On Global Model Checking Trees Generated by Higher-Order Recursion Schemes

Christopher Broadbent and Luke Ong

Oxford University Computing Laboratory

Abstract. Higher-order recursion schemes are systems of rewrite rules on typed non-terminal symbols, which can be used to define infinite trees. The *Global Modal Mu-Calculus Model Checking Problem* takes as input such a recursion scheme together with a modal μ-calculus sentence and asks for a finite representation of the set of nodes in the tree generated by the scheme at which the sentence holds. Using a method that appeals to game semantics, we show that for an order-n recursion scheme, one can effectively construct a non-deterministic order-n collapsible pushdown automaton representing this set. The level of the automaton is strict in the sense that in general no non-deterministic order-$(n-1)$ automaton could do likewise (assuming the requisite hierarchy theorem). The question of determinisation is left open. As a corollary we can also construct an order-n collapsible pushdown automaton representing the constructible winning region of an order-n collapsible pushdown parity game.

Keywords: Recursion Scheme, Model Checking, Game Semantics, Collapsible Pushdown Automaton, Parity Game.

1 Introduction

Whilst *local* model checking asks whether a property holds at the root of a structure, a *global* model checking algorithm is designed to return a finite representation of the set of states in a structure at which a property holds.

Our own focus is on model checking modal μ-calculus properties of (possibly infinite) ranked trees generated by *higher-order recursion schemes*, which are systems of rewrite rules on typed non-terminals. A number of results exist concerning the local version [13,14] and it turns out that for an order-n recursion scheme the local problem is n-EXPTIME complete [14]. The computationally intensive part of our algorithm for the global result in fact consists of solving a local version of the problem. We have to compute the winning region of a finite parity game arising from Ong's method [14]. Algorithms for solving such games from a given node usually follow the global paradigm and compute the winning region in the process.

Owing to equivalences between recursion schemes and various flavours of higher-order pushdown automata (PDA) [13,10], the present work is very much related to computing winning regions of parity games played over the configuration graphs of such automata. Cachat and Serre independently showed that the winning regions of parity games over 1-PDA and prefix-recognizable graphs are regular [16,2]. Piterman and Vardi [15] have also presented a generalisation of automata-theoretic techniques used to solve the local problem over these graphs to obtain the same result. (Indeed we borrow an aspect of their method with what we call in the sequel *'the versatile automaton'*.) It was subsequently discovered by Carayol et al. that the winning regions of order-n pushdown parity games are regular and that the problem is n-EXPTIME complete [4]. As a corollary to this they show that the global model checking problem for order-n recursion schemes satisfying a syntactic constraint called *'safety'* can be solved in n-EXPTIME, with solution represented by a *deterministic* order-n pushdown automaton.

An analogous approach to general recursion schemes would require a regular characterisation of the winning region of a *collapsible pushdown* parity game [10], as provided by Serre[1]. The approach we consider here, however, does not go via collapsible pushdown parity games. Despite the difference in method, our final result is similar insofar as our algorithm represents the required set of tree nodes using an order-n collapsible pushdown automaton (CPDA). There is an unfortunate difference, however, in that our CPDA is *non-deterministic*. Even if this diminishes the practical utility of the output of our algorithm, our result nevertheless establishes that the μ-calculus definable node-sets of trees generated by order-n recursion schemes can themselves be generated by an order-n recursion scheme. In doing so we show how two different incarnations of a game-semantic approach to the Local Problem [14,10] can be merged.

As a corollary we are able to characterise the configurations with *constructible stacks* that are winning in a collapsible pushdown parity game using a CPDA. Constructible stacks are represented by sequences of stack operations that generate them. This resembles a similar result by Carayol and Slaats for (non-collapsible) PDA [5], although our version lacks the canonicity exhibited in *op. cit.*

An Outline Proof of the Local Problem. Space constraints limit the degree to which we can introduce apparatus from Ong's original paper [14] but we try to refer to a section of the *long version* for the reader interested in more details.

Fix a space of types formed from a single ground type o and the arrow-constructor $\rightarrow$. The order and arity of a type are given their standard definitions. An ***order-n recursion scheme*** ([14], Sec. 1.2) is a 5-tuple $\langle \Sigma, \mathcal{N}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle$ where Σ is a finite ranked alphabet with a symbol of arity k given the order-1 type of arity k; $\mathcal{N}$ is a finite set of non-terminals assigned types of order no greater than n; $\mathcal{V}$ is a finite set of typed variables; $\mathcal{S} \in \mathcal{N}$ is a distinguished 'initial symbol' with type o and $\mathcal{R}$ is a finite set of rewrite rules of the form $F\zeta_1 \ldots \zeta_m \rightarrow t(\zeta_1, \ldots, \zeta_m)$ where $F \in \mathcal{N}$ and $\zeta_1, \ldots, \zeta_m \in \mathcal{V}$; $F\zeta_1, \ldots, \zeta_m$ has type o as does the term $t(\zeta_1, \ldots, \zeta_m)$, which is formed from variables $\zeta_1, \ldots, \zeta_m$, non-terminals from $\mathcal{N}$ and Σ-symbols. There should be precisely one rule for each non-terminal. The ***value-tree*** $[\![G]\!]$ defined by a recursion scheme G is the tree with nodes labelled in Σ that is the limit of the recursion scheme as it unfolds from $\mathcal{S}$ ([14], p. 7).

Given a μ-calculus sentence ϕ and a recursion scheme G, the local problem asks whether ϕ holds at the root of $[\![G]\!]$. Ong's proof of decidability for this [14] makes use of ideas from innocent game semantics [11] via the notion of ***traversal***. A traversal is a sequence of nodes obeying certain rules in an infinite lambda term $\lambda(G)$, called the

[1] Private communication with Olivier Serre, 7 October 2008.

computation tree, which represents the recursion scheme G ([14], Sec. 2). The manner in which a traversal hops around the computation tree can be viewed as both a form of evaluation of the scheme (linear head reduction) and a manifestation of its game-semantic denotation.

Thanks to Emerson and Jutla [7], we can convert the μ-calculus sentence ϕ to an *alternating parity tree automaton (APT)* $\mathcal{B}$, which we refer to as ***the property APT***, such that $\mathcal{B}$ has an accepting run-tree on the value-tree $[\![G]\!]$ generated by G just in case $[\![G]\!] \vDash \phi$.[2] We can map accepting run-trees to accepting ***traversal trees*** (and *vice versa*), where the latter allow $\mathcal{B}$ to jump over $\lambda(G)$ according to the rules for traversals ([14], Definition 2.11). We can then simulate such traversal trees using a ***traversal simulating APT*** $\mathcal{C}$ that reads $\lambda(G)$ in a 'normal' top-down manner ([14], Sec. 3). Since $\lambda(G)$ is regular, as witnessed by a finite graph $\mathrm{Gr}(G)$ ([14], p. 51), it can be decided whether $\mathcal{C}$ accepts $\lambda(G)$ and this gives the result.

Overview. Fix a ranked alphabet Σ, a tree-generating recursion scheme G and a μ-calculus sentence ϕ. Let $\mathcal{B}$ be the *property APT* associated with ϕ. The *Global Model-Checking Problem* asks for a finite representation of the set of nodes in the Σ-labelled tree $[\![G]\!]$ at which ϕ holds. We explicate a method that, given an order-n recursion scheme, constructs an n-CPDA word acceptor that accepts precisely these nodes, where nodes are represented in the standard way as strings over a 'directions alphabet'.

