Model Checking and Functional Program Transformations

Axel Haddad

Model Checking and Functional Program Transformations

Axel Haddad[1]

1 **LIAFA (Université Paris Diderot & CNRS) LIGM (Université Paris Est & CNRS)**

Abstract

We study a model for recursive functional programs called, higher order recursion schemes (HORS). We give new proofs of two verification related problems: reflection and selection for HORS. The previous proofs are based on the equivalence between HORS and collapsible pushdown automata and they looses the structure of the initial program. The constructions presented here are based on shape preserving transformations, and can be applied on actual programs without losing the structure of the program.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Higher-order recursion schemes, Model checking, Tree automata

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Recursion schemes were introduced in the early 70s as a model of computation, describing the syntactical part of a functional program [19, 6, 7, 8]. Originally they only handled order-0 (constants) or order-1 (functions on constants) expressions, but not higher-order types. Higher-order versions of recursion schemes were later introduced to deal with functions taking functions as arguments [13, 9, 10].

Recently, the focus came back on higher-order recursion schemes when considering them as generators of possibly infinite trees [14, 20, 12]. Indeed, roughly speaking a recursion scheme is a deterministic rewriting system on typed terms that generates ad infinitum a unique infinite tree. As the trees they generate are very general and as they can capture the computation tree of (higher-order) functional programs, studying their logical properties leads to very natural and challenging problems.

The most popular one is *(local) model-checking*: for a given recursion scheme and a formula *e.g.* from monadic second order logic (MSO) or μ-calculus, decide whether the tree generated by the scheme satisfies the formula. Following partial decidability results for the subclass of *safe* recursion schemes [14, 5], Ong proved, using the notion of traversals, the decidability of MSO model-checking for the whole class of trees generated by recursion schemes [20]. Since then, other proofs of this result have been obtained using different approaches: Hague, Murawski, Ong and Serre established the equivalence of schemes and higher-order collapsible pushdown automata (CPDA), and then showed the MSO decidability by reduction to parity games on collapsible pushdown automata [12]; following ideas from [1], Kobayashi and Ong [17] developed the type system of [15] to obtain a new proof. Finally, Salvati and Walukiewicz used Krivine machines to establish the MSO decidability of λ-Y-calculus, which is a typed lambda calculus with recursion, equivalent to higher order recursion schemes [22].

Another important problem is *global model-checking*: for a given recursion scheme and a formula, provide a finite representation of the set of nodes in the tree generated by the

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 1–35

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

scheme where the formula holds. Broadbent, Carayol, Ong and Serre answered this question using a so-called endogenous approach to represent the set of nodes: they showed how to construct another scheme generating the same tree as the original scheme except that now those nodes where the formula holds are marked [3]. They refer to this property as the *logical reflection*. The technique they used relies on the equivalence between schemes and CPDA.

Going further with the idea of marking a set of nodes, Carayol and Serre considered in [4] the following problem called *logical selection* and generalising global model-checking: for a given recursion scheme and an MSO formula $\varphi[X]$ with one free set variable, provide (if it exists) a finite representation of a set S of nodes in the tree generated by the scheme such that $\varphi[S]$ holds. They show that one can construct another scheme generating the same tree as the original scheme except that now those nodes are marked and describe a set S with the previous property. The technique they used relies on the equivalence between schemes and CPDA.

One interest for logical reflection and logical selection is that they can be used to modify a scheme in a useful way. Indeed, assume some (syntax tree of a) program is described by a scheme and that it contains bad behaviours: using logical reflection, one can get a new scheme where those bad behaviours are marked and this latter scheme can then easily be modified to remove these behaviours. Using logical selection, one can *e.g.* select branches in the syntax tree so that a given property is satisfied in the resulting subtree. A drawback of the approach in [3, 4] that goes back and forth between schemes and CPDA is that the scheme that is finally obtained is structurally very far from the original one (the names of the non terminals as well as the shape of the rewriting rules have been lost): hence this is a serious problem if one is interested in doing automated correction of programs or even synthesis.

Our main contribution in this paper is to provide proofs of both logical reflection and selection without appealing to CPDA as we only reason on schemes. Our constructions avoid the loss of the structure, *i.e.* the solution scheme is obtained only by duplicating and annotating some parts of the initial scheme and the transformation is easily reversible. Our constructions are based on the type system and the game used by Kobayashi and Ong in their proof of the model-checking decidability [17]. There is no known correspondence between these proofs and the former ones. In order to prove the logical reflection, we introduce the notion of morphism which is a slight generalisation of denotational semantics, and we give a denotational semantics for MSO. In order to prove the logical selection, we embed carefully a winning strategy of the game into the scheme.

In Section 2 we give the basic definitions and in Section 3 we introduce the two problems we are looking at in the paper. In Section 4 we introduce morphisms, explain how to "embed" a morphism into a scheme, and give some possible applications. In Section 5, we show how to use morphisms to obtain a new proof of logical reflection. In Section 6, we deal with logical selection. In Section 7, we present a simple example that describes how we can use a morphism describing a property to transform a scheme.

2 Preliminaries

In this section we give the definition of a higher order recursion scheme, which is a deterministic rewriting system that produces an infinite tree. It handles terms of typed symbols, i.e. each symbol is a constant or a function that can have functions as arguments. Terms may represent expressions of higher order programming language, for example the term `Apply Copy File` means "apply the function `Copy` to the file". In this example an intuitive rewrite rule for

the function `Apply` would be the term where the first argument is applied on the second argument, written: `Apply` $\varphi\ x \to \varphi\ x$.

Types. ***Types*** are defined by the grammar $\tau ::= o \mid \tau \to \tau$ and o is called the ***ground type***. Considering that $\to$ is associative to the right (i.e. $\tau_1 \to (\tau_2 \to \tau_3)$ can be written $\tau_1 \to \tau_2 \to \tau_3$), any type τ can be written uniquely as $\tau_1 \to \ldots \to \tau_k \to o$. The integer k is called the ***arity*** of τ. We define the ***order of a type*** by $\text{order}(o) = 0$ and $\text{order}(\tau_1 \to \tau_2) = \max(\text{order}(\tau_1) + 1, \text{order}(\tau_2))$.We let $\tau^\ell \to \tau'$ be a shorthand for $\underbrace{\tau \to \ldots \to \tau}_{\ell \text{ times}} \to \tau'$.

Terms. Let Γ be a finite set of typed symbols, and Let Γ^τ denote the set of symbols of type τ. For all type τ, we define the set of ***terms*** of type τ, $\mathcal{T}^\tau(\Gamma)$ as the smallest set containing the symbols of type τ and the application of a term of type τ' to a term of type $\tau \to \tau'$ for all τ'; formally: $\Gamma^\tau \subseteq \mathcal{T}^\tau(\Gamma)$ and $\bigcup_{\tau'}\{t\ s \mid t \in \mathcal{T}^{\tau' \to \tau}(\Gamma), s \in \mathcal{T}^{\tau'}(\Gamma)\} \subseteq \mathcal{T}^\tau(\Gamma)$. We shall write $\mathcal{T}(\Gamma)$ for the set of terms of any type, and $t : \tau$ if t has type τ. The arity of a term t, $arity(t)$, is the arity of its type. Remark that any term t can be uniquely written as $t = \alpha\ t_1 \ldots t_k$ with $\alpha \in \Gamma$ and $0 \leq k \leq \text{arity}(\alpha)$. We say that α is the ***head*** of the term t.

A set of symbols of order at most 1 (i.e. each symbols has type o or $o \to o \to \ldots \to o$) is called a ***signature***. In the following, we use the letters t, r, s to denote terms, and given a tuple of term $\vec{t} = (t_1, \ldots, t_k)$ we may use the shorthand $s\ \vec{t}$ to denote $s\ t_1 \ldots t_k$.

Contexts. Let $t : \tau$, $t' : \tau'$ be two terms, $x : \tau'$ be a symbol of type τ', then we write $t_{[x \mapsto t']} : \tau$ the term obtained by substituting all occurrences of x by t' in the term t. A ***τ-context*** is a term $C[\bullet^\tau] \in \mathcal{T}(\Gamma \uplus \{\bullet^\tau : \tau\})$ containing exactly one occurrence of $\bullet^\tau$; it can be seen as an application turning a term into another, such that for all $t : \tau$, $C[t] = C[\bullet^\tau]_{[\bullet^\tau \mapsto t]}$. In general we will only talk about ground type context where $\tau = o$ and we will omit to specify the type when it is clear.

