cdots\}$, then for all $i \geq 0$, we let $\widetilde{\lambda}_i = v_{n_i} \cdots v_{n_{i+1}}$. We call the sequence $(\widetilde{\lambda}_i)_{i\geq 0}$ the ***round factorisation*** of $\widetilde{\lambda}$. For every $i \geq 0$, $\widetilde{\lambda}_i$ is a round and the first vertex in $\widetilde{\lambda}_{i+1}$ equals the last one in $\widetilde{\lambda}_i$. Moreover, $\widetilde{\lambda} = \widetilde{\lambda}_0 \odot \widetilde{\lambda}_1 \odot \widetilde{\lambda}_2 \odot \cdots$, where $\widetilde{\lambda}_i \odot \widetilde{\lambda}_{i+1}$ denotes the concatenation of $\widetilde{\lambda}_i$ with $\widetilde{\lambda}_{i+1}$ without its first vertex.

In order to prove both implications of Lemma 6.9, we build from a winning strategy for Éloïse in one game a winning strategy for her in the other game. The main argument to prove that the new strategy is winning is to prove a correspondence between the factorisations of plays in both games.

6.4.2 *Direct implication.*

Assume that the configuration $(p_0, \perp)$ is winning for Éloïse in $\mathbb{G}_{\text{abs}}$, and let φ be a corresponding winning strategy for her.

Using φ, we define a strategy $\widetilde{\varphi}$ for Éloïse in $\widetilde{\mathbb{G}}$ from $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$. The strategy $\widetilde{\varphi}$ maintains as a memory a partial play λ in $\mathbb{G}_{\text{abs}}$. At the beginning λ is initialised to the vertex

$(p_0, \perp)$. We first describe $\widetilde{\varphi}$, and then we explain how λ is updated. Both the strategy $\widetilde{\varphi}$ and the update of λ, are described for a round.

Choice of the move. Assume that the play is in some vertex $(p, a, \overrightarrow{R}, \theta)$ for $p \in Q_E$. The move given by $\widetilde{\varphi}$ depends on $\varphi(\lambda)$:

- If $\varphi(\lambda) = (r, \varepsilon)$, then Éloïse goes to (q_t, a) (Proposition 6.11 will prove that this move is always possible).
- If $\varphi(\lambda) = (q, a')$, then Éloïse goes to $(q, a', \overrightarrow{R}, \min(\theta, \rho(q)))$.
- If $\varphi(\lambda) = (q, a'b)$, then Éloïse goes to $(p, a', \overrightarrow{R}, \theta, q, b)$.

In this last case, or in the case where $p \in Q_A$ and Abelard goes to $(p, a', \overrightarrow{R}, \theta, q, b)$, we also have to explain how Éloïse behaves from $(p, a', \overrightarrow{R}, \theta, q, b)$. She has to provide a vector $\overrightarrow{R'} \in (2^Q)^{d+1}$ that describes which states can be reached if b (or its successors by top rewriting) is eventually popped, depending on the smallest visited colour in the meantime. In order to define $\overrightarrow{R'}$, Éloïse considers the set of all possible continuations of $\lambda \cdot (q, sa'b)$ (where (p, sa) 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', sa') is visited after $\lambda \cdot (q, sa'b)$, that is if the stack level of b is eventually left. If it is the case, she considers the first configuration (r', sa') appearing after $\lambda \cdot (q, sa'b)$ and the smallest colour i since b and (possibly) its successors by top-rewriting were on the stack. For every $i \in \{0, \ldots, d\}$, R'_i is exactly the set of states $r' \in Q$ such that the preceding case happens. More formally,

$$R'_i = \{r' \mid \exists\, \lambda \cdot (q, sa'b)v_0 \cdots v_k (r', sa') \cdots \text{ play in } \mathbb{G}_{\text{abs}} \text{ where Éloïse respects } \varphi \text{ and}$$
$$\text{s.t. } |v_j| \geq |sa'b|,\ \forall j = 0, \ldots, k \text{ and } \min(\{\rho(v_j) \mid j = 0, \ldots, k\} \cup \{\rho(q)\}) = i\}$$

Finally, we let $\overrightarrow{R'} = (R'_0, \ldots, R'_d)$ and Éloïse moves to $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'})$.

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

- The round is a trivial bump and therefore a (q, a') transition was simulated. Let (p, sa) be the last vertex in λ, then the updated memory is $\lambda \cdot (q, sa')$.
- The round is a bump, and therefore a bump of colour i (where i is the colour of the round) starting with some transition $(q, a'b)$ and ending in a state $r' \in R'_i$ was simulated. Let (p, sa) be the last vertex in λ. Then the memory becomes λ extended by $(q, sa'b)$ followed by a sequence of moves, where Éloïse respects φ, that ends by popping b and reaches (r', sa') whilst visiting i as smallest colour. By definition of R'_i such a sequence of moves always exists.
- The round is a stair and therefore we have simulated a $(q, a'b)$ transition. If (p, sa) denotes the last vertex in λ, then the updated memory is $\lambda \cdot (q, sa'b)$.

Therefore, with any partial play $\widetilde{\lambda}$ in $\widetilde{\mathbb{G}}$ in which Éloïse respects her strategy $\widetilde{\varphi}$, is associated a partial play λ in $\mathbb{G}_{\text{abs}}$. An immediate induction shows that Éloïse respects φ in λ. The same arguments work for an infinite play $\widetilde{\lambda}$, and the corresponding play λ is therefore infinite, starts from $(p_0, \perp)$ and Éloïse respects φ in that play. Therefore it is a winning play.

The following proposition is a direct consequence of how $\widetilde{\varphi}$ was defined.

Proposition 6.11. *Let $\widetilde{\lambda}$ be a partial play in $\widetilde{\mathbb{G}}$ that starts from $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$, ends in a vertex of the form $(p, a, \overrightarrow{R}, \theta)$, and where Éloïse respects $\widetilde{\varphi}$. Let λ be the partial play associated with $\widetilde{\lambda}$ built by the strategy $\widetilde{\varphi}$. Then the following holds:*

(1) *λ ends in a vertex of the form (p, sa) for some $s \in A^*$.*

(2) *θ is the smallest visited colour in λ since a (or a symbol that was later rewritten as a) has been pushed.*
(3) *Assume that λ is extended, that Éloïse keeps respecting φ and that the next move after (p, sa) is to some vertex (r, s). Then $r \in R_\theta$.*

Proposition 6.11 implies that the strategy $\widetilde{\varphi}$ is well defined when it provides a move to some (q_{t}, a). Moreover, one can deduce that, if Éloïse respects $\widetilde{\varphi}$, no vertex of the form $(q_{\dagger}, a)$ is reached.

For plays that never reach a sink vertex (q_{t}, a), using the definitions of $\widetilde{\mathcal{G}}$ and $\widetilde{\varphi}$, we easily deduce the following proposition.