We actually establish a slightly stronger result than we need. We provide a finite representation of the set of ordered pairs (q, α), where q is a state of $\mathcal{B}$ and α is a node of $[\![G]\!]$, such that $\mathcal{B}$ accepts the subtree of $[\![G]\!]$ rooted at α starting from state q. The solution to the Global Model-Checking Problem for G and ϕ is then provided by restricting this set to those pairs of the form (q_0, α), where q_0 is the initial state of $\mathcal{B}$.

The construction begins with what we describe as the *versatile property APT*, $\mathcal{B}^\perp$, which is able to navigate to an arbitrary node in $[\![G]\!]$ before proceeding to adopt the behaviour of $\mathcal{B}$ starting at an arbitrary state q. Since $\mathcal{B}^\perp$ is just an ordinary APT, there exists a *traversal-simulating* APT $\mathcal{C}^\perp$ for $\mathcal{B}^\perp$. We can thus move on to consider the *finite* parity game $\mathcal{G}_{G,\mathcal{C}^\perp}$ induced by $\mathcal{C}^\perp$ and the computation graph $\mathrm{Gr}(G)$ of G. The two players of parity games are named Éloïse and Abelard. Éloïse can be viewed as trying to establish a μ-calculus formula whilst Abelard is trying to refute it. We can use a standard algorithm to find the winning region of $\mathcal{G}_{G,\mathcal{C}^\perp}$ and thereby label with a symbol 'W' the nodes of $\mathcal{G}_{G,\mathcal{C}^\perp}$ from which Éloïse has a winning strategy. The annotated graph is called $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$.

Since $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$ is induced (in part) by $\mathrm{Gr}(G)$, it makes sense to speak of traversals over $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$. Consider the set of traversals of $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$ travelling only over nodes labelled with W and halting at a node corresponding to a point where $\mathcal{B}^\perp$ starts to simulate $\mathcal{B}$ from state q at node n of the tree. It turns out that this set, when projected to the Σ-labelled nodes, corresponds to the set of ordered pairs (q, α) that we want to finitely represent. Since we can program an n-CPDA to navigate traversals of $\mathrm{Gr}(G)$ [10], we can also program it to navigate the traversals of $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$ in the set. This provides the requisite n-CPDA word acceptor.

[2] For an introduction to the modal μ-calculus and parity automata / games we direct the reader to Bradfield and Stirling's survey [1].

2 The Versatile Property APT and Its Simulation

By convention the nodes of a (ranked and ordered) Σ-labelled tree, $T : dom(T) \longrightarrow \Sigma$ (say), are represented in the standard way by strings in Dir^* where $\mathrm{Dir} = \mathbb{N}$, so that $dom(T) \subseteq \mathrm{Dir}^*$; elements of $\mathrm{Dir} \cup \{\epsilon\}$ (ϵ the empty string) are referred to as ***directions***. Thus the label of a node α is $T(\alpha)$ (e.g. $[\![G]\!](\alpha)$ means the label of the node α in a value-tree $[\![G]\!]$). A *path* p in a tree is viewed as a sequence of nodes such that the successor of an element of a sequence is its child. The *trace* $\mathrm{trace}(p)$ of a path $p = (p_i)_{i \in I}$ is the sequence $(T(p_i))_{i \in I}$. For a node r of a tree T we write T_r for the maximal subtree of T rooted at r.

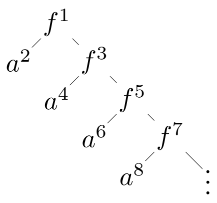


Fig. 1. A Value Tree

An APT that operates over a Σ-labelled tree T is a 5-tuple $\langle\, \Sigma, Q, \delta, q_0, \Omega \,\rangle$ consisting of a finite-set Q of control-states, transition function $\delta : (\Sigma \times Q) \to B^+((\mathrm{Dir} \cup \{\epsilon\}) \times Q)$ (where $B^+(S)$ is the set of positive boolean formulae with set of atoms S), initial state $q_0 \in Q$ and priority function $\Omega : Q \to \mathbb{N}$. Whilst reading a node u of T in state q the automaton will pick a *minimal* set $S \subseteq ((\mathrm{Dir} \cup \{\epsilon\}) \times Q)$ satisfying $\delta(T(u), q)$, and for each $(i, q') \in S$ will spawn an automaton in state q' reading the ith child of u, the ϵ-child of a node being itself. A ***run-tree*** of the APT is an unranked $(dom(T) \times Q)$-labelled tree representing such a branching run starting in state q_0 at the root of T. It is deemed accepting if the Q-projection $q_1\ q_2\ \ldots$ of the trace of every path satisfies ***the parity condition*** meaning that $\max(\{\Omega(q)\ :\ q \in \inf(q_1\ q_2\ \ldots)\})$ is even, where $\inf(\sigma)$ is the set of states occurring infinitely often in σ.

Suppose that the property APT $\mathcal{B}$ has initial state q_0 so that $\mathcal{B} = \langle\, \Sigma, Q, \delta, q_0, \Omega \,\rangle$. A ***template for an APT*** is a quadruple $\mathfrak{B} = \langle\, \Sigma, Q, \delta, \Omega \,\rangle$ with Q a finite set such that for any $q \in Q$ it is the case that $\mathcal{B}_q = \langle\, \Sigma, Q, \delta, q, \Omega \,\rangle$ is an APT. We may view $\mathfrak{B}$ as representing the family of automata: $\mathfrak{B} = \{\, \mathcal{B}_q\ :\ q \in Q \,\}$.

Consider a ranked and ordered tree T. It is possible to construct an automaton $\mathcal{B}_\perp$ that can behave as any member of $\mathfrak{B}$ acting on any ranked and ordered subtree of T. We call this automaton *the versatile property APT for* $\mathfrak{B}$. The versatile APT traverses the tree T starting at its root whilst in a kind of 'nascent state' $\perp$. Once it reaches the desired node r of the tree, it switches into the required state q and starts behaving as though it were $\mathcal{B}_q$. We call this point *q-initialisation.*

Definition 1. *Let* $\mathfrak{B} = \langle\, \Sigma, Q, \delta, \Omega \,\rangle$ *be a template for an APT. The* ***versatile automaton*** $\mathcal{B}^\perp$ *for* $\mathfrak{B}$ *is the APT* $\mathcal{B}^\perp$ *given by:*

$$\mathcal{B}^\perp = \left\langle \Sigma, Q \uplus \{\perp\}, \delta^\perp, \perp, \Omega^\perp \right\rangle$$

where $\delta^\perp$ *extends* δ *by the rule:* $\delta^\perp : (\perp, f) \mapsto \bigvee_{1 \le i \le \mathrm{ar}(f)} (i, \perp) \vee \bigvee_{q \in Q} (\epsilon, q)$ *and* $\Omega^\perp$ *extends* Ω *with* $\Omega^\perp(\perp) := -1$.

So the APT $\mathcal{B}^\perp$ has an 'initialisation phase' during which it is in state $\perp$.

Definition 2. *Let* t *be a run-tree of the versatile APT* $\mathcal{B}^\perp$ *on a* Σ*-labelled tree* T*. Let* $t^\perp$ *be the unique path in* t *consisting of precisely the nodes associated with* $\perp$*.*

*Let p be the unique path in T corresponding to the path in t of the form $t^{\perp}\ \beta$ where β is a node in t with label (α, q) such that $q \in Q$, the state space of the template $\mathfrak{B}$ (i.e. $q \neq \perp$). We then say that q-**initialization occurs at (the path)** p or **at (the node)** β. We call the path $t^{\perp}$ **the initialisation phase** of the automaton (during the run t).*

The following lemma summarises the significance of $\mathcal{B}^{\perp}$.