Rewrite Rules. Given two disjoint sets of symbols $\Gamma, \mathcal{V}$ where $\mathcal{V}$ is called a set of ***variables***, we define a (fully applied) ***rewrite rule*** on Γ and $\mathcal{V}$ as a p air of terms of $\mathcal{T}(\Gamma \uplus \mathcal{V})$ of type o written $F\ x_1 \ldots x_k \to e$ with $F \in \Gamma$, $x_1, \ldots, x_k \in \mathcal{V}$ such that for all $i \neq j$ $x_i \neq x_j$, and $e \in \mathcal{T}(\Gamma \uplus \{x_1, \ldots, x_k\})$. Given a set of rewrite rules $\mathcal{R}$, we define the ***rewriting relation*** $\to \in \mathcal{T}(\Gamma)^2$ as $t \to t'$ iff there exists a context $C[\bullet]$, a rewrite rule $F\ x_1 \ldots x_k \to e$, and a term $F\ t_1\ \ldots\ t_k : o$ such that $t = C[F\ t_1 \ldots t_k]$ and $t' = C[e_{[x_1 \mapsto t_1] \ldots [x_k \mapsto t_k]}]$. We call $F\ t_1\ \ldots\ t_k : o$ a ***redex***. We define $\to^*$ as the reflexive and transitive closure of $\to$. Finally we say that a set of rewrite rules is ***deterministic*** if for all $F \in \Gamma$ there exists at most one rule of the form $F\ x_1 \ldots x_k \to e$.

Trees. Let Σ be a finite signature, m be the maximum arity in Σ and $\bot : o$ be a fresh symbol. A (ranked) ***tree*** t over $\Sigma \uplus \bot$ is a mapping $t : dom^t \to \Sigma \uplus \bot$, where dom^t is a prefix-closed subset of $\{1, \ldots, m\}^*$ such that if $u \in dom^t$ and $t(u) = a$ then $\{j \mid u \cdot j \in dom^t\} = \{1, \ldots, arity(a)\}$. Given a node $u \in dom^t$, we define the subtree of t rooted at u as the tree t_u such that $dom^{t_u} = \{j \mid u \cdot j \in dom^t\}$ and for all $v \in dom^{t_u}$, $t_u(v) = t(u \cdot v)$. Given $a : o^k \to o$ and some trees $t_1, \ldots, t_k$ we use the notation $a\ t_1 \ldots t_k$ to denote the tree t' whose domain is $dom^{t'} = \bigcup_i i \cdot dom^{t_i}$, $t'(\varepsilon) = a$ and $t'(i \cdot u) = t_i(u)$. Note that there is a direct bijection between ground terms of $\mathcal{T}^o(\Sigma \uplus \bot)$ and finite trees. Hence we will freely allow ourselves to treat ground terms over $\Sigma \uplus \bot$ as trees. We define the partial order $\sqsubseteq$ over trees as $t \sqsubseteq t'$ if $dom^t \subseteq dom^{t'}$ and for all $u \in dom^t$, $t(u) = t'(u)$ or $t(u) = \bot$. Given a (possibly infinite) sequence of trees $t_0, t_1, t_2, \ldots$ such that $t_i \sqsubseteq t_{i+1}$ for all i, the set of all t_i has a supremum that is called the ***limit tree*** of the sequence.

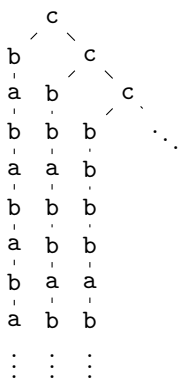
Higher Order Recursion Schemes. A ***higher order recursion scheme (HORS)*** $\mathcal{G} = \langle \mathcal{V}, \Sigma, \mathcal{N}, \mathcal{R}, S \rangle$ is a tuple such that: $\mathcal{V}$ is a finite set of typed symbols called ***variables***; Σ is a finite signature, called the ***set of terminals***; $\mathcal{N}$ is a finite set of typed symbols

called ***non-terminals***; $\mathcal{R}$ is a deterministic set of rewrite rules on $\Sigma \uplus \mathcal{N}$ and $\mathcal{V}$, such that there is exactly one rule per non terminal and no rule for terminals; $S \in \mathcal{N}$ is the ***initial non-terminal***.

We define inductively the ***$\perp$-transformation*** $(\cdot)^\perp : \mathcal{T}^o(\mathcal{N} \uplus \Sigma) \to \mathcal{T}^o(\Sigma \uplus \{\perp : o\})$ turning a ground term into a finite tree as: $(F\ t_1\ \dots\ t_k)^\perp = \perp$ for all $F \in \mathcal{N}$ and $(a\ t_1\ \dots\ t_k)^\perp = a\ t_1^\perp \dots t_k^\perp$ for all $a \in \Sigma$. We define a ***derivation***, as a possibly infinite sequence of terms linked by the rewrite relation, and we will mainly look at derivations where the first term of the sequence is equal to the initial non terminal. Let $t_0 = S \to t_1 \to t_2 \to \dots$ be a derivation, then one can check that $(t_0)^\perp \sqsubseteq (t_1)^\perp \sqsubseteq (t_2)^\perp \sqsubseteq \dots$, hence it admits a limit. One can prove that the set of all such limit trees has a greatest element that we denote $\|\mathcal{G}\|$ and refer to as the ***value tree*** of $\mathcal{G}$. Note that $\|\mathcal{G}\|$ is the supremum of $\{t^\perp \mid S \to^* t\}$. Given a term $t : o$, we denote by $\mathcal{G}_t$ the scheme obtained by transforming G such that it starts derivations with the term t, formally, $\mathcal{G}_t = \langle \mathcal{V}, \Sigma, \mathcal{N} \uplus \{S'\}, \mathcal{R} \uplus \{S' \to t\}, S' \rangle$. One can prove that if $t \to t'$ then $\|\mathcal{G}_t\| = \|\mathcal{G}_{t'}\|$. We call $\|\mathcal{G}_t\|$ the value tree of t in $\mathcal{G}$.

Parallel derivation. Intuitively, the ***parallel rewriting*** from a term t is obtained by rewriting all the redexes in t simultaneously. Formally, given a term $t = \alpha\ t_1 \dots t_k$ with $\alpha \in \Sigma \uplus \mathcal{N}$, we define inductively the parallel rewriting $t^\star$ of t as if $\alpha \in \Sigma$ or $k < \text{arity}(\alpha)$, then $t^\star = \alpha\ t_1^\star \dots t_k^\star$, if $\alpha \in \mathcal{N}$ and $k = \text{arity}(\alpha)$, let $\alpha\ x_1 \dots x_k \to e$ be the rewrite rule associated to α, we have $t^\star = e_{[\forall i\ x_i \mapsto t_i^\star]}$. Given t, t', we write $t \Rightarrow t'$ the relation "$t' = t^\star$". Notice that if $t \Rightarrow t'$ and $t' \to^* t''$ then $t \to^* t''$ (in particular $t \to^* t'$). Furthermore the derivation $S \Rightarrow t_1 \Rightarrow t_2 \Rightarrow \dots$ leads to $\|G\|$.

▶ **Example 1.** Let $\mathcal{G} = \langle \Sigma, \mathcal{V}, \mathcal{N}, S, \mathcal{R} \rangle$ with $\Sigma = \{\mathtt{a}, \mathtt{b} : o \to o, c : o \to o \to o\}$, $\mathcal{V} = \{x : o, \varphi : o \to o\}$, $\mathcal{N} = \{S : o, I, F : (o \to o) \to o, D, A : (o \to o) \to o \to o\}$ and $\mathcal{R} = \{S \to F\ \mathtt{b},\ \ D\ \varphi\ x \to \varphi\ (\varphi\ x),\ \ A\ \varphi\ x \to \varphi\ (\mathtt{a}\ x),\ \ I\ \varphi \to \varphi\ (I\ \varphi),\ \ F\ \varphi\ \to\ \mathtt{c}\ (I\ (A\ \varphi))\ (F\ (D\ \varphi))\}$.
Remark the rewrite rule associated to D: it means that for any function t, $D\ t$ is simply the function obtained by composing t with itself. The rule associated to I is also interesting: for any function t, $I\ t$ leads to the infinite iteration of t. For example the term $I\ \mathtt{a}$ can be derived to obtain $\mathtt{a}\ (\mathtt{a}\ (\mathtt{a}\ (\ \dots\ )))$. The tree $\|\mathcal{G}\|$ is depicted on the left, its branches are labelled by $\mathtt{c}^\omega$ or $\mathtt{c}^n \cdot (\mathtt{b}^{2^{(n-1)}} \cdot \mathtt{a})^\omega$ for all $n \geq 1$.

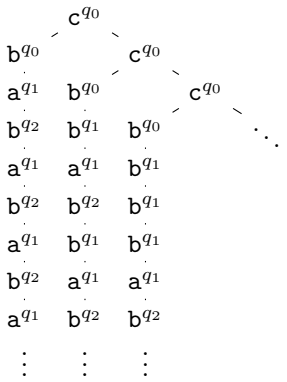


Parity tree automaton. A ***non-deterministic max parity automaton*** (we will just refer to them as automata in the following) is a tuple $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, \Omega \rangle$ with Σ a finite signature, Q a finite set called the ***set of states***, $\delta \subseteq \{q \xrightarrow{a} (q_1, \dots, q_{\text{arity}(a)}) \mid a \in \Sigma, q, q_1, \dots, q_{\text{arity}(a)} \in Q\}$ is called the ***transition relation***, $q_0 \in Q$ the ***initial state***, $\Omega : Q \to \{1, \dots, m_{\max}\}$ for some $m_{\max} \in \mathbb{N}$ the ***colouring function***.