Proposition 6.12. *Let $\widetilde{\lambda}$ be a play in $\widetilde{\mathbb{G}}$ that starts from $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$, and where Éloïse respects $\widetilde{\varphi}$. Assume that $\widetilde{\lambda}$ never visits q_{t}, let λ be the associated play built by the strategy $\widetilde{\varphi}$, and let $(\lambda_i)_{i\geq 0}$ be its rounds factorisation. Let $(\widetilde{\lambda}_i)_{i\geq 0}$ be the rounds factorisation of $\widetilde{\lambda}$. Then, for every $i \geq 0$ the following hold:*

(1) *$\widetilde{\lambda}_i$ is a bump if and only if λ_i is a bump*
(2) *$\widetilde{\lambda}_i$ has colour $mcol_i^{\lambda}$.*

Now consider a play $\widetilde{\lambda}$ in $\widetilde{\mathbb{G}}$ starting from $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$ where Éloïse respects $\widetilde{\varphi}$. Either $\widetilde{\lambda}$ loops in some (q_{t}, a) (hence, is won by Éloïse). Or, thanks to Proposition 6.12 the sequence of visited colours in $\widetilde{\lambda}$ is $(mcol_i^{\lambda})_{i\geq 0}$ for the corresponding play λ in $\mathbb{G}_{\mathrm{abs}}$. Hence, using Proposition 6.8 we conclude that $\widetilde{\lambda}$ is winning if and only if λ is winning; as λ is winning for Éloïse, it follows that $\widetilde{\lambda}$ is winning for her as well.

6.4.3 Converse implication.

First note that in order to prove the converse implication one could follow the same approach as for the direct implication by considering now the point of view of Abelard. Nevertheless the proof we give here starts from a winning strategy for Éloïse in $\widetilde{\mathbb{G}}$ and constructs a strategy for her in $\mathbb{G}_{\mathrm{abs}}$: this induces a more involved proof but has the advantage of leading to an effective construction of a winning strategy for Éloïse in $\mathbb{G}_{\mathrm{abs}}$ if one has an effective winning strategy for her in $\widetilde{\mathbb{G}}$.

Assume now that Éloïse has a winning strategy $\widetilde{\varphi}$ in $\widetilde{\mathbb{G}}$ from $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$. Using $\widetilde{\varphi}$, we build a strategy φ for Éloïse in $\mathbb{G}_{\mathrm{abs}}$ for plays starting from $(p_0, \perp)$.

The strategy φ maintains as a memory a partial play $\widetilde{\lambda}$ in $\widetilde{\mathbb{G}}$, that is an element in $\widetilde{V}^*$. At the beginning $\widetilde{\lambda}$ is initialised to $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$.

For any play λ where Éloïse respects φ the following will hold.

- $\widetilde{\lambda}$ is a play in $\widetilde{\mathbb{G}}$ that starts from $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$ and where Éloïse respects her winning strategy $\widetilde{\varphi}$.
- The last vertex of $\widetilde{\lambda}$ is some $(p, a, \overrightarrow{R}, \theta)$ if and only if the current configuration in λ is of the form (p, sa).
- If Éloïse keeps respecting φ, and if a (or a symbol that rewrites it later) is eventually popped the configuration reached will be of the form (r, s) for some $r \in R_i$, where i is the smallest visited colour since a (or some symbol that was later rewritten as a) was on the stack.

Note that initially the previous invariants trivially hold.

In order to describe φ, we assume that we are in some configuration (p, sa) and that the last vertex of $\widetilde{\lambda}$ is some $(p, a, \overrightarrow{R}, \theta)$. We first describe how Éloïse plays if $p \in Q_{\mathrm{E}}$, and then we explain how $\widetilde{\lambda}$ is updated.

Choice of the move. Assume that $p \in Q_{\mathrm{E}}$. Then the move given by φ depends on $\widetilde{\varphi}(\widetilde{\lambda})$.

- If $\widetilde{\varphi}(\widetilde{\lambda}) = (q, a', \overrightarrow{R}, \min(\theta, \rho(q)))$, Éloïse plays transition (q, a').
- If $\widetilde{\varphi}(\widetilde{\lambda}) = (p, a', \overrightarrow{R}, \theta, q, b)$, then Éloïse applies plays transition $(q, a'b)$.
- If $\widetilde{\varphi}(\widetilde{\lambda}) = (q_{\mathrm{t}}, a)$, Éloïse plays transition (r, ε) for some state $r \in R_\theta$. Lemma 6.13 will prove that such an r always exists.

Update of $\widetilde{\lambda}$. The memory $\widetilde{\lambda}$ is updated after each move (played by any of the two players). We have several cases depending on the last transition.

- If the last move was from (p, sa) to (q, sa') then the updated memory is $\widetilde{\lambda} \cdot (q, a', \overrightarrow{R}, \min(\theta, \rho(q)))$.
- If the last move was from (p, sa) to $(q, sa'b)$, let $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'}) = \widetilde{\varphi}(\widetilde{\lambda} \cdot (p, a', \overrightarrow{R}, \theta, q, b))$. Then the updated memory is $\widetilde{\lambda} \cdot (p, a', \overrightarrow{R}, \theta, q, b) \cdot (p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{R'}) \cdot (q, b, \overrightarrow{R'}, \rho(q))$.
- If the last move was from (p, sa) to (r, s) the update of $\widetilde{\lambda}$ is as follows. One backtracks in $\widetilde{\lambda}$ until one finds a configuration of the form $(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})$ that is not immediately followed by a vertex of the form $(s, a'', \overrightarrow{R}, \theta'', i)$. This configuration is therefore in the stair that simulates the pushing of a'' onto the stack (here if $a'' \neq a$ it simply means that a'' was later rewritten as a). Call $\widetilde{\lambda}'$ the prefix of $\widetilde{\lambda}$ ending in this configuration. The updated memory is $\widetilde{\lambda}' \cdot (r, a', \overrightarrow{R'}, \min(\theta', \theta, \rho(r)), \theta) \cdot (r, a', \overrightarrow{R'}, \min(\theta', \theta, \rho(r)))$. Formally, write $\widetilde{\lambda} = \widetilde{\lambda}_0 \odot \widetilde{\lambda}_1 \odot \cdots \odot \widetilde{\lambda}_k$ where $(\widetilde{\lambda}_i)_{0 \le i \le k}$ is the round factorisation of $\widetilde{\lambda}$. Let $h \le k$ be the largest integer such that $\widetilde{\lambda}_h$ is a stair and let $\widetilde{\lambda}_h = (p', a', \overrightarrow{R'}, \theta')(p', a', \overrightarrow{R'}, \theta', p'', a'')(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})(p'', a'', \overrightarrow{R}, \rho(p''))$. Define $\widetilde{\lambda}'_h = (p', a', \overrightarrow{R'}, \theta')(p', a', \overrightarrow{R'}, \theta', p'', a'')(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})(r, a', \overrightarrow{R'}, \min(\theta', \theta, \rho(r)), \theta) \cdot (r, a', \overrightarrow{R'}, \min(\theta', \theta, \rho(r)))$. Then the updated memory is $\widetilde{\lambda}_1 \odot \widetilde{\lambda}_2 \odot \cdots \odot \widetilde{\lambda}_{h-1} \odot \widetilde{\lambda}'_h$.