Lemma 1. *Let $\mathcal{B}^{\perp}$ be a versatile automaton and let T be a Σ-labelled tree. Given a state q of $\mathfrak{B}$ and a node r of T, it is the case that $\mathcal{B}_q$ accepts T_r if and only if $\mathcal{B}^{\perp}$ has an accepting run-tree on T with q-initialisation taking place at r.*

Example 1. We use the following automaton as our working example. It acts on trees with nodes labelled by f and a with arities 2 and 0 respectively. It has state space $\{q_0, q_1, q_2\}$ each of which is given priority 2.

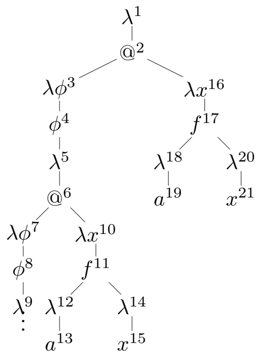


Fig. 2. A Computation Tree

$$(f, q_0) \mapsto (1, q_1) \wedge (2, q_1) \qquad (a, \_) \mapsto \mathrm{t}$$
$$(f, q_1) \mapsto (1, q_1) \wedge (1, q_2) \wedge (2, q_1)$$

We additionally use as an example the recursion scheme with initial non-terminal $\mathcal{S} : o$, non-terminal $F : (o \to o) \to o$ and rules: $\mathcal{S} \to F(fa)$ and $F\phi \to \phi(F(fa))$. The scheme's value tree and computation tree are illustrated in Figures 1 and 2 respectively.

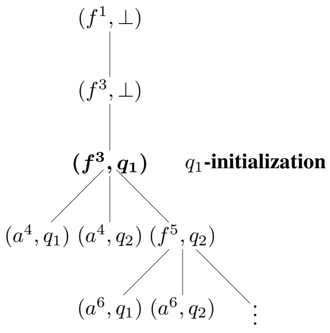


Fig. 3. A Run-Tree of a Versatile APT

Example 2. The versatile APT for the property APT in Example 1 has state space $\{q_0, q_1, q_2\} \cup \{\perp\}$ with $\perp$ the initial state. All states of the form q_i have priority 2 but $\perp$ has priority -1. Its transition function is given by:

$$(f, q_0) \mapsto (1, q_1) \wedge (2, q_1) \qquad (f, \perp) \mapsto (1, \perp) \vee (2, \perp) \vee \bigvee_{0 \leq i \leq 2} (\epsilon, q_i)$$
$$(f, q_1) \mapsto (1, q_1) \wedge (1, q_2) \wedge (2, q_1) \qquad (a,\ q_{\_}) \mapsto \mathbf{t}$$

with $\mathbf{t}$ the positive boolean formula that is always true (i.e. the empty conjunction) and $\mathbf{f}$ is that which is always false (i.e. the empty disjunction).

A run-tree of this versatile APT on the value tree in Figure 1 is given in Figure 3.

Traversals and the Versatile APT. Now consider a Σ-labelled tree $[\![G]\!]$ generated by some higher-order recursion scheme G. Let us fix a versatile APT $\mathcal{B}^{\perp}$ that can run on Σ-labelled trees.

We now make use of the notions of *traversals* and *the traversal tree* of an APT on the computation tree $\lambda(G)$ of the recursion scheme. We speak interchangeably of traversals over the computation graph $\mathrm{Gr}(G)$, which unravels to form $\lambda(G)$. The *Path-Traversal Correspondence Theorem* from the proof of the decidability of the Local Model-Checking Problem ([14], Thm. 7) ensures that the following definition is well-defined, which for any node α in $[\![G]\!]$ gives the corresponding node α^{Λ} in $\lambda(G)$:

Definition 3. *Let $\alpha = a_1 \ldots a_m$ (with $a_i \in \mathrm{Dir}$ for $1 \leq i \leq m$) be a node in $[\![G]\!]$. Let t_α be a traversal of $\lambda(G)$ and for $1 \leq i \leq m+1$ let us name v_i the i-th occurrence of a terminal-labelled node in t_α. Suppose that v_1 bears the same label as the root of $[\![G]\!]$ and for $i \geq 2$, $[\![G]\!](a_1 \ldots a_{i-1}) = [\![\lambda(G)]\!](v_i)$ (where $a_1 \ldots a_{i-1}$ is a node in $[\![G]\!]$). Further assume that for $1 \leq i \leq m$ the successor of v_{i-1} in t_α is its a_i-th child in $\lambda(G)$. We define α^{Λ} to be v_{m+1}.*

Note that α^{Λ} also makes sense when speaking of traversals over $\mathrm{Gr}(G)$ except that in this case it should be viewed as a *particular instance* of a $\mathrm{Gr}(G)$-node in a traversal.

We can also speak of q-initialisation in a traversal tree of $\mathcal{B}^{\perp}$ in a completely analogous way – q-initialisation is the point in the traversal at which the automaton switches from being in state $\perp$ to being in state q. We illustrate in Figure 4 a traversal tree of the versatile APT in Example 2 on the computation tree in Figure 2.

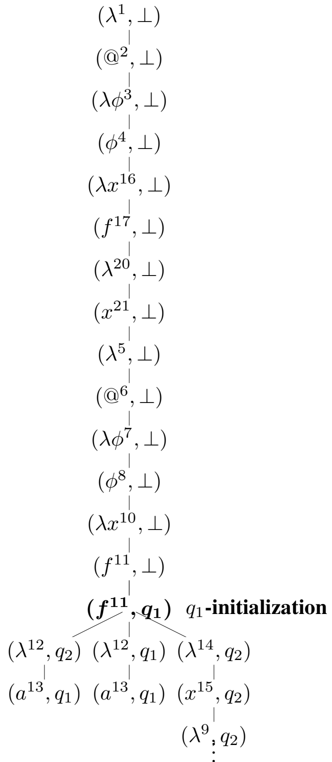
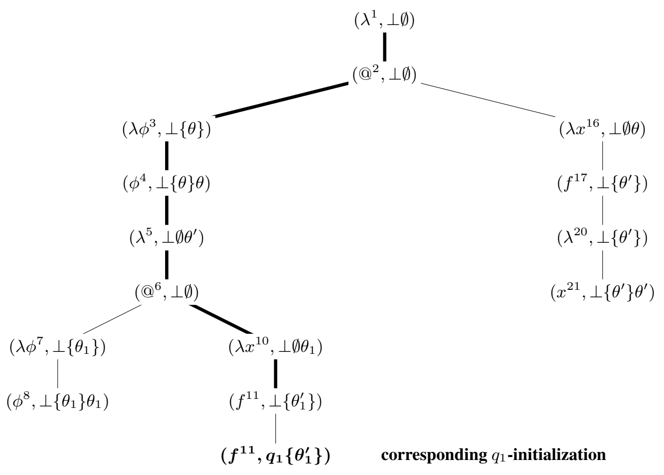


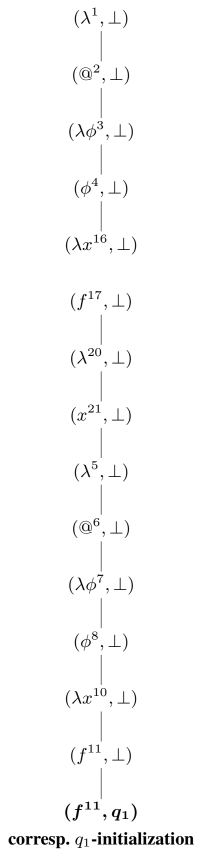
Fig. 4. A Traversal Tree of a Versatile APT

Now we make use of the ***traversal-simulating APT*** $\mathcal{C}^{\perp}$ associated with $\mathcal{B}^{\perp}$ (in the sense of Ong ([14], sec. 3)). The essential property of $\mathcal{C}^{\perp}$ is that it is possible to convert an accepting *traversal tree* of $\mathcal{B}^{\perp}$ on $\lambda(G)$ into an accepting *run-tree* of $\mathcal{C}^{\perp}$ on $\lambda(G)$ and conversely an accepting *run-tree* of $\mathcal{C}^{\perp}$ can be converted into an accepting *traversal tree* of $\mathcal{B}^{\perp}$.