Given an infinite tree t on Σ we define a ***run*** r of $\mathcal{A}$ on t as a tree on $\Sigma \times Q = \{a^q : o^k \to o \mid a : o^k \to o, q \in Q\}$ such that $dom^r = dom^t$, for all $u \in dom^t$, if $r(u) = a^q$ then $t(u) = a$ and there exists $q \xrightarrow{a} q_1, \dots, q_k \in \delta$ such that for all i, $r(u \cdot i) = t(u \cdot i)^{q_i}$, and $r(\varepsilon) = t(\varepsilon)^{q_0}$.

We say that the automaton $\mathcal{A}$ accepts the tree t, written $t \models \mathcal{A}$, if there exists a run r on t such that for every infinite branch $b = (a_0, q_0) \cdot j_0 \cdot (a_1, q_1) \cdot j_1 \cdot \dots$ in r, the greatest colour seen infinitely often in $\Omega(q_0), \Omega(q_1), \Omega(q_2), \dots$ is even.
We define $\mathcal{A}_q$ as the automaton obtained by changing the initial state to q: $\mathcal{A}_q = \langle \Sigma, Q, \delta, q, \Omega \rangle$, and we say that $\mathcal{A}$ accepts the tree t from state q, if $t \models \mathcal{A}_q$.

▶ **Example 2.** Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \Omega \rangle$ be an automaton with $\Sigma = \{\mathtt{a}, \mathtt{b} : o \to o, c : o \to o \to o\}$, $Q = \{q_0, q_1, q_2\}$, $\Omega = \{q_0 \mapsto 0, q_1 \mapsto 1, q_2 \mapsto 2\}$ and $\delta = \{q_0 \xrightarrow{\mathtt{c}} (q_0, q_0), q_0 \xrightarrow{\mathtt{b}} q_1, q_1 \xrightarrow{\mathtt{a}} q_2, q_1 \xrightarrow{\mathtt{b}} q_1, q_2 \xrightarrow{\mathtt{a}} q_2, q_2 \xrightarrow{\mathtt{b}} q_1\}$.
The automaton $\mathcal{A}$ accepts the trees whose branches are labelled by c^ω or by $c^* \cdot b \cdot (b^* \cdot \mathtt{a})^\omega$. An accepting run of $\mathcal{A}$ on the tree $\|\mathcal{G}\|$ of Example 1 is depicted on the right.



3 Logical Reflection and Logical Selection

In this section we formalise the notion of annotated tree and we define the problems of logical reflection and logical selection we are interested in.

Given a signature Σ, a set X and a tree t on the signature Σ, we define the signature $\Sigma \times X = \{(a, x) : o^k \to o \mid a : o^k \to o \in \Sigma, x \in X\}$ and we say that the tree t' on the signature $\Sigma \times X$ is an ***X*-annotation** of t, if $dom^t = dom^{t'}$ and if for all $u \in dom^t$, $t'(u) = (t(u), x)$ for some $x \in X$. For example a run of an automaton over a tree t is a Q-annotation of t, with Q being the set of states of the automaton. Furthermore, for any tree t' on the signature $\Sigma \times X$, we define $Unlab(t')$ as the tree on Σ obtained by turning all nodes (a, x) of t' into a *i.e.* the tree obtained by removing the annotated part of the tree.

Given a set of nodes S in a tree t, we define an ***S*-marking of** t as a $\{\mathtt{0}, \mathtt{1}\}$-annotation t' of t such that for all node u, $u \in S$ if and only if $t'(u) = (t(u), \mathtt{1})$. Given a μ-calculus formula φ, we say that t' is a ***φ*-reflection** of t if it is a marking of the set of nodes u such that the subtree of t rooted at u satisfies the formula φ. Given an MSO formula $\varphi[x]$ with a first order free variable, we say that t' is a ***φ*[*x*]-reflection** of t if it is a marking of the set of nodes u satisfying $\varphi[u]$. Given an MSO formula $\varphi[X]$ with a second order free variable, we say that t' is a ***φ*[*X*]-selection** of t if it is a marking of a set of nodes S satisfying $\varphi[S]$.

Finally, we define the ***μ*-calculus reflection**, MSO ***reflection***, and MSO ***selection*** as follow. Given a class R of generators of trees (*i.e.* a set of finitely described elements such that to each $g \in R$ we associate a unique tree $\|g\|$), we say that R is (effectively) ***reflective with respect to the μ-calculus*** (*resp.* MSO) if for any μ-calculus formula φ (*resp.* any MSO formula $\varphi[x]$) and any tree generator $g \in R$, one can construct another generator g' such that the $\|g'\|$ is a φ-reflection (resp. $\varphi[x]$-reflection) of $\|g\|$. We say that R is (effectively) ***selective with respect to*** MSO if for any MSO formula $\varphi[X]$ and any generator $g \in R$ such that there exists a subset S of nodes of $\|g\|$ satisfying $\varphi[S]$, one can construct another generator g' such that $\|\mathcal{G}'\|$ is a $\varphi[X]$-selection.

The main contribution of this paper is to give new proofs of the fact that schemes have these properties. In order to do so we use the equivalence results between logic and automata, and we rather prove automata reflexivity and automata selectivity defined as follows.

Given a tree t on the signature Σ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \Omega \rangle$, we define an ***A*-reflection** of $\mathcal{A}$ on t as a 2^Q-annotation t' of t, such that for all node u, $t'(u) = (t(u), Q')$ with $Q' \subseteq Q$ being the set of states q such that $\mathcal{A}_q$ accepts the subtree of t rooted on u; we define an ***A*-selection** of $\mathcal{A}$ on t as an accepting run of $\mathcal{A}$ on t. We say that a class R of generators of trees is ***reflective with respect to automata*** if for any automaton $\mathcal{A}$ and any generator $g \in R$, one can construct another generator g' such that $\|g'\|$ is an $\mathcal{A}$-reflection

of $\|g\|$. We say that R is ***selective with respect to automata*** if for any automaton $\mathcal{A}$ and any generator $g \in R$ such that $\|g\| \models \mathcal{A}$, one can construct another generator g' such that $\|g'\|$ is an $\mathcal{A}$-selection of $\|g\|$.

From the equivalence between logics and automata [21] we have that schemes are reflective with respect to the μ-calculus iff they are reflective with respect to automata, and they are selective with respect to MSO iff they are reflective with respect to automata. Furthermore, it is shown in [3] that μ-calculus reflection for schemes implies MSO reflection.

4 Morphisms

4.0.1 Definitions.

In the following we fix a scheme $\mathcal{G} = \langle \Sigma, \mathcal{N}, \mathcal{V}, S, \mathcal{R} \rangle$. We define a ***typed domain*** $\mathcal{D}$ as a set such that each element is typed, and to each element $d : \tau_1 \to \tau_2 \in \mathcal{D}$ is associated a *partial* mapping f_d from $\mathcal{D}^{\tau_1}$ to $\mathcal{D}^{\tau_2}$, where $\mathcal{D}^\tau = \{d \in \mathcal{D} \mid d : \tau\}$. We write $d\ d'$ the element $f_d(d')$.

We define a ***morphism*** $[\![\cdot]\!] : \mathcal{T}(\Sigma \uplus \mathcal{N}) \to \mathcal{D}$ from terms on $\Sigma \uplus \mathcal{N}$ to the domain $\mathcal{D}$ as a mapping such that (1) if $t : \tau$ then $[\![t]\!] : \tau$, (2) if $t_0 : \tau_1 \to \tau_2$ and $t_1 : \tau_1$, then $[\![t]\!]\ [\![t']\!]$ is defined and equal to $[\![t\ t']\!]$. In the following, we refer to $[\![t]\!]$ as the $\mathcal{D}$-value of the term t.

▶ **Example 3.** Let $\mathcal{D} = \bigcup_\tau \{\bot^\tau : \tau, \top^\tau : \tau\}$ such that for all τ_1, τ_2: $\top^{\tau_1 \to \tau_2}\ \top^{\tau_1} = \top^{\tau_2}$, $\top^{\tau_1 \to \tau_2}\ \bot^{\tau_1} = \top^{\tau_2}$, $\bot^{\tau_1 \to \tau_2}\ \top^{\tau_1} = \top^{\tau_2}$, $\bot^{\tau_1 \to \tau_2}\ \bot^{\tau_1} = \top^{\tau_2}$ if $\tau_2 = o$, $\bot^{\tau_2}$ otherwise. For $t : \tau$, we define $[\![t]\!]$ as $[\![t]\!] = \top^\tau$ if t contains a ground subterm and $[\![t]\!] = \bot^\tau$ otherwise. Then $[\![\cdot]\!]$ is a morphism since $t_1\ t_2$ contains a ground subterm iff t_1 contains a ground subterm, or t_2 contains a ground subterm, or $t_1\ t_2$ is ground.