The following lemma gives the meaning of the information stored in $\widetilde{\lambda}$.

Lemma 6.13. *Let λ be a partial play in $\mathbb{G}_{\mathrm{abs}}$, where Éloïse respects φ, that starts from $(p_0, \perp)$ and ends in a configuration (p, sa). We have the following facts:*

1. *The last vertex of $\widetilde{\lambda}$ is of the form $(p, a, \overrightarrow{R}, \theta)$ with $\overrightarrow{R} \in (2^Q)^{d+1}$ and $0 \le \theta \le d$.*
2. *$\widetilde{\lambda}$ is a partial play in $\widetilde{\mathbb{G}}$ that starts from $(p_0, \perp, (\emptyset, \ldots, \emptyset), \rho(p_0))$, that ends with $(p, a, \overrightarrow{R}, \theta)$ and where Éloïse respects $\widetilde{\varphi}$.*
3. *θ is the smallest colour visited since a (or some symbol that was later rewritten as a) was pushed.*
4. *If λ is extended by some move that pops a, the configuration (r, s) that is reached is such that $r \in R_\theta$.*

Proof. We first note that the last point is a consequence of the second and third points. Indeed, assume that the next move after (p, sa) is to play a transition $(r, \varepsilon) \in \Delta(p, a)$. The second point implies that $(p, a, \overrightarrow{R}, \theta)$ is winning for Éloïse in $\widetilde{\mathbb{G}}$. If $p \in Q_{\mathrm{E}}$, by definition of φ, there is some edge from that vertex to (q_{t}, a), which means that $r \in R_\theta$ and allows us to conclude. If $p \in Q_{\mathrm{A}}$, note that there is no edge from $(p, a, \overrightarrow{R}, \theta)$ (winning position for Éloïse) to the losing vertex $(q_{\dagger}, a)$. Hence we conclude the same way.

Let us now prove the other points by induction on λ. Initially, they trivially hold. Now assume that the result is proved for some play λ, and let λ' be an extension of λ. We have two cases, depending on how λ' extends λ:

- λ' is obtained by applying a transition of the form (q, a') or $(q, a'b)$. The result is trivial in that case.
- λ' is obtained by applying a transition of the form (r, ε). Let (p, sa) be the last configuration in λ, and let $\overrightarrow{R}$ be the last vector component in the last vertex of $\widetilde{\lambda}$ when in configuration

(p, sa). By the induction hypothesis, it follows that $\lambda' = \lambda \cdot (r, s)$ with $r \in R_\theta$. Considering how $\widetilde{\lambda}$ is updated, and using the fourth point, we easily deduce that the new memory $\widetilde{\lambda}$ is as desired.

□

Actually, we easily deduce a more precise result.

Lemma 6.14. *Let λ be a partial play in $\mathbb{G}_{\mathrm{abs}}$ starting from $(p_0, \perp)$ and where Éloïse respects φ and let $(\lambda_i)_{i\geq 0}$ be its rounds factorisation. Let $(\widetilde{\lambda}_i)_{i=0,\dots,k}$ be the rounds factorisation of $\widetilde{\lambda}$. Then the following holds for every $i \geq 0$.*

- *$\widetilde{\lambda}_i$ is a bump if and only if λ_i is a bump.*
- *$\widetilde{\lambda}_i$ has colour $mcol_i^{\lambda}$.*

Both lemmas 6.13 and 6.14 are for partial plays. A version for infinite plays would allow us to conclude. Let λ be an infinite play in $\mathbb{G}_{\mathrm{abs}}$. We define an infinite version of $\widetilde{\lambda}$ by considering the limit of the $(\widetilde{\lambda}_i)_{i\geq 0}$ where $\widetilde{\lambda}_i$ is the memory after the i first moves in λ. See Footnote 4 on page 29 for a similar construction. It is easily seen that such a limit always exists, is infinite and corresponds to a play won by Éloïse in $\widetilde{\mathbb{G}}$. Moreover the results of Lemma 6.14 remain true.

Let λ be a play in $\mathbb{G}_{\mathrm{abs}}$ with initial vertex $(p_0, \perp)$, and where Éloïse respects φ, and let $\widetilde{\lambda}$ be the associated play in $\widetilde{\mathbb{G}}$. Therefore $\widetilde{\lambda}$ is won by Éloïse. Using Lemma 6.14 and Proposition 6.8, we conclude, as in the direct implication that λ is winning.

6.5 Main Result

Following Example 6.3 we see an n-CPDA that does not create n-links as an abstract pushdown automaton and we apply the construction of Section 6.4. We argue that the resulting game $\widetilde{\mathbb{G}}$ is associated with an $(n-1)$-CPDA, which leads the following result.

Theorem 6.15. *For any n-CPDA $\mathcal{A}_{\mathrm{lf}} = (\Gamma_{\mathrm{lf}}, Q_{\mathrm{lf}}, \Delta_{\mathrm{lf}}, q_{0,\mathrm{lf}})$ that* does not create n-links *and any associated parity game $\mathbb{G}_{\mathrm{lf}}$, one can construct an $(n-1)$-CPDA $\widetilde{\mathcal{A}} = (\widetilde{\Gamma}, \widetilde{Q}, \widetilde{\Delta}, \widetilde{q_0})$ and an associated parity game $\widetilde{\mathbb{G}}$ such that the following holds.*

- *$(q_{0,\mathrm{lf}}, \perp_n)$ is winning for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ if and only if $(\widetilde{q_0}, \perp_{n-1})$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*
- *If the set of winning configurations for Éloïse in $\widetilde{\mathbb{G}}$ is regular, then the set of winning configurations for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ is regular as well.*
- *If there is an $(n-1)$-CPDA transducer $\widetilde{\mathcal{S}}$ synchronised with $\widetilde{\mathcal{A}}$ realising a well-defined winning strategy for Éloïse in $\widetilde{\mathbb{G}}$ from $(\widetilde{q_0}, \perp_{n-1})$, then one can effectively construct an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ synchronised with $\mathcal{A}_{\mathrm{lf}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ from the initial configuration $(q_{0,\mathrm{lf}}, \perp_n)$.*

Proof. Following Example 6.3, $\mathcal{A}_{\mathrm{lf}}$ can be seen as an abstract pushdown automaton hence, we can apply the construction of Section 6.4. We claim that the resulting game $\widetilde{\mathbb{G}}$ is associated with an $(n-1)$-CPDA.

Indeed, one simply needs to consider how the graph $\widetilde{G}$ is defined and make the following observations concerning the local structure given in Figure 4 when $\mathbb{G}$ is played on the transition graph of an n-CPDA that does not create links.

(1) For every vertex of the form $(p, a, \overrightarrow{R}, \theta)$, (q_{t}, a),$(q_{\dagger}, a)$,$(p, a, \overrightarrow{R}, \theta, q, b)$, $(p, a, \overrightarrow{R}, \theta, q, b, \overrightarrow{R'})$ or $(s, a, \vec{R}, \theta', i)$, a and b are $(n-1)$-stacks.