Each state s of $\mathcal{C}^{\perp}$ includes a component $\mathrm{sim}(s)$ which is the state of $\mathcal{B}^{\perp}$ that is being simulated. Similarly for a sequence of states $\sigma = (s_i)_{i \in X}$ we write $\mathrm{sim}(\sigma)$ to denote the sequence $(\mathrm{sim}(s_i))_{i \in X}$. In contrast to $\mathcal{B}^{\perp}$, however, the transition function of $\mathcal{C}^{\perp}$ maps $\perp$-states to boolean formulae that are not necessarily disjunctions so that it can both guess how the simulated traversal will evolve as well as later verify these guesses. As a result q-initialization cannot be considered unique for $\mathcal{C}^{\perp}$ and so we adjust the definition appropriately.

Definition 4. *Let t be a run-tree of $\mathcal{C}^{\perp}$ on $\lambda(G)$ and q be a state of $\mathfrak{B}$. For a* finite *path p in $\lambda(G)$ (starting at the root), we say that* ***an instance of q-initialisation occurs at*** *p if there exists some path $t^{\perp}$ in t such that $\mathrm{sim}(\mathrm{trace}(t^{\perp}))$ consists only of $\perp$ and β is some node such that $t^{\perp}\,\beta$ is also a path in t with the projection of $\mathrm{trace}(t^{\perp})$ onto the*

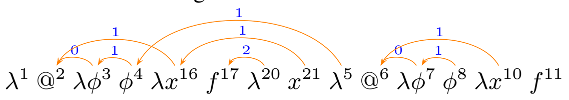

$(\lambda^1, \bot\emptyset)$
$(@^2, \bot\emptyset)$
$(\lambda\phi^3, \bot\{\theta\})$
$(\phi^4, \bot\{\theta\}\theta)$
$(\lambda^5, \bot\emptyset\theta')$
$(@^6, \bot\emptyset)$
$(\lambda\phi^7, \bot\{\theta_1\})$
$(\phi^8, \bot\{\theta_1\}\theta_1)$
$(\lambda x^{10}, \bot\emptyset\theta_1)$
$(f^{11}, \bot\{\theta'_1\})$
$(f^{11}, q_1\{\theta'_1\})$
corresponding q_1-initialization
$(\lambda x^{16}, \bot\emptyset\theta)$
$(f^{17}, \bot\{\theta'\})$
$(\lambda^{20}, \bot\{\theta'\})$
$(x^{21}, \bot\{\theta'\}\theta')$



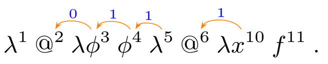
Part of a run-tree of the traversal-simulating APT on $\lambda(G)$. The states of the traversal-simulating automaton are either of the form $q\theta$ or $qS\theta$ where q represents the property APT state being simulated and S and θ describe how the automaton has guessed the traversal being simulated should evolve ([14], Sec. 3). The fragment of the run-tree illustrated here is precisely the fragment that corresponds to the fragment of the traversal tree illustrated to the right.

The part of the traversal tree up to the point of q-initialisation.

The traversal associated with the two diagrams above is:


$\lambda^1\ @^2\ \lambda\phi^3\ \phi^4\ \lambda x^{16}\ f^{17}\ \lambda^{20}\ x^{21}\ \lambda^5\ @^6\ \lambda\phi^7\ \phi^8\ \lambda x^{10}\ f^{11}$
0
1
1
1
1
2
0
1
1

which has P-View:


$\lambda^1\ @^2\ \lambda\phi^3\ \phi^4\ \lambda^5\ @^6\ \lambda x^{10}\ f^{11}$.
0
1
1
1

The path corresponding to this P-view in the traversal-simulating APT run-tree has been highlighted.

Fig. 5. An illustration of Lemma 2

nodes of $\lambda(G)$ equal to p and β labelled (s, α) where α is the last node in p and s is a state of $\mathcal{C}^{\perp}$ such that $\mathrm{sim}(s) = q$.

Given a sequence of nodes $s := (\alpha_1, p_1) \ldots (\alpha_m, p_m)$ in $\mathrm{Gr}(G) \times Q_{\mathcal{C}^{\perp}}$ let us write $\pi_{\mathrm{Gr}(G)}(s)$ to mean $\alpha_1, \ldots, \alpha_m$ and $\pi^{\mathrm{NDup}}_{\mathrm{Gr}(G)}(s)$ to mean the largest subsequence of $\pi_{\mathrm{Gr}(G)}(s)$ whose adjacent elements are pairwise distinct. Given a traversal t we write $\ulcorner t \urcorner$ to denote its *P-view* ([14], Def. 2.5) which is a subsequence of t of a certain game-semantic significance.

Lemma 2. *Let α be a node in $[\![G]\!]$ and q be a state of $\mathfrak{B}$. The following are equivalent:*

1. *Property APT $\mathcal{B}_q \in \mathfrak{B}$ accepts $[\![G]\!]_\alpha$.*
2. *The APT $\mathcal{B}^{\perp}$ has an accepting traversal tree t with q-initialisation occurring at α^{Λ}.*
3. *There exists a finite subtree T_α of a run-tree of $\mathcal{C}^{\perp}$, all of whose nodes are associated with states simulating $\perp$. For some traversal t_α on $\lambda(G)$ ending in α^{Λ} it is the case that*

   $$\{\pi^{\mathrm{NDup}}_{\mathrm{Gr}(G)}(p) : p \text{ is a path in } T_\alpha\} = \{\ulcorner \pi_{\mathrm{Gr}(G)}(s) \urcorner : s \text{ is an initial segment of } t_\alpha\} \,.$$

   In particular there is a maximal branch b in T_α such that $\ulcorner \pi^{\mathrm{NDup}}_{\mathrm{Gr}(G)}(b) \urcorner = \ulcorner t_\alpha \urcorner$. Moreover, T_α can be extended to an accepting run-tree of $\mathcal{C}^{\perp}$ such that q-initialisation occurs on the tip of b.

Proof. The equivalence of 1 and 2 is given by the Path-Traversal Correspondence Theorem of Ong ([14], Thm 7). The equivalence of 2 and 3 is given by the inter-translations between traversal trees of $\mathcal{B}^{\perp}$ and run-trees of $\mathcal{C}^{\perp}$ given in *op. cit.* (Sec 4 and 5) together with the result from that same paper ([14], Prop. 6) stating that there is a $1-1$ correspondence between P-views of traversals and paths in the computation tree. This is illustrated in Figure 5. □

3 The Versatile Parity Game

We now move on to consider the parity game induced by $\mathcal{C}^{\perp}$ acting on $\mathrm{Gr}(G)$. Let us call this parity game the *versatile parity game* $\mathcal{G}_{G,\mathcal{C}^{\perp}}$. To retain a simple description, we assume that $\mathcal{C}^{\perp}$ is presented in such a form that the image of its transition function consists of just pure disjunctions and pure conjunctions. Let us write

$$\mathcal{C}^{\perp} \quad := \quad \langle \Lambda_G, Q_{\mathcal{C}^{\perp}}, \delta_{\mathcal{C}^{\perp}}, p_{0\mathcal{C}^{\perp}}, \Omega_{\mathcal{C}^{\perp}} \rangle$$

for the traversal-simulating automaton in such a form. This means that every element in the image of $\delta_{\mathcal{C}^{\perp}}$ can be written as $\bigwedge_{i \in I}(d_i, p_i)$ or $\bigvee_{i \in I}(d_i, p_i)$.