Note that a morphism is entirely defined by its value on $\Sigma \uplus \mathcal{N}$, i.e. from those values one can compute $[\![t]\!]$ for any term t. Also remark that given a context $C[\bullet]$ and two terms t and t', if $[\![t]\!] = [\![t']\!]$ then $[\![C[t]]\!] = [\![C[t']]\!]$.

We say that a morphism $[\![\cdot]\!]$ is ***stable by rewriting*** if for $t, t' \in \mathcal{T}$ such that $t \to t'$, $[\![t]\!] = [\![t']\!]$. Finally, given a set of terms $\mathcal{T}' \subseteq \mathcal{T}(\Sigma \uplus \mathcal{N})$, we say that the morphism $[\![\cdot]\!]$ ***recognises*** $\mathcal{T}'$ if there exists a subset $\mathcal{D}'$ of $\mathcal{D}$ such that $t \in \mathcal{T}'$ if and only if $[\![t]\!] \in \mathcal{D}'$. In Example 3, the morphism recognises the set of terms containing a ground term as a subterm.

4.0.2 Embedding a Morphism Into a Scheme.

We fix a scheme $\mathcal{G} = \langle \mathcal{V}, \Sigma, \mathcal{N}, \mathcal{R}, S \rangle$ and a morphism $[\![\cdot]\!] : \mathcal{T}(\Sigma \uplus \mathcal{N}) \to \mathcal{D}$ on $\mathcal{G}$, stable by rewriting, such that for all type τ, $\mathcal{D}^\tau$ is finite. In Appendix A, we transform $\mathcal{G}$ into $\mathcal{G}' = \langle \mathcal{V}', \Sigma', \mathcal{N}', \mathcal{R}', S \rangle$ which, while it is producing a derivation, evaluates $[\![t']\!]$ for any subterm t' of the current term and annotates the term with all these $\mathcal{D}$-values. The new symbols of $\mathcal{G}'$ are symbols of $\mathcal{G}$ annotated with elements of the domain $\mathcal{D}$, and we define a transformation $(\cdot)^+$ from terms of $\mathcal{G}$ to terms of $\mathcal{G}'$ such that the transformation t^+ of t will be annotated with the $\mathcal{D}$-values of the subterms of t. The tree generated by $\mathcal{G}'$ will be annotated and when one remove these annotations, one retreives back the tree generated by $\mathcal{G}$. More precisely we show the following.

▶ **Theorem 4** (Embedding a Morphism). *Given two terms $t, t' : o \in \mathcal{T}(\Sigma \uplus \mathcal{N})$, if $t \Rightarrow_{\mathcal{G}} t'$, then $t^+ \Rightarrow_{\mathcal{G}'} t'^+$. In particular $Unlab(\|\mathcal{G}'_{t^+}\|) = \|\mathcal{G}_t\|$ (where $Unlab$ is the function that removes the annotations).*

▶ Remark. This transformation keeps the structure of the original scheme *i.e.* the new symbols are simple labelings of the original ones, new rewrite rules are obtained by duplicating

some subterms and labeling the symbols, a very simple transformation allows to get back to the original scheme, and there is a direct correspondence between derivations of the two schemes.

4.0.3 Applications.

Embedding properties of subterms during a derivation, or properties of subtrees of the value tree, can be useful for program analysis: instead of saying "There will be a forbidden behaviour in the program" reflection allows to say during the execution of a program "Here, the forbidden behaviour will appear in this subexpression, but the rest of the program is valid". Furthermore, once a property is embeded into a scheme, one can add some new rewrite rules that deal explicitly with wether the property is valid or not. For example one could replace all forbidden subtrees of the value tree by a special symbol FORBIDDEN.

Some morphisms generated by type systems are used in [15] to model check a subclass of trivial acceptance condition automata (i.e. automata where there is no colouring function, and the acceptance of a tree simply asks if there exists a run of the automaton on the tree). Then the construction allows to reflect the accepting states of the automaton (as defined in Section 3). This result has been improved in [23] to deal with the whole class of trivial acceptance condition automata. In Section 5 we extend this result to show that for any parity tree automaton one can create a morphism that reflects the acceptance of the automaton on the value tree.

One can create a model (for example in [2]) to decide whether or not a ground term t would be productive or not (i.e. $\|t\| \neq \bot$). Answering this question is very useful since deriving a non productive term is somehow useless. Reflecting the productivity of terms into a scheme $\|\mathcal{G}\|$ allows to create a scheme $\mathcal{G}'$ on the signature $\Sigma \uplus \{\bot\}$ such that $\|\mathcal{G}'\| = \|\mathcal{G}\|$, but such that no derivation will create some non productive terms. In [11] we developed this idea to compare evaluation policies.

5 Logical Reflection

In the following we present a morphism based on [17] that recognises the acceptance of a parity tree automaton. Using the construction introduced in Section 4.0.2, one can construct a scheme that reflects the accepting states of the automaton, which is equivalent to reflect the subtrees accepted by a formula of the μ-calculus. In [3], μ-calculus reflection (and MSO reflection) on schemes is already proven, but this construction uses the equivalence between schemes and collapsible pushdown automata, and the successive transformations (scheme $\to$ pushdown automaton $\to$ reflective pushdown automaton $\to$ reflective scheme) lose the structure of the scheme. In our construction, the structure of the scheme is the preserved, in the sense of Remark 4.0.2.

Since our proof of the logical reflection, as long as the one of logical selection in Section 6, is build on top of the MSO model checking proof of Kobayashi and Ong [17], we first recall their construction and then we explain how to use this result to obtain the logical reflection.

5.0.4 Kobayashi-Ong Result

We fix a non deterministic parity tree automaton $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, \Omega \rangle$ and a scheme $\mathcal{G} = \langle \Sigma, \mathcal{N}, \mathcal{V}, S, \mathcal{R} \rangle$. We let $\text{arity}_{\max}$, $\text{order}_{\max}$, and $m_{\max}$ be the maximum arity in $\Sigma \uplus \mathcal{N}$, order in $\Sigma \uplus \mathcal{N}$, colour in $\Omega(Q)$. The idea of the result is to define a type system, and to use this

type system in the construction of a two player parity game, such that Eve wins the game if and only if the automaton accepts the value tree of the scheme.

The type system. Kobayashi and Ong introduced some ***judgement rules*** that allow to type a term by an element of the typed set Map, called the set of ***mappings***. Mappings of type o are the states Q, and mappings of type $\tau \to \tau'$ are of the form $(\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \to \theta$ with for all i θ_i is a mapping of type τ, m_i is a color, and θ is a mapping of type τ'. The judgements are of the form $\Gamma \vdash t \triangleright \theta$ meaning under the environment Γ, one can judge t with the mapping θ. The environment Γ associates some mapping and colors to non terminal and variables, and give some restriction on the judgements one can make. Terminals are judged according to the transition of the automaton, *i.e.* $a : o^k \to o \in \Sigma$ may be judge as $\emptyset \vdash a \triangleright (q_1, m_1) \to \ldots \to (q_k, m_k) \to q$ with for all i, $m_i = \max(\Omega(q_i), \Omega(q))$ and $q \xrightarrow{a} q_1, \ldots, q_k \in \delta$. This type system keeps track of the colours in order to know exactly what colour has been seen along the term. It is given formally in Appendix B.

The game. Now we define a game $\mathbb{G}_{\mathcal{A}}$ in which Eve's states will be triples made of a non terminal, mapping and a colour, and Adam's states will be environments. Eve chooses an environment that can judge the rewrite rule of the current nonterminal with the current atomic mapping, while Adam picks one binding in the current environment.

Intuitively, Eve tries to show a well-typing of the terms with respect to the rewrite rules, that would induce a well-coloured run of the automaton, and Adam tries to show that she is wrong. From the state (F, θ, m), Eve has to find an environment Γ such that she can prove $\Gamma \vdash r_F : \theta$, then Adam picks a F and θ' in Γ and asks Eve to prove that θ' is chosen correctly according to the rewrite rule of F. If at some point of a play, Eve cannot find a correct environment, she loses the play; if she can choose the empty environment, Adam would have nothing to choose then she wins the play; if the play is infinite, Eve wins if and only if the greatest colour seen infinitely often is even. The game is also given formally in Appendix B.