(2) For every vertex of the form $(p, a, \overrightarrow{R}, \theta, q, b)$ or $(p, a, \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$, one has $a = b$.

This implies that any vertex in $\widetilde{G}$ can be seen as a pair formed by a state in a finite set and an $(n-1)$-stack. Then one concludes the proof by checking that the edge relation is the one of an $(n-1)$-CPDA.

Therefore, the first point follows from Lemma 6.9 and the second one follows by combining Lemma 6.6 with Proposition 6.5 and Lemma 6.9.

We now turn to the third point and therefore assume that there is an $(n-1)$-CPDA transducer $\widetilde{\mathcal{S}}$ synchronised with $\widetilde{\mathcal{A}}$ realising a well-defined winning strategy $\widetilde{\varphi}$ for Éloïse in $\widetilde{\mathbb{G}}$ from $(\widetilde{q_0}, \perp_{n-1})$. We argue that the strategy φ constructed in the proof of Lemma 6.9 can be realised, when $\mathbb{G}_{\text{abs}}$ is obtained from an n-CPDA $\mathcal{A}_{\text{lf}}$ that does not create n-links, by an n-CPDA transducer $\mathcal{S}_{\text{lf}}$ synchronised with $\mathcal{A}_{\text{lf}}$.

For this, let us first have a closer look at φ. The key ingredient in φ is the play $\widetilde{\lambda}$ in $\widetilde{\mathbb{G}}$, and the value of φ uniquely depends on $\widetilde{\varphi}(\widetilde{\lambda})$. In particular, if $\widetilde{\varphi}$ is realised by an $(n-1)$-CPDA transducer $\widetilde{\mathcal{S}}$, it suffices to know the configuration of $\widetilde{\mathcal{S}}$ after reading $\widetilde{\lambda}$ in order to define φ. We claim that it can be computed by an n-CPDA transducer $\mathcal{S}_{\text{lf}}$ (synchronised with $\mathcal{A}_{\text{lf}}$); the hard part being to establish that such a device can update correctly its memory.

Let $\widetilde{\lambda} = v_0 v_1 \cdots v_\ell$ and let $r_{\widetilde{\lambda}} = (p_0, s_0)(p_1, s_1) \cdots (p_\ell, s_\ell)$ be the run of $\widetilde{\mathcal{S}}$ associated with $\widetilde{\lambda}$, *i.e.* after having played $v_0 \cdots v_k$, $\widetilde{\mathcal{S}}$ is in configuration (p_k, s_k). Denote by $Last(r_{\widetilde{\lambda}})$ the last configuration of $r_{\widetilde{\lambda}}$, *i.e.* (p_ℓ, s_ℓ). To define φ, $Last(r_{\widetilde{\lambda}})$ suffices but of course, in order to update $Last(r_{\widetilde{\lambda}})$, we need to recall some more configurations from $r_{\widetilde{\lambda}}$. In the case where the last transition applies an order-k stack operation with $k < n$ (*i.e.* it is neither pop_n nor $push_n$), then the update is simple, as it consists in simulating one step of $\widetilde{\mathcal{S}}$. If the last stack operation is $push_n$ then the update of $\widetilde{\lambda}$ consists in adding three vertices and the corresponding update of $r_{\widetilde{\lambda}}$ is simple (as the only operation on the $(n-1)$-stack is to rewrite the top_1-element). If the last stack operation is pop_n one needs to backtrack in $\widetilde{\lambda}$ (hence in $r_{\widetilde{\lambda}}$): the backtrack is to some v_k with k maximal such that v_k is of the form $(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})$ and $v_{k+1} = (p'', a'', \overrightarrow{R}, \rho(p''))$. Once v_k has been found, the update is fairly simple for both $\widetilde{\lambda}$ and $r_{\widetilde{\lambda}}$ (one simply extends the remaining prefix of $\widetilde{\lambda}$ by two extra vertices whose stack content is unchanged compared with the one in v_k).

Define the following set of indices where $\widetilde{\lambda} = v_0 v_1 \cdots v_\ell$

$$Ext(\widetilde{\lambda}) = \{h \mid v_h \text{ is of the form } (p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R}) \text{ and } v_{h+1} = (p'', a'', \overrightarrow{R}, \rho(p''))\} \cup \{\ell\}$$

Note that after a partial play λ the cardinality of $Ext(\widetilde{\lambda})$ is equal to the height of the stack in the last configuration of λ.

For any partial play λ in $\mathbb{G}_{\text{lf}}$ define the following n-stack (note that it does not contain any n-link)

$$Mem(\lambda) = [s'_{k_1} s'_{k_2} \cdots s'_{k_h}]$$

where we let

- $Ext(\widetilde{\lambda}) = \{k_1 < \cdots < k_h\}$, $\widetilde{\lambda}$ being the memory associated with λ as in the proof of Lemma 6.9;
- s'_j is the $(n-1)$-stack obtained from s_j (recall that (p_j, s_j) denotes the j-th configuration of $r_{\widetilde{\lambda}}$) by appending p_j to its top_1-symbol (*i.e.* we work on an enriched stack alphabet).

Note that $Last(r_{\widetilde{\lambda}})$ is essentially $top_1(Mem(\lambda))$ as the only difference is that now the control state is stored in the stack. Moreover $Mem(\lambda)$ can easily be updated by an n-CPDA transducer: for the case of a transition involving an order-k stack operation with $k < n$ one simulates $\widetilde{\mathcal{S}}$ on $top_1(Mem(\lambda))$; for the case of a transition involving a $push_n$ one first simulates $\widetilde{\mathcal{S}}$ on $top_1(Mem(\lambda))$ (as one may do a rew_1 before $push_n$) and then makes a $push_n$ to duplicate the topmost $(n-1)$-stack in $Mem(\lambda)$; finally, for the case of a pop_n, one simply needs to do a pop_n in $Mem(\lambda)$ to backtrack and then update the control state. This is how we define $\mathcal{S}_{\mathrm{lf}}$[5].

The fact that $\mathcal{S}_{\mathrm{lf}}$ is synchronised with $\mathcal{A}_{\mathrm{lf}}$ comes from the definition of how $\mathcal{S}_{\mathrm{lf}}$ behaves when the transition in $\mathcal{A}_{\mathrm{lf}}$ involves a pop_n or a $push_n$, and for the other cases it follows from the initial assumption of $\widetilde{\mathcal{S}}$ being synchronised with $\widetilde{\mathcal{A}}$. □

Remark 6.16. When applying the general construction of Section 6.4 to an n-CPDA $\mathcal{A}_{\mathrm{lf}}$ that does not create links, we can safely enforce the following extra constraint on the vectors $\overrightarrow{R}$ and $\overrightarrow{S}$: they should be element in $(2^{Q_{\mathrm{lf}}^{pop_n}})^{d+1}$ where we let $Q_{\mathrm{lf}}^{pop_n}$ denote the set of control states of $\mathcal{A}_{\mathrm{lf}}$ from which a pop_n operation can be performed. Indeed, the various component of such vectors aims at representing set of states reachable by doing a pop_n. This is important later in the overall complexity for Theorem 3.1.