Definition 5. *Let N be the set of nodes in* $\mathrm{Gr}(G)$ *and let $Q_{\mathcal{C}^{\perp}}$ be the state space of $\mathcal{C}^{\perp}$. The* ***versatile parity game*** *is the parity game played on a directed graph with nodes in $N \times Q_{\mathcal{C}^{\perp}}$ such that:*

1. *The start node of the game is* $(n_0, p_{0\mathcal{C}^\perp})$ *where* n_0 *is the root of* $\mathrm{Gr}(G)$.
2. *There is an edge from* (n, p) *to* (n', p') *just in case*

$$\delta_{\mathcal{C}^\perp}(l, p) = \bigwedge_{i \in I}(d_i, p_i) \quad \text{or} \quad \delta_{\mathcal{C}^\perp}(l, p) = \bigvee_{i \in I}(d_i, p_i)$$

   where l *is the label of* n *and for some* $i \in I$, $p_i = p'$ *and* n' *is the* d_i*th child of* n. *Note that we may have* $d_i = \epsilon$ *(the automaton does not move in the tree), in which case* $n' = n$.
3. *A game node* (n, p) *is owned by Éloïse if it is mapped by* $\delta_{\mathcal{C}^\perp}$ *onto a* $\bigvee$*-formula and is owned by Abelard if it is mapped onto a* $\bigwedge$*-formula.*
4. *The only nodes in the game are those reachable from its start node.*
5. *The priority of a node* (n, p) *is* $\Omega_{\mathcal{C}^\perp}(p)$.

We write $\mathcal{G}_{G,\mathcal{C}^\perp}$ *to denote this parity game and let us write* $\mathbf{legalMove}((n, p), (n', p'))$ *if there is an edge from* (n, p) *to* (n', p').

We refer to a run-tree of $\mathcal{C}^\perp$, whose nodes associated with a $\bigvee$-state must have a unique child, as ***strategies for Éloïse*** in the game $\mathcal{G}_{G,\mathcal{C}^\perp}$. Such a strategy is termed ***winning*** just in case it is an accepting run-tree. A *finite* subtree of a strategy is called a ***partial strategy***.

By the definition of a traversal-simulating APT we may assume the following w.l.o.g:

Lemma 3

1. *Suppose that* (n, p) *is a node in* $\mathcal{G}_{G,\mathcal{C}^\perp}$ *such that the label of* n *is either an* $@$ *symbol or a variable* ϕ. *It is then the case that* (n, p) *is owned by Abelard and* $n' \neq n$ *for* every (n', p') *such that* $\mathbf{legalMove}((n, p), (n', p'))$ *if and only if there exists* some (n', p') *such that* $n \neq n'$ *and* $\mathbf{legalMove}((n, p), (n', p'))$.
2. *A game node* (n, p) *with* n *labelled by a terminal* $f \in \Sigma$ *and* $\mathrm{sim}(p) = \perp$ *is owned by Éloïse. The successor nodes of* (n, p) *include nodes of the form:*
   (a) (n, p_q) *with* $\mathrm{sim}(p_q) = q$ *for each* $q \in Q$, *the state space of* $\mathcal{B}$.
   (b) (n_i, p_i) *for* $1 \leq i \leq \mathrm{ar}(f)$ *where* n_i *is the* i*th child of* n *and* $\mathrm{sim}(p_i) = \perp$.
3. *Any node* (n, p) *in* $\mathcal{G}_{G,\mathcal{C}^\perp}$ *such that the label of* n *is a* λ*-node is owned by Éloïse.*

We can extend the notion of traversal on $\mathrm{Gr}(G)$ (or $\lambda(G)$) to $\mathcal{G}_{G,\mathcal{C}^\perp}$. Such traversals must respect the edge-relation of $\mathcal{G}_{G,\mathcal{C}^\perp}$ in the sense that they could be 'reassembled' into a tree embeddable in $\mathcal{G}_{G,\mathcal{C}^\perp}$.

Definition 6. *Consider a finite sequence of nodes* $(n_1, p_1), \ldots, (n_m, p_m)$ *in* $\mathcal{G}_{G,\mathcal{C}^\perp}$ *such that an element* (n_i, p_i) *might be endowed with an integer labelled pointer to an element* (n_j, p_j) *for* $1 \leq j < i$. *We say that such a sequence is a* ***traversal of*** $\mathcal{G}_{G,\mathcal{C}^\perp}$ *just in case all of the following conditions hold:*

1. (n_1, p_1) *is the initial node of* $\mathcal{G}_{G,\mathcal{C}^\perp}$ *(so* n_1 *will have label* λ*).*
2. *The sequence* $n_1, \ldots, n_m$ *together with pointers is a traversal of* $\mathrm{Gr}(G)$, *which we refer to as the* underlying traversal.

3. *Suppose that the traversal includes an instance of*

$$\begin{tikzcd}[column sep=small] \ldots (n,p) & \ldots & (n',p') \arrow[ll, bend right, "i"'] & \ldots \end{tikzcd}$$

where n' is a λ-node, or $\ldots\ (n,p)\ (n',p')\ \ldots$ where n is a λ-node and (n',p') may or may not source a pointer.
We require that there is a path from (n,p) to (n',p') in $\mathcal{G}_{G,\mathcal{C}^{\perp}}$. Note that this path will necessarily be of the form

$$(n,p),\ (s_1,p_1),\ \ldots,\ (s_l,p_l),\ (s_{l+1},p_{l+1}),\ \ldots,\ (s_k,p_k),\ (n',p')$$

for some $l,k \in \mathbb{N}$ such that for all $1 \le i \le l$ we have the label of s_i being the label of n and for all $l+1 \le j \le k$ we have the label of s_j being the label for n'.
4. *Every occurrence of (n_i,p_i) such that n_i is an @ or a variable node is owned by Abelard.*

Our definition of traversal respects the rules of $\mathcal{G}_{G,\mathcal{C}^{\perp}}$, equivalently the transition function of $\mathcal{C}^{\perp}$, in the following sense:

Lemma 4. *For every traversal t of $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ there exists a partial strategy T for Éloïse in $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ such that*

$$\{\pi^{\mathrm{NDup}}_{\mathrm{Gr}(G)}(\mathrm{trace}(r))\ :\ r \textit{ is a path in } T\} = \{\ulcorner \pi_{\mathrm{Gr}(G)}(s) \urcorner\ :\ s \textit{ is an initial segment of } t\}\ .$$

Traversals of $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ consisting of nothing but nodes (n,p) with $\mathrm{sim}(p) = \perp$ are particularly pleasant because nodes associated with a terminal $f \in \Sigma$ never occur in immediate succession; they also allow access to arbitrary children of the f labelled node in $\mathrm{Gr}(G)$. We are also interested in such traversals that then finish with a node (n,p) with $\mathrm{sim}(p) = q$ for $q \in Q$, the state space of $\mathcal{B}$.