▶ **Theorem 5** (Kobayashi, Ong 09). *The tree $\|\mathcal{G}\|$ is accepted by $\mathcal{A}$ from the state q if and only if Eve has a winning strategy from the vertex $(S, q, \Omega(q))$ in the game $\mathbb{G}_{\mathcal{A}}$.*

5.0.5 A Morphism for Automata Reflection.

From the winning strategy of Eve, we define a morphism $[\![\cdot]\!] : \mathcal{T}(\Sigma \uplus \mathcal{N}) \to \mathcal{D}$ as follows. The domain $\mathcal{D}$ contains the sets of mappings of the same type: for all τ, $\mathcal{D}^\tau = 2^{\mathrm{Map}^\tau}$. Given a nonterminal F, $[\![F]\!] = \{\theta \mid \exists m \text{ Eve wins from } (F, \theta, m)\}$, given $a \in \Sigma$, $[\![a]\!] = \{\theta \mid \emptyset \vdash a : \theta\}$, and given $d : \tau_1 \to \tau_2 \in \mathcal{D}$ and $d' : \tau_1 \in \mathcal{D}$ $d\ d' = \{\theta \mid \exists (\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \to \theta \in d\ \forall i\ \theta_i \in d'\}$.

▶ **Theorem 6.** *The morphism $[\![\cdot]\!]$ recognises the states of the automaton, i.e. for each state $q \in Q$ of the automaton, it recognises the set $\mathcal{T}_q = \{t : o \mid \|\mathcal{G}_t\| \models \mathcal{A}_q\}$ which is the set of ground terms whose associated value tree is recognised by the automaton from state q. Furthermore, it is stable by rewriting.*

Using the construction of Section 4.0.2, we have the following result.

▶ **Corollary 7** (Automata Reflection). *Higher order recursion schemes are reflective with respect to automata (hence they are reflective with respect to μ-calculus, and to* MSO*).*

6 Selection

Given a signature Σ, we recall the selection problem: given an MSO formula $\varphi[X]$ having one free monadic second order variable X, and a scheme $\mathcal{G}$ such that $\|\mathcal{G}\|$ satisfies the formula

$\exists X\ \varphi[X]$, produce another scheme $\mathcal{G}'$ on the signature $\Sigma \times \{0,1\}$ such that their exists a set S satisfying $\varphi[S]$ and $\|\mathcal{G}'\|$ is a S-marking of $\|\mathcal{G}\|$.

Note that MSO selection implies MSO reflection. Indeed, being able to mark the nodes u satisfying the formula $\varphi[x]$ is equivalent to being able to mark a (unique) set satisfying $\psi[X] = \forall x\ x \in X \Leftrightarrow \varphi[x]$.

As mentioned in Section 3, MSO selection is equivalent to automata selection, therefore we show a construction of a scheme annotating itself with an accepting run of the automaton. Since we cannot embed this problem into a morphism, we have to define another construction, similar in some ways to the one of Section 4.0.2. The construction is also based on Kobayashi-Ong result presented in Section 10. The main difference between the two results (reflexivity and selectivity) is that to construct a reflection of an automaton we embedded the winning region of the game into the scheme, while here we will embed the winning strategy of the game to prove the selection.

▶ **Theorem 8** (Automata Selection)**.** *Higher order recursion schemes are selective with respect to automata (hence to MSO).*

Sketch. In the following, we give an informal glimpse on the proof, the full (technical) proof can be found in Appendix C. Take a scheme $\mathcal{G}$ and an automata $\mathcal{A}$ such that $\|\mathcal{G}\| \models \mathcal{A}$. We want to construct a scheme $\mathcal{G}'$ such that $\|\mathcal{G}'\|$ is an accepting run of $\mathcal{A}$ on $\|\mathcal{G}\|$. The first observation is that there are some great similarities between the structure of the proof trees in the type system system and the definition of a term. Indeed, one can type proofs and one can apply proofs to one another to get new proofs. For example take two terms $t_0 : o \to o$ and $t_1 : o$ and assume that we have a proof $\mathcal{P}_0$ of $t_0 \triangleright (\theta_1, m_1) \to \theta$ and a proof $\mathcal{P}_1$ of $t_1 \triangleright \theta_1$ under some environments. Then one can put together $\mathcal{P}_0$ and $\mathcal{P}_1$ to obtain a proof of $\vdash t_0\ t_1 \triangleright \theta$. We can see this proof as $\mathcal{P}_0\ \mathcal{P}_1$: the application of $\mathcal{P}_1$ to $\mathcal{P}_0$.

In the actual construction, the transformed scheme will not deal with such proofs, but we use this similarity between proofs and terms to create annotations of terms. Given a term t and a proof $\mathcal{P}$ of a judgement $\Gamma \vdash t \triangleright \theta$, we will define the annotated term $t^{\mathcal{P}}$ where each symbol is annotated by an atomic mapping and a color, verifying that if a non terminal F is annotated by (θ, m) then Γ assiciate F to (θ, m). The term $t^{\mathcal{P}}$ is somehow a trace of the proof $\mathcal{P}$.

Now given a rewrite rule $F\ x_1 ... x_k \to e$ in the original scheme, we want to define for all annotated version $F^{(\theta,m)}$ an associated rewrite rule in the transformed scheme. If (F, θ, m) is a winning vertex of the game, then Eve can choose an environment Γ such that $\Gamma \vdash e \triangleright \theta$. We take a proof $\mathcal{P}$ of this judgement and we define $F^{(\theta,m)} x_1 \dots x_k \to e^{\mathcal{P}}$,

Since Eve has chosen the environment Γ with respect to her winning strategy, then for all annotated non terminals $H^{(\theta',m')}$ appearing in $e^{\mathcal{P}}$, (H, θ', m') is winning in the game. In particular, if the initial non terminal of the transformed scheme is $S^{(\theta,m)}$ with (S, θ, m) winning in the game (as it will be), any non terminal in any term of any derivation will be annotated in order to represent a winning vertex in the game. Therefore we do not need to care about which rewrite rule is chosen for $F^{(\theta,m)}$ when (F, θ, m) is not winning.

As we said, the initial non terminal is $(S, q_0, \Omega(q_0))$ which is winning in the game since $\mathcal{A}$ accepts $\|\mathcal{G}\|$ from state q_0.

Any terminal a will be annotated by some $(q_1 \to ... \to q_k \to q, m)$. Then to obtain elements of $\Sigma \times Q$, we just turn any $a^{(q_1 \to \dots \to q_k \to q, m)}$ into a non terminal and we add a rewrite rule transforming it into a^q.

The intuition of why this construction works is the following, based on the proof of the soundness of Kobayashi and Ong construction. To a derivation in the transformed scheme

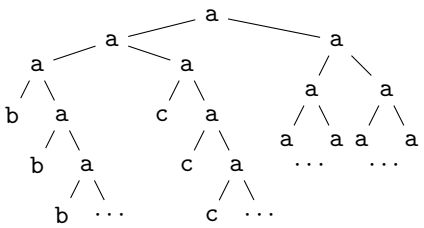


Figure 1 The value tree of the scheme of example **??**.

we associate a tree of plays in the game. The terms will be labeled by some $F^{(\theta,m)}$ that Eve has chosen in the environments she picked, and each time Adam choose one such $F^{(\theta,m)}$, it is rewritten according to Eve's strategy. Due to the colour constraints in the type systems, we can show that from the point where $F^{(\theta,m)}$ is created to the point where it is on the head of a redex, the maximum colour that has been seen is m. Furthermore, we have that along an infinite branch, there is an infinite sequence of nonterminals that are rewritten such that each non terminal is created when the previous one is rewritten. This means that we can map an infinite branch to an infinite play in the game. Furthermore the greatest colour seen infinitely often along this branch is equal to the greatest colour seen infinitely often in the sequence of maximum colours appearing between the non terminals of the sequence. And this is equal to the greatest colour seen infinitely often in the play. Since Eve wins in the game, this colour is even, then for any branch of the value tree of $\mathcal{G}'$ the greatest colour seen infinitely often is even, hence it is an accepting run.

◀

7 An example of scheme transformation

In this section, we present a simple example that describes how one can use the embedding procedure to transform a program.

Let $\mathrm{Map}(\{0,1\})$ be the typed domain inductively defined by $\mathrm{Map}(\{0,1\})^o = \{0,1\}$, $\mathrm{Map}(\{0,1\})^{\tau\to\tau'}$ is the set of total functions from $\mathrm{Map}(\{0,1\})^\tau$ to $\mathrm{Map}(\{0,1\})^{\tau'}$. And given $f : \tau \to \tau'$ and $h : \tau$ in $\mathrm{Map}(\{0,1\})$, $f \cdot h = f(h)$.