6.6 Complexity

If we summarise, the overall blowup in the transformation from $\mathbb{G}_{\mathrm{lf}}$ to $\widetilde{\mathbb{G}}$ given by Theorem 6.15 is as follows.

Proposition 6.17. *Let $\mathcal{A}_{\mathrm{lf}}$ and $\widetilde{\mathcal{A}}$ be as in Theorem 6.15. Then the set of states of $\widetilde{\mathcal{A}}$ has size $O(2^{2^{|C||Q_{\mathrm{lf}}|}})$ and the stack alphabet of $\widetilde{\mathcal{A}}$ has size $O(|\Gamma_{\mathrm{lf}}|)$. Finally, the set of colours used in $\mathbb{G}_{\mathrm{lf}}$ and $\widetilde{\mathbb{G}}$ are the same.*

Proof. By construction. □

7 PROOF OF THEOREM 3.1 AND COMPLEXITY

The proof of Theorem 3.1 consists in combining theorems 4.8, 5.1 and 6.15. Indeed, starting from an n-CPDA, we apply Theorem 4.8 to obtain a rank-aware n-CPDA, then Theorem 5.1 to remove the order-n links, and finally Theorem 6.15 to obtain an $(n-1)$-CPDA. By $(n-1)$ successive applications of these three results, we end-up with a 1-CPDA parity game. If we apply to this latter (pushdown) game the construction of Section 6.4 we end up with a game on a finite graph. Solving this game and following the chain of equivalences provided by theorems 4.8, 5.1 and 6.15 concludes the proof.

Concerning complexity, one step of successive application of the construction in theorems 4.8, 5.1 and 6.15 results in an $(n-1)$-CPDA with a state set of size $O(2^{2^{|Q|(|C|+3)^{n+5}}})$, a stack alphabet of size $O(|\Gamma|^2 \cdot 2^{|Q|(|C|+1)^{n+5}})$ and an unchanged number of colours. Indeed,

- by Proposition 4.10 one has $|Q_{\mathrm{rk}}| = O(|Q| \cdot (|C|+1)^{n+3})$ and $|\Gamma_{\mathrm{rk}}| = O(|\Gamma| \cdot (|C|+1)^{2n+5})$;
- by Proposition 5.7 one has $|Q_{\mathrm{lf}}| = O(|Q_{\mathrm{rk}}| \cdot (|C|+3)) = O(|Q| \cdot (|C|+3)^{n+4})$ and $|\Gamma_{\mathrm{lf}}| = O(|\Gamma_{\mathrm{rk}}|^2 \cdot 2^{|Q_{\mathrm{rk}}||C|}) = O(|\Gamma|^2 \cdot (|C|+1)^{4n+10} \cdot 2^{|Q|(|C|+1)^{n+4}}) = O(|\Gamma|^2 \cdot 2^{|Q|(|C|+1)^{n+5}})$;
- and finally, by Proposition 6.17, one has $|\widetilde{Q}| = O(2^{2^{|C||Q_{\mathrm{lf}}|}}) = O(2^{2^{|Q|(|C|+3)^{n+5}}})$ and $|\widetilde{\Gamma}| = O(|\Gamma_{\mathrm{lf}}|) = O(|\Gamma|^2 \cdot 2^{|Q|(|C|+1)^{n+5}})$.

If one lets, for a constant K, Exp_h^K be the function defined by $\mathrm{Exp}_0^K(x) = x$ for all x and $\mathrm{Exp}_{h+1}^K(x) = 2^{K\mathrm{Exp}_h^K(x)}$, we conclude that the 1-CPDA obtained after $(n-1)$ successive applications of the three reductions has

- a state set of size $O(\mathrm{Exp}_{n-1}^{2(|C|+3)^{n+5}}(|Q|))$ and
- a stack alphabet of size $O(|\Gamma|^{2(n-1)} \cdot \mathrm{Exp}_{n-1}^{(|C|+1)^{n+5}}(|Q|))$.

[5]Technically speaking, if we impose that a transition of $\mathcal{S}_{\mathrm{lf}}$ does a rew_1 (or *id*) followed by another stack operation, we may not be able to do the update of the stack after doing a pop_n. However, we can use the same trick as the one used to define $\mathcal{A}_{\mathrm{rk}}$, *i.e.* we postpone the rew_1 action to the next transition (see Remark 4.9).

Solving this latter game can be done by reducing it using the construction of Section 6.4 which leads to solve a parity game on a finite graph with $O(\mathrm{Exp}_n^{2(|C|+3)^{n+5}}(|Q|)\cdot(|\Gamma|^{2(n-1)}\cdot\mathrm{Exp}_{n-1}^{(|C|+1)^{n+5}}(|Q|))^2)$ vertices. Solving this game can be achieved in time $O(N^{|C|})$ where N denotes the number of vertices. Hence, the overall complexity of deciding the winner in an n-CPDA parity game is:

- n-times exponential in the number of states of the CPDA;
- n-times exponential in the number of colours;
- polynomial in the size of the stack alphabet of the CPDA.

Regarding lower bound, the problem is n-ExpTime-hard. In fact, hardness already holds when one considers reachability condition (*i.e.* does the play eventually visit a configuration with a final control state?) for games generated by higher-order pushdown automata (*i.e.* CPDA that never use *collapse*). A self-contained proof of this result was established by Cachat and Walukiewicz, but is fairly technical [12].

Here we sketch a much simpler proof of this result that relies on the following well-known result: checking emptiness of a nondeterministic order-n higher-order pushdown automaton is an $(n-1)$-ExpTime-complete problem [20] (here one uses higher-order pushdown automata as word acceptors)[6]. Trivially, this result is still true if we assume that the input alphabet is reduced to a single letter. Now consider an order-$(n+1)$ nondeterministic higher-order pushdown automaton $\mathcal{A}$ whose input alphabet is reduced to a single letter. The language accepted by $\mathcal{A}$ is non-empty if and only if there is a path from the initial configuration of $\mathcal{A}$ to a final configuration of $\mathcal{A}$ in the transition graph G of $\mathcal{A}$. Equivalently, the language accepted by $\mathcal{A}$ is non-empty if and only if Éloïse wins the reachability game $\mathbb{G}$ over G where she controls all vertices (and where the play starts from the initial configuration of $\mathcal{A}$ and where final vertices are those corresponding to final configurations of $\mathcal{A}$). Now, consider the reduction used to prove Theorem 3.1 and apply it to $\mathbb{G}$. As $\mathcal{A}$ does not use links, we only need to do the third step, which leads to an *equivalent* reachability game $\widetilde{\mathbb{G}}$ that is now played on the transition graph of an order-n higher order pushdown automaton. In the new arena, the main vertices are of the form $(p, s, \overrightarrow{R}, \theta)$: here s is an n-stack (without links), $\overrightarrow{R}$ is actually a pair (R_0, R_1) (we consider a reachability condition) and θ is either 0 or 1. The important fact is that R_0 and R_1 can be forced to be singletons: this follows from the fact that all vertices in $\mathbb{G}$ are controlled by Éloïse (and thus she can precisely force in which state the play goes if some pop_{n+1} is eventually done). Therefore, one concludes that the size of the arena associated with $\widetilde{\mathbb{G}}$ is polynomial in the size of $\mathcal{A}$. Hence, one has shown the following: checking emptiness for an order-$(n+1)$ nondeterministic higher-order pushdown automaton whose input alphabet is reduced to a single letter can be polynomially reduced to solve a reachability game over the transition graph of an order-n higher-order pushdown automaton. In conclusion, this latter problem is n-ExpTime-hard.