Definition 7. *A $\perp$-**traversal** of $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ is a traversal of $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ consisting entirely of nodes (n,p) that satisfy $\mathrm{sim}(p) = \perp$. If q is a state of $\mathcal{B}$, then a q-**tipped traversal** of $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ is a traversal of the form $s\ (n,p)$ where s is a $\perp$-traversal but $\mathrm{sim}(p) = q$.*

Using known algorithms (such as Jurdziński's [12]) we can compute the winning region for Éloïse of finite parity games. We may thus effectively annotate with the symbol 'W' the states of $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ from which Éloïse has a winning strategy. Let us write $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^{\perp}}$ for this annotated game and refer to it as the ***decorated game***. We interchangeably refer to traversals as being over $\mathcal{G}_{G,\mathcal{C}^{\perp}}$ and $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^{\perp}}$. A traversal of $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^{\perp}}$ containing only nodes annotated with W is referred to as a ***winning traversal***. In particular, the following lemma is useful to us and comes as a corollary to Lemmas 3 and 4 together with the fact that a partial strategy labelled everywhere with W can be extended to a winning strategy:

Lemma 5. *Let α be a node in $[\![G]\!]$ and q a state of the APT template $\mathfrak{B}$, and let t_α be a* winning *q-tipped traversal t_α in $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^{\perp}}$ whose final element is of the form (α^{Λ},p) (and* $\mathrm{sim}(p) = q$*). There is a minimal partial strategy T_α for Éloïse on $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^{\perp}}$* that can be extended to a winning strategy (accepting run-tree of $\mathcal{C}^{\perp}$ on $\lambda(G)$) *satisfying*

$$\{\pi^{\mathrm{NDup}}_{\mathrm{Gr}(G)}(\mathrm{trace}(r)) \ : \ r \textit{ is a path in } T_\alpha\} \ = \ \{\ulcorner \pi_{\mathrm{Gr}(G)}(s)\urcorner \ : \ s \textit{ initial segment of } t_\alpha\} \ .$$

In particular there is a maximal branch b in T_α such that

$$\pi^{\mathrm{NDup}}_{\mathrm{Gr}(G)}(\mathrm{trace}(b)) = \ulcorner \pi_{\mathrm{Gr}(G)}(t_\alpha)\urcorner$$

with the tip of b being the sole node to be given a label (n, p) where $\mathrm{sim}(p) = q$.

Note that α^{Λ} is well-defined as used above due to the second observation in Lemma 3, which ensures that during the initialization phase of a traversal of $\mathcal{G}_{G,\mathcal{C}^\perp}$ one can leave a terminal node in any direction and the only time at which an ϵ-transition may be made at a terminal node is to q-initialize.

We now make the following claim:

Lemma 6. *Let α be a node in $[\![G]\!]$ and q a state of the property APT template $\mathfrak{B}$. TFAE:*

1. *Property APT $\mathcal{B}_q \in \mathfrak{B}$ accepts $[\![G]\!]_\alpha$.*
2. *There exists a winning q-tipped traversal in $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$ whose final element is of the form (α^{Λ}, p) (and $\mathrm{sim}(p) = q$).*

The proof from 2 to 1 just consists of combining Lemmas 5 and 2. To go in the other direction, we use the second equivalence in Lemma 5 and then use Ong's construction of an accepting run tree of $\mathcal{C}^\perp$ from the accepting traversal tree of $\mathcal{B}^\perp$ ([14], Sec. 5). We appeal to observations concerning @ and variable nodes in $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$ made in Lemma 3 to ensure that Abelard owns the @ and variable elements of the q-tipped traversal.

4 Construction of an n-CPDA Recogniser

Let us formally consider what it means to have an automaton as a solution to the Global Model-Checking Problem for a tree generated by a higher-order recursion scheme.

Definition 8. *Let $\mathfrak{B}$ be a template for an APT with state space $Q_{\mathfrak{B}}$ and let G be a higher-order recursion scheme. Let $n \in \mathbb{N}$ be the maximal rank of any terminal occurring in G and let $\mathrm{Dir}(\Sigma) = \{1, \ldots, n\}$ be the corresponding set of directions (so that nodes in $[\![G]\!]$ are denoted by elements in $\mathrm{Dir}(\Sigma)^*$).*

Now let $\mathcal{A}$ be an automaton (of any type) that reads (finite) words over the alphabet $\mathrm{Dir}(\Sigma)$ with a finite state-set $Q_{\mathcal{A}}$ (and possibly additional memory of some kind such as a stack). We say that $\mathcal{A}$ is an ***automaton-solution to the Global Model Checking Problem*** *(GMCP) for (the tree generated by) G with respect to $\mathfrak{B}$ just in case it can be endowed with a map $\mathcal{Q} : Q_{\mathcal{A}} \longrightarrow Q_{\mathcal{B}} \cup \{\perp\}$ such that the following set equality holds for every state $q \in Q_{\mathfrak{B}}$:*

$$\{w \in \mathrm{Dir}(\Sigma)^* : \ \mathcal{B}_q \textit{ accepts } [\![G]\!]_w\} = \{w \in \mathrm{Dir}(\Sigma)^* : \exists s \in \mathrm{ctl}(w) \,.\, \mathcal{Q}(s) = q\}$$

where $\mathrm{ctl}(w)$ is the set of control *states of $\mathcal{A}$ that are reachable on reading word w.*[3]

In particular if we have an APT $\mathcal{B}_{q_0}$ (with initial state q_0) then we can represent the set of subtrees of $[\![G]\!]$ accepted by $\mathcal{B}_{q_0}$ with an automaton-solution $\mathcal{A}$ to the Global Model-

[3] The $\perp$ label of a state in $\mathcal{A}$ allows one to avoid a state in $\mathcal{A}$ being associated with any element of $\mathcal{B}$. That is, $\mathcal{Q}$ could be viewed as a *partial* function from $\mathcal{A}$ to $\mathcal{B}$.

Checking Problem which is given final states $\{s \in Q_{\mathcal{A}} : \mathcal{Q}(s) = q_0\}$ and the standard acceptance condition for finite strings.

An ***order-n pushdown automaton (n-PDA)*** is an automaton equipped with a stack of $(n-1)$-stacks where a 1-stack contains only atoms and a $(k+1)$-stack is a stack of k-stacks. For $m \geq 2$ an order-m *push* operation copies the top-most $(m-1)$-stack whilst an order-m *pop* operation discards it. The order-1 *push* and *pop* are the standard pushdown operations acting on the top-most 1-stack. We write top_1 to denote the top-most element of the top-most 1-stack. An ***order-n* collapsible *pushdown automaton* *(n-CPDA)*** [10] has an n-stack that allows a *pointer* from any atomic element (stack symbol) to a k-stack below it (where $1 \leq k < n$). It has a *collapse* operation that discards the stack's contents above the target of top_1's pointer.

We claim that an order-n collapsible pushdown automaton (n-CPDA) can be a solution to the GMCP for an order-n recursion scheme. Moreover we claim that it is possible to effectively construct the requisite n-CPDA from $\mathfrak{B}$ (or $\mathcal{B}_q$) and G. We adapt the automaton $\mathrm{CPDA}(G)$ introduced by Hague et al. [10] that is able to compute traversals of $\mathrm{Gr}(G)$ so that instead it computes *winning traversals of* $\mathcal{G}^{\mathcal{W}}_{G,\mathcal{C}^\perp}$. Its direction at terminal symbols is guided by reading a node α of $[\![G]\!]$ (which is just a word in $\mathrm{Dir}(\Sigma)^*$). By Lemma 6 this enables the automaton to fulfil the task demanded of it.