Let $\mathcal{G} = \langle \mathcal{V}, \Sigma, \mathcal{N}, S, \mathcal{R}\rangle$ be defined by $\mathcal{V} = \{y : o \to o, x : o\}$, $\Sigma = \{\mathtt{a} : o^2 \to o, \mathtt{b} : o, \mathtt{c} : o\}$, $\mathcal{N} = \{S : o, H : o \to o, J : o \to o, F : (o \to o) \to o\}$, $\mathcal{R}$ contains the following rewrite rules:

$$\begin{array}{lcllcl} S & \to & \mathtt{a}\ (F\ H)\ (F\ J) & H\ x & \to & \mathtt{a}\ x\ (H\ x) \\ J\ x & \to & \mathtt{a}\ (J\ x)\ (J\ x) & F\ y & \to & \mathtt{a}\ (y\ \mathtt{b})\ (y\ \mathtt{c}). \end{array}$$

The value tree of $\mathcal{G}$ is (partially) depicted in Figure 1. We write $[u, v]$ the mapping $f : \{0,1\} \to \{0,1\}$ such that $f(0) = u$ and $f(1) = v$ for all u, v. We define the morphism φ as follows:

$$\begin{array}{llll} \varphi(\mathtt{b}) = 0 & \varphi(\mathtt{c}) = 1 & \varphi(S) = 1 & \varphi(\mathtt{a})\ u\ v = u \vee v \\ \varphi(H)\ u = u & \varphi(J)\ u = 0 & \varphi(F)\ [u,v] = u \vee v. & \end{array}$$

One can check that the morphism φ is stable by rewrite. Furhtermore it recognises the property "t has type o, and its value tree contains a $\mathtt{c}$", with the subset $A' = \{1\}$.

We construct a scheme $\mathcal{G}' = \langle \mathcal{V}', \Sigma', \mathcal{N}', S, \mathcal{R}\rangle$ that consists of an embedding of the morphism φ inside the scheme $\mathcal{G}$. $\mathcal{V}' = \{x : o,\ y^0, y^1 : o \to o\}$, $\Sigma' = \{\mathtt{a}^{0,0}, \mathtt{a}^{0,1}, \mathtt{a}^{1,0}, \mathtt{a}^{1,1} :$

$o^2 \to o,\ \mathtt{b}, \mathtt{c} : o\}$, $\mathcal{N}' = \{S : o,\ H^0, H^1, J^0, J^1 : o,\ F^{[0,0]}, F^{[0,1]}, F^{[1,0]}, F^{[1,1]} : (o \to o)^2 \to o\}$, $\mathcal{R}'$ contains the following rewrite rules:

$$\begin{array}{lcl}
S & \to & \mathtt{a}^{1,0}\ (F^{[0,1]}\ H^0\ H^1)\ (F^{[0,0]}\ J^0\ J^1) \\
H^0\ x & \to & \mathtt{a}^{0,0}\ x\ (H^0\ x) \\
H^1\ x & \to & \mathtt{a}^{1,1}\ x\ (H^0\ x) \\
J^0\ x & \to & \mathtt{a}^{0,0}\ (J^0\ x)\ (J^0\ x) \\
J^1\ x & \to & \mathtt{a}^{0,0}\ (J^1\ x)\ (J^1\ x) \\
F^{[0,0]}\ y^0\ y^1 & \to & \mathtt{a}^{0,0}\ (y^0\ \mathtt{b})\ (y^1\ \mathtt{c}) \\
F^{[0,1]}\ y^0\ y^1 & \to & \mathtt{a}^{0,1}\ (y^0\ \mathtt{b})\ (y^1\ \mathtt{c}) \\
F^{[1,0]}\ y^0\ y^1 & \to & \mathtt{a}^{1,0}\ (y^0\ \mathtt{b})\ (y^1\ \mathtt{c}) \\
F^{[1,1]}\ y^0\ y^1 & \to & \mathtt{a}^{1,1}\ (y^0\ \mathtt{b})\ (y^1\ \mathtt{c}).
\end{array}$$

Let us explain how the rewrite rule related to $F^{[0,1]}$ has been produced. Recall the original rule: $F\ y \to \mathtt{a}\ (y\ \mathtt{b})\ (y\ \mathtt{c})$.

The first occurrence of y is applied to $\mathtt{b}$, therefore it should be annotated with $\varphi(\mathtt{b}) = 0$. Similarly, the second occurrence of y should be annotated with $\varphi(\mathtt{c}) = 1$. This justifies the occurrence of y^0 and y^1 on the left hand part of the rule.

The annotation $[0, 1]$ means that this rule will be applied to an argument whose evaluation is the mapping $[0, 1]$, thus the evaluation of $y\ \mathtt{b}$ is equal to $[0, 1]\ \varphi(\mathtt{b}) = [0, 1]\ 0 = 0$ and by a similar reasoning the evaluation of $y\ \mathtt{c}$ is equal to 1. Therefore the occurrence of $\mathtt{a}$ should be annotated with $(0, 1)$.

We want to transform the scheme in order to forbid the derivation of a subterm when its associated value tree will not include any $\mathtt{c}$. For instance, the occurrence of a $\mathtt{c}$ would correspond to a completed service, and thus such a situation witnesses a useless derivation. In the embedded scheme, this can be detected by applying the evaluation of the head over its annotation. For instance $F^{[0,0]}t_1\ t_2$ is the annotation of a term $F\ t$ whose value is $\varphi(F)\ [0, 0] = 0$. Therefore we might turn the rule associated to $F^{[0,0]}$ into $F^{[0,0]}y_1\ y_2 \to$ FORBIDDEN, where FORBIDDEN $: o$ is a new terminal added to the scheme. Here is the whole set of rewrite rules transformed this way.

$$\begin{array}{lcl}
S & \to & \mathtt{a}^{1,0}\ (F^{[0,1]}\ H^0\ H^1)\ (F^{[0,0]}\ J^0\ J^1) \\
H^0\ x & \to & \text{FORBIDDEN} \\
H^1\ x & \to & \mathtt{a}^{1,1}\ x\ (H^0\ x) \\
J^0\ x & \to & \text{FORBIDDEN} \\
J^1\ x & \to & \text{FORBIDDEN} \\
F^{[0,0]}\ y^0\ y^1 & \to & \text{FORBIDDEN} \\
F^{[0,1]}\ y^0\ y^1 & \to & \mathtt{a}^{0,1}\ (y^0\ \mathtt{b})\ (y^1\ \mathtt{c}) \\
F^{[1,0]}\ y^0\ y^1 & \to & \mathtt{a}^{1,0}\ (y^0\ \mathtt{b})\ (y^1\ \mathtt{c}) \\
F^{[1,1]}\ y^0\ y^1 & \to & \mathtt{a}^{1,1}\ (y^0\ \mathtt{b})\ (y^1\ \mathtt{c}).
\end{array}$$

8 Conclusion

We have given new shape preserving constructions for logical reflection and logical selection using a scheme-only approach, which can be useful for correction or synthesis of programs. The complexity is the same as in the solutions proposed so far, *i.e.* the problem is n-EXPTIME complete, and the size of the new scheme is n-EXP the size of the original one n being the order of the scheme. However, if this approach is implemented for functional

program analysis, the fact that we do not go back and forth to automata will allow to transform the terms on the run, instead of fully dealing with the transformed scheme.

As possible continuation of these work, we may be interesting to see if these results scale for actual program verification, and if they can be included in tools like T-RecS [16], a model checker for HORS.

References

1 Klaus Aehlig. A finite semantics of simply-typed lambda terms for infinite runs of automata. In *CSL'06*, volume 4207 of *LNCS*, pages 104–118, 2006.
2 R. M. Amadio and P-L. Curien. *Domains and Lambda-Calculi*. CTTCS, 1998.
3 Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. Recursion schemes and logical reflection. In *LICS'10*, pages 120–129, 2010.
4 Arnaud Carayol and Olivier Serre. Collapsible pushdown automata and labeled recursion schemes. In *LICS'12*, pages 165–174, 2012.
5 Didier Caucal. On infinite terms having a decidable monadic theory. In *MFCS'02*, volume 2420 of *LNCS*, pages 165–176, 2002.
6 Bruno Courcelle. A representation of trees by languages I. *TCS*, 6:255–279, 1978.
7 Bruno Courcelle. A representation of trees by languages II. *TCS*, 7:25–55, 1978.
8 Bruno Courcelle and Maurice Nivat. The algebraic semantics of recursive program schemes. In *MFCS'78*, volume 64 of *LNCS*, pages 16–30, 1978.
9 Werner Damm. Higher type program schemes and their tree languages. In *Theoretical Computer Science, 3rd GI-Conference*, volume 48 of *LNCS*, pages 51–72, 1977.
10 Werner Damm. Languages defined by higher type program schemes. In *ICALP'77*, volume 52 of *LNCS*, pages 164–179, 1977.
11 Axel Haddad. IO vs OI in higher-order recursion schemes. In *FICS'12*, pages 23–30, 2012.
12 Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *LICS'08*, pages 452–461, 2008.
13 Klaus Indermark. Schemes with recursion on higher types. In *MFCS'76*, volume 45 of *LNCS*, pages 352–358, 1976.
14 Teodor Knapik, Damian Niwiński, and Pawel Urzyczyn. Higher-order pushdown trees are easy. In *FOSSACS'02*, volume 2303 of *LNCS*, pages 205–222, 2002.
15 Naoki Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *POPL'09*, pages 416–428, 2009.
16 Naoki Kobayashi. A practical linear time algorithm for trivial automata model checking of higher-order recursion schemes. In *FOSSACS'11*, pages 260–274, 2011.
17 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *LICS'09*, pages 179–188, 2009.
18 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. Private communication, 2012.
19 M. Nivat. On the interpretation of recursive program schemes. In *Symposia Matematica*, 1972.
20 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS'06*, pages 81–90, 2006.
21 M.O. Rabin. Decidability of second-order theories and automata on infinite trees. In *Trans. Amer. Math. Soc.*, 1969.
22 Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. In *ICALP'11*, pages 162–173, 2011.
23 Sylvain Salvati and Igor Walukiewicz. Using models to model-check recursive schemes. Private communication, 2012.