8 CONSEQUENCES

8.1 Marking The Winning Region

If one combines the fact that the winning region in a CPDA parity game is regular (Theorem 3.1) together with the fact that the model of CPDA can perform regular test (Theorem 2.8) one directly gets the following result.

[6] The following result is also proved in [20]: checking emptiness of an alternating order-n higher-order pushdown automaton is an n-EXPTIME complete problem. Nevertheless, note that this result does not directly imply hardness for games on higher-order pushdown graphs. Indeed, in general it is *more difficult* to check emptiness for an alternating device than to solve a reachability game on the corresponding class of graphs: for instance, solving a reachability game on a finite graph is in P while checking emptiness for an alternating automata on finite word (even if one considers a 1-letter alphabet) is PSPACE-complete; the problems are trivially equivalent only when considering infinite words on a single letter alphabet.

Corollary 8.1. *Let $\mathcal{A} = (\Gamma, Q, \delta, q_0)$ be an n-CPDA and let $\mathbb{G}$ be an n-CPDA parity game defined from $\mathcal{A}$. Then, one can build an order-n CPDA $\mathcal{A}'$ with a state-set Q', a subset $F \subseteq Q'$ and a mapping $\chi : Q' \to Q$ such that the following holds.*

(1) *Restricted to the reachable configurations from their respective initial configuration, the transition graph of $\mathcal{A}$ and $\mathcal{A}'$ are isomorphic.*
(2) *For every configuration (q, s) of $\mathcal{A}$ that is reachable from the initial configuration, the corresponding configuration (q', s') of $\mathcal{A}'$ is such that $q = \chi(q')$, and (q, s) is winning for Éloïse in $\mathbb{G}$ if and only if $q' \in F$.*

In other words, it means that from $\mathbb{G}$ one can build a new game that behaves the same but where the winning region is explicitly marked (thanks to the subset F).

8.2 Logical Consequences

We now discuss the consequences of our main result regarding logical properties of structures generated by CPDA. Due to its strong connections with parity games, we obtain positive results regarding the μ-calculus. Before discussing them, we will start with some consideration regarding monadic second-order (MSO) logic.

For both μ-calculus and MSO logic, it is usual to consider structures given by an edge-labelled graphs coming with a labelling function that maps each vertex to a set of properties that hold in it.

In the setting of CPDA, a natural way to define such a structure is by adding an input alphabet to the CPDA and defining the transition relation as a partial function depending on the current control state, the current top stack symbol and the input letter; the labelling function mapping vertices (*i.e.* configurations) to properties can simply depend on the current control state (as we did when defining the colour in CPDA parity games). Rather than giving a formal definition we give an example that illustrates how to generate an edge-labelled graph using a CPDA with an input alphabet.

Example 8.2. Let $\mathcal{A} = (\Gamma, Q, \Delta, q_0)$ be an order-2 CPDA over the input alphabet $A = \{a, b, c, 1, 2\}$ where $\Gamma = \{\alpha, \beta, \bot\}$, $Q = \{q_0, q_1, q_2\}$ and $\Delta : Q \times \Gamma \times A \to 2^{Q \times Op_2^\Gamma \times Op_2^\Gamma}$ is defined by

- $\Delta(q_0, \bot, 2) = \Delta(q_0, \alpha, 2) = \{(q_1, id; push_2)\}$;
- $\Delta(q_1, \bot, a) = \Delta(q_1, \alpha, a) = \{(q_0, id; push_1^{\alpha,2})\}$;
- $\Delta(q_1, \bot, b) = \Delta(q_1, \alpha, b) = \{(q_2, id; push_1^{\beta,2})\}$;
- $\Delta(q_2, \alpha, 1) = \Delta(q_2, \beta, 1) = \{(q_2, id; pop_1)\}$;
- $\Delta(q_2, \alpha, c) = \Delta(q_2, \beta, c) = \{(q_0, id; collapse)\}$;

Then $\mathcal{A}$ generates the edge labelled graph from Figure 5.

8.3 Monadic Second-Order Logic

We refer the reader to [35] for classical definitions regarding MSO logic over graphs seen as relational structures.

If one restricts its attention to higher-order pushdown automata, *i.e.* CPDA that do not use the *collapse* operation, MSO logic is known to be decidable.

Theorem 8.3. *[17] The structures generated by higher-order pushdown automata have decidable MSO theories.*

The next theorem shows that this is no longer the case for collapsible pushdown automata. In the statement below, FO(TC) is the *transitive closure first-order logic* which is defined by extending

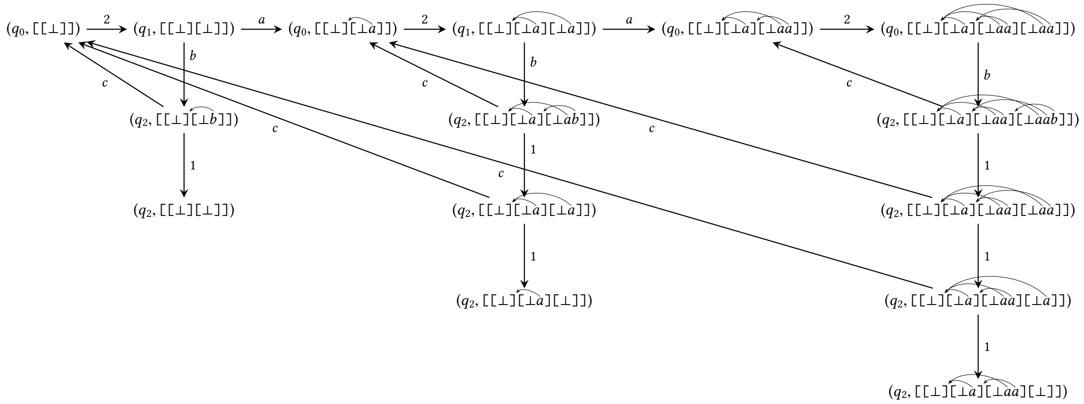

Fig. 5. The edge labelled graph generated from the CPDA with input from Example 8.2.

the first-order logic with a transitive closure operator (see *e.g.* [41]); in particular it subsumes the extension of first-order logic with a reachability predicate.