Theorem 1. *Let G be an order-n recursion scheme and $\mathfrak{B}$ a property APT template. We can construct an n-CPDA that is a solution to the associated Global Model Checking Problem in n-EXPTIME. The constructed automaton has n-exponential size.*

Note that deciding whether an n-CPDA accepts a given finite-word turns out to be $(n-1)$-EXPTIME complete in the size of the n-CPDA. We can establish this by first showing the emptiness problem to be $(n-1)$-EXPTIME complete.

In general an $(n-1)$-CPDA cannot provide a solution to GMCP for order-n schemes unless $(n-1)$-CPDA are equi-expressive with n-CPDA.

Lemma 7. *Let G be a (non-deterministic) order-n recursion scheme that generates a finite-word language $\mathcal{L}$ over an alphabet Σ. There exists a* deterministic *order-n recursion scheme G' generating a ranked and ordered tree together with a μ-calculus sentence ϕ such that the language $\mathcal{L}' := \{\alpha \in \mathrm{Dir}^* : [\![G]\!]_\alpha \vDash \phi\}$ can be viewed as being over an alphabet Σ' with $\Sigma \subseteq \Sigma'$ and*

$$\mathcal{L}' \upharpoonright_\Sigma := \{ w \in \Sigma^* : w \textit{ is the maximal } \Sigma\textit{-sub-sequence of an element of } \mathcal{L}' \} = \mathcal{L}$$

Proof. Let G be a *non-deterministic* order-n recursion scheme generating a finite word language $\mathcal{L}$ over a finite alphabet Σ. The elements of Σ can be viewed as terminals of arity 1 and we also have an *end-of-string marker* $e \notin \Sigma$ with arity 0. The rules of G will be of the form $F_i\zeta^i_1 \ldots \zeta^i_{m_i} \longrightarrow t^i_1 \mid \ldots \mid t^i_{k_i}$ for each non-terminal F_i where i ranges in $1 \leq i \leq N$ (say). Let K be the least integer with $k_i \leq K$ for all $1 \leq i \leq N$.

We form a deterministic recursion scheme G' that generates a single tree. The ranked alphabet Γ used by G' consists of two arity-0 terminals e and b together with a terminal h of arity $|\Sigma| + K$. Let $\sigma : \Sigma \longrightarrow \{ i : 1 \leq i \leq |\Sigma| \}$ be some bijection.

We give G' a terminal F'_i for every terminal F_i in G such that F'_i has the same type as F_i. We take the initial non-terminal of G' to be S'; we further provide a non-terminal C_c with type $o \to o$ for each $c \in \Sigma$. The rules of G' are as follows:

$$F'_i\, \zeta^i_1 \dots \zeta^i_{m_i} \longrightarrow h\ \underbrace{b \dots b}_{|\Sigma| \text{ times}}\ t^{i\,\star}_1 \dots t^{i\,\star}_{k_i}\ \underbrace{b \dots b}_{(K-k_i) \text{ times}}$$

$$C_c\, x \longrightarrow h\ \underbrace{b \dots b}_{\sigma(c)-1 \text{ times}}\ x\ \underbrace{b \dots b}_{|\Sigma|-\sigma(c) \text{ times}}\ \underbrace{b \dots b}_{K \text{ times}}$$

for $1 \le i \le N$ and $c \in \Sigma$, where $t^{i\star}_l$ $(1 \le l \le k_i)$ is formed from t^i_l by replacing each occurrence of a terminal $c \in \Sigma$ with a non-terminal C_j. Note that the end-of-string marker e is never replaced as our convention leaves it out of Σ.

Let α be a node of $[\![G]\!]$. Let us identify Σ with the first $|\Sigma|$ directions $\{1, \dots, |\Sigma|\}$ of h. By the construction of G' from G we have $[\![G]\!](\alpha) = e$ if and only if $\alpha \upharpoonright_\Sigma\ \in \mathcal{L}$. The μ-calculus formula ϕ asserting a node is labelled with e then gives the result. $\square$

The language $\mathcal{L}'$ in the above Lemma is the set that should be recognised by the solution to the GMCP for G' and ϕ. If an n-CPDA can recognise $\mathcal{L}'$, then there must exist an n-CPDA that can recognise $\mathcal{L}' \upharpoonright_\Sigma$. There exists a hierarchy theorem of Damm [6] for PDA and modulo the assumption of a similar theorem for CPDA we obtain the following:

Theorem 2. *Assuming that the CPDA generated word-languages form a strict hierarchy, there exists an order-n recursion scheme G and a μ-calculus sentence ϕ such that no m-CPDA with $m < n$ can be a solution to the corresponding GMCP.*

5 Winning Region of a Collapsible Pushdown Game

We characterise the *constructible* winning region of a collapsible pushdown parity game in terms of the sequences of stack operations that can generate the winning configurations. We refer to the automaton-generator of the underlying digraph as a ***collapsible pushdown system*** (CPDS) and its configuration graph as a ***CPDS graph***.

Some stacks cannot be constructed by operations on the empty-stack: $[\,[\,a\,]\,[\,\overset{\curvearrowleft}{a\,]\,[\,b}\,]\,]$.

The Unravelling of a CPDS Graph and Winning Condition APT. The *unravelling* of a CPDS graph $\mathcal{G}$ is a tree $\mathrm{unrav}(G)$ formed by labelling each node (q, s) (q a control state and s a stack) of the configuration graph with $(q, top_1\, s)$ and then unfolding from the initial configuration. We can view this tree as being ranked and ordered by giving a label (q, a) an arity equal to the size of the set $\{\, (q', \theta) : (q, a, q', \theta) \in \Delta \,\}$, where $\Delta \subseteq Q \times \Gamma \times Q \times Op_n$ is the transition relation and Op_n the set of order-n stack operations. We make the tree ordered by placing a linear order on the set.

For any CPDS parity game with underlying CPDS graph $\mathcal{G}$ the ownership $O(q)$ and priority $\Omega(q)$ of a configuration (q, s) are given entirely by q. We can thus [7] construct an APT $\mathcal{B}$ that, for a given node r in $\mathrm{unrav}(\mathcal{G})$ corresponding to a configuration (q, s) in $\mathcal{G}$, accepts $\mathrm{unrav}(\mathcal{G})_r$ if and only if Éloïse has a winning strategy from (q, s). Whenever $\mathcal{B}$ reads a node labelled (q, a), it transitions to a state with priority $\Omega(q)$ that is a $\bigvee$ state if $O(q)$ is Abelard and $\bigwedge$ otherwise. We call $\mathcal{B}$ ***the winning condition APT (WCAPT)***.

The Versatile CPDS Parity Game. Let us fix an n-CPDS parity game $\mathcal{A}$. We convert it to a game $\mathcal{A}^{\mathbf{0}}$, which by analogy with the work in previous sections is referred to as

the versatile CPDS parity game. The game $\mathcal{A}^{\mathbf{0}}$ extends $\mathcal{A}$ with a single control state $\mathbf{0}$. The priority and owner of $\mathbf{0}$ does not matter and so may be arbitrarily selected.

We make $\mathbf{0}$ the initial control state of $\mathcal{A}^{\mathbf{0}}$. Whilst in control state $\mathbf{0}$, Éloïse is allowed to perform arbitrary stack operations whilst remaining at $\mathbf{0}$. She may also opt at any point to transition from $\mathbf{0}$ into a control state q of $\mathcal{A}$ without performing any stack operation. After doing so, play proceeds as in $\mathcal{A}$. Consider the set:

$$S_{\mathbf{0}} := \{(\mathbf{0}, \theta) : \theta \text{ a stack operation}\} \cup \{(q, id) : q \text{ a control state of } \mathcal{A}\}$$

where id is the stack operation that leaves the stack unchanged. Let $\mathcal{G}^{\mathbf{0}}$ be the underlying CPDS-graph of $\mathcal{A}^{\mathbf{0}}$. The directions emanating from a node r in $\mathrm{unrav}(\mathcal{G}^{\mathbf{0}})$ having label $(\mathbf{0}, a)$, for any stack symbol a, are in $1-1$ correspondence with $S_{\mathbf{0}}$. We may thus label a direction of such a node r with θ if this direction corresponds to a transition $(\mathbf{0}, \theta)$ and q (q a control state of $\mathcal{A}$) if it corresponds to a transition (q, id).