A Proofs of Results in Section 4

A.1 Embedding a Morphism Into a Scheme, the construction

The construction is adapted from [11]. For technical convenience, we will annotate all symbols by the $\mathcal{D}$-value of their arguments, for example in the term $F\ t_1 ... t_k$, we will label F with the k-tuple $([\![t_1]\!], ..., [\![t_k]\!])$.

A problem may appear if some of the arguments are not fully applied. For example, imagine we want to annotate $F\ H$, where $F : (o \to o) \to o$ and $H : o \to o$: we will annotate F with $[\![H]\!]$, but since H misses its argument we do not know how to annotate it. The problem is that we cannot wait to annotate it because once a non-terminal is created, the derivation does not deal explicitly with it. The solution is to create one copy of H per possible $\mathcal{D}$-value of its arguments, i.e. one per element of $\mathcal{D}^o$. For example assume that there are 4 of them: $\mathcal{D}_o = \{d_1, d_2, d_3, d_4\}$ hence, $F\ H$ will be annotated in the following way: $F^{[\![H]\!]}\ H^{d_1}\ H^{d_2}\ H^{d_3}\ H^{d_4}$. Note that $F^{[\![H]\!]}$ will not have the same type as F: F has type $(o \to o) \to o$, but $F^{[\![H]\!]}$ has type $(o \to o)^4 \to o$. Also note that, even if F has 4 arguments, it only has to be labelled with a single $\mathcal{D}$-value since all four arguments represent different labellings of the same term. We now formalise these notions.

Types

To a type $\tau = \tau_1 \to ... \to \tau_k \to o$ we associate the integer

$$n_\tau = Card\big(\big\{(d_1, ..., d_k) \mid \forall i\ d_i \in \mathcal{D}^{\tau_i}\big\}\big),$$

and we fix an ordering $\{\vec{d^\tau_1}, ..., \vec{d^\tau_{n_\tau}}\}$ over the set $\{(d_1, ..., d_k) \mid \forall i\ d_i \in \mathcal{D}^{\tau_i}\}$. We define inductively the type τ^+ as $\tau^+ = (\tau_1^+)^{n_{\tau_1}} \to ... \to (\tau_k^+)^{n_{\tau_k}} \to o$. Note that, as $n_o = 1$, $(o^k \to o)^+ = o^k \to o$.

Symbols

To a terminal $a : o^k \to o$ and a tuple $d_1, ..., d_k \in \mathcal{D}^o$, we associate the terminal $a^{d_1,...,d_k} : o^k \to o \in \Sigma'$.

To a non terminal $F : \tau = \tau_1 \to ... \to \tau_k \to o$ (resp. a variable $x : \tau$) and a tuple $d_1 : \tau_1, ..., d_k : \tau_k$, we associate the non-terminal $F^{d_1,...,d_k} : \tau^+ \in \mathcal{N}'$ (resp. the variable $x^{d_1,...,d_k} : \tau^+ \in \mathcal{V}'$).

Terms

Given a mapping $\rho : \mathcal{W} \to \mathcal{D}$ with $\mathcal{W} \subseteq \mathcal{V}$, such that if $x : \tau$, $\rho(x) : \tau$, let $[\![\cdot]\!]_\rho$ be the morphism from $\mathcal{T}(\Sigma \uplus \mathcal{N} \uplus \mathcal{W})$ to $\mathcal{D}$ defined by letting $[\![\alpha]\!]_\rho = [\![\alpha]\!]$ for $\alpha \in \Sigma \uplus \mathcal{N}$ and $[\![x]\!]_\rho = \rho(x)$ for $x \in \mathcal{W}$.

Given a term $t : \tau = \tau_1 \to ... \to \tau_k \to o \in \mathcal{T}(\mathcal{V} \uplus \Sigma \uplus \mathcal{N})$ and a mapping $\rho : \mathcal{W} \to \mathcal{D}$ such that $\mathcal{W} \subseteq \mathcal{V}$ contains all the variables appearing in t, we define inductively the term $t_\rho^{+d_1,...,d_k} : \tau^+ \in \mathcal{T}(\mathcal{V}' \uplus \Sigma' \uplus \mathcal{N}')$ for all $d_1 : \tau_1, ..., d_k : \tau_k$. If $t = F \in \mathcal{N}$ (resp. $t = x \in \mathcal{V}$), $t_\rho^{+d_1,...,d_k} = F^{d_1,...,d_k}$ (resp. $t_\rho^{+d_1,...,d_k} = x^{d_1,...,d_k}$), if $t = a \in \Sigma$, $t_\rho^{+d_1,...,d_k} = a^{d_1,...,d_k}$. Finally consider the case where $t = t_1\ t_2$ with $t_1 : \tau' \to \tau$ and $t_2 : \tau'$. Let $d = [\![t_2]\!]_\rho$. Remark that ${t_1}_\rho^{+d,d_1,...,d_k}$ has type $(\tau'^+)^{n_{\tau'}} \to \tau^+$. We define

$$(t_1\ t_2)_\rho^{+d_1,...,d_k} = {t_1}_\rho^{+d,d_1,...,d_k}\ \vec{t_2}_\rho,$$

where $\vec{t_{2\rho}} = t_{2\rho}^{+\vec{d}_1^{\tau'}}, \ldots, t_{2\rho}^{+\vec{d}_{n_{\tau'}}^{\tau'}}$. Note that since this transformation is only duplicating and annotating, given a term $t^{+d_1,\ldots,d_k}$ we can uniquely find the unique term t associated to it. In the same way, a tree on Σ' is simply an annotated tree on Σ.

Rewrite Rules

Let $F : \tau_1 \to ... \to \tau_k \to o \in \mathcal{N}$, $d_1 : \tau_1, ..., d_k : \tau_k$, and $\rho = x_1 \mapsto d_1, ..., x_k \mapsto d_k$. If $F\ x_1 ... x_k \ \to\ e \in \mathcal{R}$, we define in $\mathcal{R}'$ the rule

$$F^{d_1,\ldots,d_k}\ \vec{x}_1 ... \vec{x}_k \ \to\ e_\rho^+,$$

where for all i, $\vec{x}_i = x_i^{\vec{d}_1^{\tau_i}}, \ldots, x_1^{\vec{d}_{n_{\tau_i}}^{\tau_i}}$. Finally, we let $\mathcal{G}' = \langle \mathcal{V}', \Sigma', \mathcal{N}', \mathcal{R}', S \rangle$.

A.2 Embedding a Morphism Into a Scheme, The Proof..

In the following we prove that this construction satisfies Theorem 4:

Given two terms $t, t' : o \in \mathcal{T}(\Sigma \uplus \mathcal{N})$, if $t \Rightarrow_{\mathcal{G}} t'$, then $t^+ \Rightarrow_{\mathcal{G}'} t'^+$. In particular $Unlab(\|\mathcal{G}'_{t^+}\|) = \|\mathcal{G}_t\|$ (where $Unlab$ is the function that removes the annotations).

Proof of Theorem 4.0.2. First we need the following lemma.

▶ **Lemma 9.** *Given $F : \tau_1 \to ... \to \tau_k \to o \in \mathcal{N}$, $t_1 : \tau_1, ..., t_k : \tau_k \in \mathcal{T}(\Sigma \uplus \mathcal{N})$, $F\ x_1 ... x_k \to e \in \mathcal{R}$ and $\rho = x_1 \mapsto [\![t_1]\!], ..., x_k \mapsto [\![t_k]\!]$. Then*

$$(e_{[\forall i\ x_i \mapsto t_i]})^+ = e^+_{\left[\forall i,j\ x_i^{\vec{d}_j^{\tau_i}} \mapsto t_i^{\vec{d}_j^{\tau_i}}\right]}.$$

Proof. The proof consists in an induction on the structure of e. ◀

We prove a slightly more general result that the one stated in the theorem: given two terms $t, t' : \tau'_1 \to ... \to \tau'_\ell \to o \in \mathcal{T}(\Sigma \uplus \mathcal{N})$, and $d_1 : \tau'_1, ..., d_\ell : \tau'_\ell \in \mathcal{D}$, if $t \Rightarrow t'$, then $t^{+d_1,\ldots,d_\ell} \Rightarrow t'^{+d_1,\ldots,d_\ell}$. We prove this result by induction on the structure of t.