Theorem 8.4. *There exists a structure generated by a collapsible pushdown automata that has an undecidable MSO theory (actually even an undecidable FO(TC) theory).*

Proof. Consider the following MSO interpretation $\mathcal{I}$[7] (see *e.g.* [18]) applied to the structure defined by the order-2 CPDA from Example 8.2.

$$\begin{array}{rcl}\varphi_A(x,y) & = & x \xrightarrow{C} y \;\wedge\; x \xrightarrow{R} y \\ \varphi_B(x,y) & = & x \xrightarrow{1} y\end{array}$$

with $C = \overline{1}^* \,\overline{b}\, a\, 2\, b\, 1^*$ and $R = c\, 2\, a\, \overline{c} \;\vee\; \overline{1}\, c\, 2\, a\, \overline{c}\, 1$ where a bar-version of an edge label refers to an edge which is taken in the other direction. Hence, C is used to enforce that A-edges occur only between vertices from consecutive columns in the original structure while R is used to enforce that A-edges occurs only between vertices from consecutive rows in the original structure.

We observe that the image of the structure generated by $\mathcal{A}$ by the interpretation $\mathcal{I}$, when restricted to its non-isolated vertices, is the “infinite half-grid” (see Figure 6).

As the infinite (half-) grid has an undecidable MSO theory and as MSO interpretations preserve MSO decidability we conclude that the MSO theory of the structure generated by $\mathcal{A}$ is undecidable.

To refine the result to FO(TC), we simply remark that the interpretation I is FO(TC) definable and that the infinite (half) grid has an undecidable FO(TC) theory [41]. □

Remark 8.5. One can wonder about fragments of MSO weaker than FO(TC), *e.g.* FO(Reach) (the extension of first-order logic with the reachability predicate) or the classical first-order logic (FO). On a positive side, Kartzow proved in [27] that the structures generated by order-2- CPDA have decidable FO(Reach) theories. But moving to order-3 leads to undecidability, even if one restricts to FO, as proved by Broadbent in [4].

The following is a direct consequence of Theorem 8.3 and Theorem 8.4.

[7] In this proof think of an interpretation as a collection of formulas of the form $\varphi_A(x, y)$. Applying such an interpretation to a structure leads to a new structure with the same domain but different transitions: there is an A-labelled edge from x to y in the new structure if and only if $\varphi_A(x, y)$ holds in the original structure.

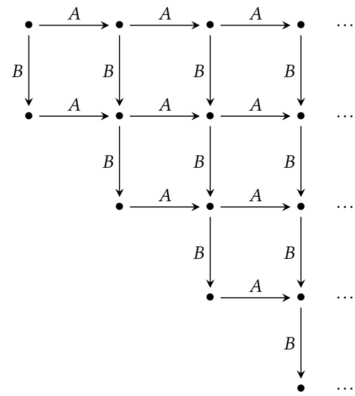

Fig. 6. The "infinite half-grid".

Corollary 8.6. *The class of graphs generated by collapsible pushdown automata strictly contains the class of graphs generated by higher-order pushdown automata.*

8.4 μ-Calculus

We refer the reader to [2] for classical definitions regarding μ-calculus as well as its connections with games.

Due to the tight connection between μ-calculus model-checking and solving parity games, and the fact that the class of structures generated by CPDA is trivially closed by taking a synchronised product with a finite graph, Theorem 3.1 directly leads the following result.

Corollary 8.7. *The following holds.*

(1) *The μ-calculus model-checking problem against structures generated by collapsible pushdown automata is decidable and its complexity (where n denotes the order of the CPDA) is n-times exponential in the number of states of the CPDA, n-times exponential in the alternation depth of greatest and smallest fixpoints in the μ-calculus formula and polynomial in the size of the stack alphabet of the CPDA.*
(2) *The sets of configurations definable by a μ-calculus formula over a graph generated by a collapsible pushdown automata are regular.*

Remark 8.8. In the case of higher-order pushdown automata, links are useless and therefore stacks can be seen as finite words over the alphabet $\Gamma \cup \{[,]\}$ (where Γ denotes the stack alphabet) and regular sets of configurations are regular languages in the traditional sense of finite words. Hence, Corollary 8.7 permits to retrieve the main result in [14, Theorem 6] where the μ-calculus global model-checking problem against higher-order pushdown automata was tackled.

Also note that in this setting, a stronger notion of regularity was introduced in [13] and shown to exactly capture MSO-definable subsets of configurations.

As we did in Section 8.1 to mark winning regions, combining item (2) from Corollary 8.7 together with the fact that the model of CPDA can perform regular test (Theorem 2.8) one directly gets the following result about marking a μ-calculus defined subset of vertices in the transition graph of a CPDA.

Corollary 8.9. *Let $\mathcal{A} = (\Gamma, Q, \delta, q_0)$ be an n-CPDA and let φ be a μ-calculus formula defining a subset of vertices in the transition graph of $\mathcal{A}$. Then, one can build an order-n CPDA $\mathcal{A}'$ with a state-set Q', a subset $F \subseteq Q'$ and a mapping $\chi : Q' \to Q$ such that the following holds.*

(1) *Restricted to the reachable configurations from their respective initial configuration, the transition graph of $\mathcal{A}$ and $\mathcal{A}'$ are isomorphic.*
(2) *For every configuration (q, s) of $\mathcal{A}$ that is reachable from the initial configuration, the corresponding configuration (q', s') of $\mathcal{A}'$ is such that $q = \chi(q')$, and φ holds in (q, s) if and only if $q' \in F$.*

8.5 Perspectives

A natural perspective is to combine the results presented here with the equi-expressivity result [15, 23, 24] between higher-order recursion schemes and collapsible pushdown automaton for generating trees. In particular they imply the decidability of the MSO model-checking problem, both its local [23] and global version (also known as reflection) [8], and the MSO selection problem (a synthesis-like problem) [15].

These results and other consequences are discussed in full detail in a companion paper [7].

REFERENCES

[1] Klaus Aehlig, Jolie de Miranda, and C.-H. Luke Ong. 2005. Safety is not a Restriction at Level 2 for String Languages. In *Proceedings of the 8th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS 2005) (Lecture Notes in Computer Science)*, Vol. 3411. Springer-Verlag, 490–501.

[2] André Arnold and Damian Niwiński. 2001. *Rudiments of mu-Calculus*. Studies in Logic and the Foundations of Mathematics, Vol. 146. Elsevier.

[3] Ahmed Bouajjani and Antoine Meyer. 2004. Symbolic Reachability Analysis of Higher-Order Context-Free Processes. In *Proceedings of the 24th International Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS 2004) (Lecture Notes in Computer Science)*, Vol. 3328. Springer-Verlag, 135–147.

[4] Christopher H. Broadbent. 2012. The Limits of Decidability for First Order Logic on CPDA Graphs. In *Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science (STACS 2012) (LIPIcs)*, Vol. 14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 589–600.

[5] Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, and Olivier Serre. 2012. A Saturation Method for Collapsible Pushdown Systems. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP 2012) (Lecture Notes in Computer Science)*, Vol. 7392. Springer-Verlag, 165–176.

[6] Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, and Olivier Serre. 2013. C-SHORe: a Collapsible Approach to Higher-Order Verification. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming (ICFP 2013)*. ACM, 13–24.

[7] Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, and Olivier Serre. 2020. Higher-Order Recursion Schemes and Collapsible Pushdown Automata: Logical Properties. (2020). https://www.irif.fr/~serre//PublisMisc/BCOS20.pdf

[8] Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. 2010. Recursion Schemes and Logical Reflexion. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LiCS 2010)*. IEEE Computer Society, 120–129.