Consider a finite path $p = p_0\ p_1\ \ldots\ p_m\ p'_m$ in $\mathrm{unrav}(\mathcal{G}^{\mathbf{0}})$, where p_0 is the root of the tree, with trace of the form $(\mathbf{0}, a_1)\ (\mathbf{0}, a_2)\ \ldots\ (\mathbf{0}, a_m)\ (q, a_m)$ such that q is a control state of $\mathcal{A}$. The node p'_m is represented as a string of directions, but this string can be represented by a string of the form $\theta_1\ \ldots\ \theta_m\ q$. The final element p'_m of p will correspond to a configuration (q, s) in $\mathcal{G}^{\mathbf{0}}$ where s is a stack produced from the empty stack by performing the composite operation $\theta_1;\ \ldots\ ; \theta_m$. Conversely, for any sequence of stack operations followed by a control state q of $\mathcal{A}$ there must exist a node in $\mathrm{unrav}(\mathcal{G}^{\mathbf{0}})$ represented by this sequence which corresponds to a configuration (q, s) with s formed by the sequence of stack operations starting at the empty stack.

Éloïse has a winning strategy from such a configuration (q, s) in $\mathcal{A}$ if and only if she has a winning strategy from (q, s) in $\mathcal{A}^{\mathbf{0}}$, since the games proceed identically from this configuration. Let us write $\mathcal{B}^{\mathbf{0}}$ for the WCAPT of $\mathcal{A}^{\mathbf{0}}$. Suppose further that s can be formed from the empty stack by a sequence $\theta_1\ \ldots\ \theta_m$ of stack operations. It follows that (q, s) is a winning configuration in $\mathcal{A}$ if and only if $\mathcal{B}^{\mathbf{0}}$ accepts the tree $\mathrm{unrav}(\mathcal{G}^{\mathbf{0}})_{\theta_1\ \ldots\ \theta_m\ q}$, viewing $\theta_1\ \ldots\ \theta_m\ q$ as a string of directions – i.e. a node.

The Constructible Winning Region of an n-CPDS Parity Game. It has been shown by Hague et al. [10] that the *unravelling* of a CPDS graph can be generated by a deterministic n-CPDA and consequently by a (deterministic) order-n recursion scheme. Let $G^{\mathbf{0}}$ be such a recursion scheme for our n-CPDS parity game $\mathcal{A}^{\mathbf{0}}$. Let us apply Theorem 1 to generate a solution $\mathcal{D}$ for the GMCP with $G^{\mathbf{0}}$ and the property expressed by $\mathcal{B}^{\mathbf{0}}$. We then restrict $\mathcal{D}$ to form an automaton $\mathcal{D}^-$ that only accepts words of the form $\theta_1\ \ldots\ \theta_m\ q$ that are also accepted by $\mathcal{D}$. The automaton $\mathcal{D}^-$ witnesses the following:

Theorem 3. *Let $\mathcal{A}$ be an n-CPDS parity game with stack operations Op_n and control states Q. We can construct in n-EXPTIME an n-CPDA that recognises a subset $\mathcal{L}$ of $(Op_n)^*Q$ such that Éloïse has a winning strategy from a configuration (q, s) with s constructible (via operations in Op_n) from the empty stack, if and only if for every operation sequence $\theta_1;\ \ldots\ ;\ \theta_m$ generating s from the empty stack, $\theta_1\ \ldots\ \theta_m\ q \in \mathcal{L}$.*

Given any configuration (q, s) with constructible stack s we can thus determine whether it is a winning configuration by picking *any* operation sequence $\theta_1\ \ldots\ \theta_m$ witnessing the constructibility of s and deciding whether $\theta_1\ \ldots\ \theta_m\ q$ is accepted by the automaton.

Further Directions. A pressing question is whether one can construct a more succinct and *deterministic* n-CPDA providing a solution to the GMCP for the trees in question.

Theorem 3 is weak as it stands. Carayol and Slaats [5] have shown that constructible n-PDS (non-collapsible) parity game winning regions are *'n-regular'* [3,9] and admit a *canonical representation*. An analogous result for CPDS games would be good.

Acknowledgements. We acknowledge the use of William Blum's `pstring.sty` and Frank Drewes' `graph.sty` packages, and thank Olivier Serre and Matthew Hague for useful discussions, and the anonymous referees for helpful and thorough comments.

References

1. Bradfield, J., Stirling, C.P.: Modal logics and mu-calculi: an introduction. In: Handbook of Process Algebra, pp. 293–332. Elsevier, North-Holland (2001)
2. Cachat, T.: Uniform solution of parity games on prefix-recognizable graphs. In: Proc. VISS. ENTCS, vol. 68. Elsevier, Amsterdam (2002)
3. Carayol, A.: Regular sets of higher-order pushdown stacks. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 168–179. Springer, Heidelberg (2005)
4. Carayol, A., Hague, M., Meyer, A., Ong, C.-H.L., Serre, O.: Winning regions of higher-order pushdown games. In: Proc. LICS, pp. 193–204. IEEE Computer Society, Los Alamitos (2008)
5. Carayol, A., Slaats, M.: Positional strategies for higher-order pushdown parity games. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 217–228. Springer, Heidelberg (2008)
6. Damm, W.: The IO- and OI -hierarchy. Theoretical Computer Science 20, 95–207 (1982)
7. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy. In: Proc. FOCS, pp. 368–377. IEEE computer society, Los Alamitos (1991)
8. Engelfriet, J.: Iterated pushdown automata and complexity classes. In: Proc. STOC, pp. 365–373. ACM, New York (1983)
9. Frantani, S.: Automates à piles de piles...de piles. PhD thesis (2005)
10. Hague, M., Murawski, A.S., Ong, C.-H.L., Serre, O.: Collapsible pushdown automata and recursion schemes. In: Proc. LICS. IEEE Computer Society, Los Alamitos (2008)
11. Hyland, M., Ong, C.-H.L.: On full abstraction for PCF: I, II and III. Information and computation 163(2), 285–408 (2000)
12. Jurdziński, M.: Small progress measures for solving parity games. In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, p. 290. Springer, Heidelberg (2000)
13. Knapik, T., Niwinski, D., Urzyczyn, P.: Higher-order pushdown trees are easy. In: Nielsen, M., Engberg, U. (eds.) FOSSACS 2002. LNCS, vol. 2303, pp. 205–222. Springer, Heidelberg (2002)
14. Ong, C.-H.L.: On model-checking trees generated by higher-order recursion schemes. In: IEEE Computer Society, IEEE Computer Society, Los Alamitos (2006); Journal version, `users.comlab.ox.ac.uk/luke.ong/publications/ntrees.pdf`
15. Piterman, N., Vardi, M.Y.: Global model-checking of infinite-state systems. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 387–400. Springer, Heidelberg (2004)
16. Serre, O.: Note on winning positions on pushdown games with ω-regular conditions. Information Processing Letters 85, 285–291 (2003)
17. Stirling, C.P.: Bisimulation, model checking and other games. In: Notes for the Mathfit instructional meeting on games and Computation (1997)