[9] Thierry Cachat. 2002. Uniform Solution of Parity Games on Prefix-Recognizable Graphs. In *4th International Workshop on Verification of Infinite-State Systems, Infinity 2002 (Electronic Notes in Theoretical Computer Science)*, Vol. 68. Elsevier Science Publishers. Issue 6.

[10] Thierry Cachat. 2003. *Games on Pushdown Graphs and Extensions*. Ph.D. Dissertation. RWTH Aachen.

[11] Thierry Cachat. 2003. Higher Order Pushdown Automata, the Caucal Hierarchy of Graphs and Parity Games. In *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP 2003) (Lecture Notes in Computer Science)*, Vol. 2719. Springer-Verlag, 556–569.

[12] Thierry Cachat and Igor Walukiewicz. 2007. The Complexity of Games on Higher Order Pushdown Automata. *CoRR* abs/0705.0262 (2007).

[13] Arnaud Carayol. 2005. Regular Sets of Higher-Order Pushdown Stacks. In *Proceedings of the 30th Symposium, Mathematical Foundations of Computer Science (MFCS 2005) (Lecture Notes in Computer Science)*, Vol. 3618. Springer-Verlag, 168–179.

[14] Arnaud Carayol, Antoine Meyer, Matthew Hague, C.-H. Luke Ong, and Olivier Serre. 2008. Winning Regions of Higher-Order Pushdown Games. In *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LiCS 2008)*. IEEE Computer Society, 193–204.

[15] Arnaud Carayol and Olivier Serre. 2012. Collapsible Pushdown Automata and Labeled Recursion Schemes: Equivalence, Safety and Effective Selection. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science (LiCS 2012)*. IEEE Computer Society, 165–174.

[16] Arnaud Carayol and Michaela Slaats. 2008. Positional Strategies for Higher-Order Pushdown Parity Games. In *Proceedings of the 33rd Symposium, Mathematical Foundations of Computer Science (MFCS 2008) (Lecture Notes in Computer Science)*, Vol. 5162. Springer-Verlag, 217–228.

[17] Didier Caucal. 2002. On Infinite Terms Having a Decidable Monadic Theory. In *Proceedings of the 27th Symposium, Mathematical Foundations of Computer Science (MFCS 2002) (Lecture Notes in Computer Science)*, Vol. 2420. Springer-Verlag, 165–176.

[18] Bruno Courcelle. 1994. Monadic Second-Order Definable Graph Transductions: A Survey. *Theoretical Computer Science* 126, 1 (1994), 53–75.

[19] E. Allen Emerson and Charanjit S. Jutla. 1991. Tree Automata, mu-Calculus and Determinacy (Extended Abstract). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FoCS 1991)*. IEEE Computer Society, 368–377.

[20] Joost Engelfriet. 1991. Iterated Stack Automata and Complexity Classes. *Information and Computation* 95, 1 (1991), 21–75.

[21] Yuri Gurevich and Leo Harrington. 1982. Trees, Automata, and Games. In *Proceedings of the Fourteenth Annual ACM Symposium on the Theory of Computing (STOC'82)*. ACM, 60–65.

[22] Matthew Hague. 2008. *Global Model Checking of Higher-Order Pushdown Systems*. Ph.D. Dissertation. University of Oxford.

[23] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. 2008. Collapsible Pushdown Automata and Recursion Schemes. In *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LiCS 2008)*. IEEE Computer Society, 452–461.

[24] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. 2017. Collapsible Pushdown Automata and Recursion Schemes. *ACM Transactions on Computational Logic* 18, 3 (2017), 25:1–25:42.

[25] Matthew Hague and C.-H. Luke Ong. 2008. Symbolic Backwards-Reachability Analysis for Higher-Order Pushdown Systems. *Logical Methods in Computer Science* 4, 4 (2008).

[26] Matthew Hague and C.-H. Luke Ong. 2011. A Saturation Method for the Modal μ-Calculus over Pushdown systems. *Information and Computation* 209, 5 (2011), 799–821.

[27] Alexander Kartzow. 2010. Collapsible Pushdown Graphs of Level 2 are Tree-Automatic. In *Proceedings of the 27th Symposium on Theoretical Aspects of Computer Science (STACS 2010) (LIPIcs)*, Vol. 5. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 501–512.

[28] Teodor Knapik, Damian Niwiński, Pawel Urzyczyn, and Igor Walukiewicz. 2005. Unsafe Grammars and Panic Automata. In *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP 2005) (Lecture Notes in Computer Science)*, Vol. 3580. Springer-Verlag, 1450–1461.

[29] Donald A. Martin. 1975. Borel Determinacy. *Ann. Math.* 102, 2 (1975), 363–371.

[30] David. E. Muller and Paul. E. Schupp. 1985. The Theory of Ends, Pushdown Automata, and Second-Order Logic. *Theoretical Computer Science* 37 (1985), 51–75.

[31] Michael O. Rabin. 1969. Decidability of Second-Order Theories and Automata on Infinite Trees. *Trans. Amer. Math. Soc.* 141 (1969), 1–35.

[32] Olivier Serre. 2003. Note on Winning Positions on Pushdown Games with ω-Regular Winning Conditions. *Information Processing Letters* 85 (2003), 285–291.

[33] Olivier Serre. 2004. *Contribution à l'étude des jeux sur des graphes de processus à pile*. Ph.D. Dissertation. Université Paris 7.

[34] Colin Stirling. 2009. Dependency Tree Automata. In *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS 2009) (Lecture Notes in Computer Science)*, Vol. 5504. Springer-Verlag, 92–106.

[35] Wolfgang Thomas. 1997. Languages, Automata, and Logic. In *Handbook of Formal Language Theory*, G. Rozenberg and A. Salomaa (Eds.). Vol. III. Springer-Verlag, 389–455.

[36] Moshe Y. Vardi. 1998. Reasoning about The Past with Two-Way Automata. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming (ICALP 1998) (Lecture Notes in Computer Science)*, Vol. 1443. Springer-Verlag, 628–641.

[37] Igor Walukiewicz. 1996. Pushdown Processes: Games and Model Checking. In *Proceeding of the 8th International Conference on Computer Aided Verification (CAV 1996) (Lecture Notes in Computer Science)*, Vol. 1102. Springer-Verlag, 62–74.

[38] Igor Walukiewicz. 2001. Pushdown Processes: Games and Model-Checking. *Information and Computation* 157 (2001), 234–263.

[39] Igor Walukiewicz. 2004. A Landscape with Games in the Background. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LiCS 2004)*. Computer Society Press, 356–366.
[40] Thomas Wilke. 2001. Alternating Tree Automata, Parity Games and Modal μ-Calculus. *Bulletin of the Belgian Mathematical Society* 8, 2 (2001), 359–391.
[41] Stefan Wöhrle and Wolfgang Thomas. 2007. Model Checking Synchronized Products of Infinite Transition Systems. *Logical Methods in Computer Science* 3, 4 (2007).