Assume that $t = \alpha\ t_1, ..., t_k$ with $\alpha \in \Sigma \uplus \mathcal{N}$. If $\alpha \in \Sigma$ or $\alpha \in \mathcal{N}$ and $k < \text{arity}(\alpha)$. Then $t' = \alpha\ t'_1 ... t'_k$ such that for all i, $t_i \Rightarrow t'_i$. Therefore

$$t^{+d_1,\ldots,d_\ell} = \alpha^{[\![t_1]\!],\ldots,[\![t_k]\!],d_1,\ldots,d_\ell}\ \ t_1^{+\vec{d}_1^{\tau_1}} ... \ t_1^{+\vec{d}_{n_{\tau_1}}^{\tau_1}}\ \ ...\ \ t_k^{+\vec{d}_1^{\tau_k}} ... \ t_k^{+\vec{d}_{n_{\tau_k}}^{\tau_k}}.$$

In this term, the head α is also not fully applied, then let s be such that $t^{+d_1,\ldots,d_\ell} \Rightarrow s$, we have

$$s = \alpha^{[\![t_1]\!],\ldots,[\![t_k]\!],d_1,\ldots,d_\ell}\ \ s_1^1 ... \ s_1^{n_{\tau_1}}\ \ ...\ \ s_k^1 ... \ s_k^{n_{\tau_k}},$$

with for all i, j, $t_i^{+\vec{d}_j^{\tau_i}} \Rightarrow s_i^j$. Then by induction hypothesis for all i, j we have $s_i^j = t_i'^{+\vec{d}_j^{\tau_i}}$. Then $s = t'^{+d_1,\ldots,d_\ell}$.

If $\alpha \in \mathcal{N}$ and $k = \text{arity}(\alpha)$ (in particular $\ell = 0$). Take $\alpha\ x_1 ... x_k \to e \in \mathcal{R}$. We have $t' = e_{[\forall i\ x_i \mapsto t'_i]}$ with $t_i \to t'_i$. Let s be such that $t^+ \Rightarrow s$. Then

$$s = e^+_{\rho\left[\forall i,j\ x_i^{\vec{d}_j^{\tau_i}} \mapsto s_i^j\right]}$$

with $t_i^{+\vec{d}_j^{r_i}} \Rightarrow s_i^j$, and $\rho = x_1 \mapsto [\![t_1]\!], ..., x_k \mapsto [\![t_k]\!]$. Then by induction hypothesis for all i, j we have $s_i^j = t_i'^{+\vec{d}_j^{r_i}}$. Lemma 9 states then that

$$s = (e_{[\forall i\ x_i \mapsto t_i]})^+ = (t')^+.$$

◀

B Proofs of Results in Section 5.0.5

B.1 Definition of a two player game

A ***max parity game*** is a tuple $\mathbb{G} = \langle V, (V_E, V_A), E, \Omega\rangle$ where V is a set called the ***set of vertices***, (V_E, V_A) is a partition of V where V_E (*resp.* V_A) is called ***the set of vertices of Eve*** (*resp.* of ***Adam***), $E \subseteq V \times V$ is called the ***transition relation***, $\Omega : (V_E \cup V_A) \to \{1, \ldots, m_{\max}\}$ for some $m_{\max} \in \mathbb{N}$ is the ***colouring function***.

A ***play*** in the game is a possibly infinite maximal sequence $p = v_0 \cdot v_1 \cdot \ldots$ in $V^* \cup V^\omega$ such that for all i, $(v_i, v_{i+1}) \in E$, and if $p = v_1 \cdot \ldots \cdot v_k$ is finite, there is no v such that $(v_k, v) \in E$. ***Eve wins the play*** p if p is finite and the last vertex is in V_A or if p is infinite and the maximum colour seen infinitely often in $\Omega(v_0), \Omega(v_1), \ldots$ is even.

A ***(positional) strategy*** φ for Eve is a mapping $\varphi : V_E \to V$ such that for all $v \in V_E$, $(v, \varphi(v)) \in E$. Given a play $v_0 \cdot v_1 \cdot \ldots$ we say that Eve respects φ if for all $v_i \in V_E$, $v_{i+1} = \varphi(v_i)$. We say that ***the strategy is winning*** from a vertex v_0 if Eve wins all play where she respects φ and that starts from v_0. A ***strategy*** φ for Eve is a mapping $\varphi : (V_E \cdot V_A)^* \cdot V_E \to V_A$ such that for all $p = v_0, \ldots, v_k \in (V_E \cdot V_A)^* \cdot V_E$, $(v_k, \varphi(p)) \in E$. Given a play p we say that Eve plays according to φ in p, if for all $v_i \in V_E$, $v_{i+1} = \varphi(v_0, \ldots, v_i)$. We say that ***the strategy is winning*** if Eve wins all play p where she plays according to φ. A strategy φ is ***positional*** (or memoryless) if there exists a mapping $f : V_E \to V_A$ such that for all or all $p = v_0, \ldots, v_k \in (V_E \cdot V_A)^* \cdot V_E$, $\varphi(p) = f(v_k)$ (usually we directly give f as the positional strategy).

A well known result is that if there exists a winning strategy for Eve, then there exist a positional one. Furthermore, deciding if there exists a winning strategy is computable, and the problem is in $\mathbf{NP} \cap \mathbf{coNP}$.

B.2 A formal presentation of Kobayashi Ong result

In the following we fix a non deterministic parity tree automaton $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, \Omega\rangle$ and a scheme $\mathcal{G} = \langle \Sigma, \mathcal{N}, \mathcal{V}, S, \mathcal{R}\rangle$. We let $\text{arity}_{\max}$, $\text{order}_{\max}$, and $m_{\max}$ be the maximum arity in $\Sigma \uplus \mathcal{N}$ (*resp.* the maximum order in $\Sigma \uplus \mathcal{N}$, the maximum colour in $\Omega(Q)$).

The idea of the result is to define a type system, and to use this type system in the construction of a two player parity game, such that Eve wins the game if and only if the automaton accepts the value tree of the scheme.

B.2.1 The Type System

We define inductively the typed set Map of ***atomic mappings*** as $\text{Map}^o = Q$ and

$$\text{Map}^{\tau\to\tau'} = \Big\{\sigma \to \theta' \mid \theta' \in \text{Map}^\tau, \quad \sigma \subseteq \{(\theta, m) \mid \theta \in \text{Map}^\tau, m \geq \max(\Omega(\theta), \Omega(\theta'))\}\Big\},$$

where $\Omega(\theta)$ is defined inductively as $\Omega(\theta) = \Omega(q)$ if $\theta = q \in Q$ and $\Omega(\theta) = \Omega(\theta')$ if $\theta = \sigma \to \theta'$. For all type τ we choose and fix a complete ordering of the finite set Map^τ, and use the notation $(\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \to \theta'$ to denote the mapping $\{(\theta_1, m_1), \ldots, (\theta_k, m_k)\} \to \theta'$ where the sequence $(\theta_1, m_1), \ldots, (\theta_k, m_k)$ is ordered. In the following we will use the letter θ to range over atomic mappings.

An environment Γ is a set of tuples $\alpha \triangleright (\theta, m)^b$ (called ***bindings***) with $\alpha \in \mathcal{N} \uplus \mathcal{V}$, θ an atomic mapping, $m \leq m_{\max}$ a colour, and $b \in \{\mathtt{f}, \mathtt{t}\}$, called ***the flag***.

Intuitively, $\alpha \triangleright (\theta, m)^{\mathtt{f}}$ means "an occurrence of α can be typed by θ if the greatest colour seen along the path from the root of the term to the occurrence of α is m". The binding $\alpha \triangleright (\theta, m)^{\mathtt{t}}$ is less restrictive, it means "an occurrence of α can be typed by θ if no colour seen in the path from the root of the term to the occurrence of α is greater or equal than m". In the type system the flag will switch from $\mathtt{f}$ to $\mathtt{t}$ whenever the colour m is seen.

Given the tuple $(\theta, m)^b$ and a colour m' we define $(\theta, m)^b \uparrow m'$ as

$$(\theta, m)^b \uparrow m' = \begin{cases} (\theta, m)^b & \text{if } m' < m \\ (\theta, m)^{\mathtt{t}} & \text{if } m' = m \\ \text{undefined} & \text{if } m' > m \end{cases}$$

Intuitively, $(\theta, m)^b \uparrow m'$ is the way to inform $(\theta, m)^b$ that the colour m' has been seen. We define $\{\alpha_1 \triangleright (\theta_1, m_1)^{b_1}, ..., \alpha_\ell \triangleright (\theta_\ell, m_\ell)^{b_\ell}\} \uparrow m' = \{\alpha_1 \triangleright (\theta_1, m_1)^{b_1} \uparrow m', ..., \alpha_\ell \triangleright (\theta_\ell, m_\ell)^{b_\ell} \uparrow m'\}$. Note that it is undefined if one of the $(\theta_i, m_i)^{b_i} \uparrow m'$ is undefined.

We define a type judgment as $\Gamma \vdash t \triangleright \theta$ with Γ an environment, t a term and θ an atomic mapping. We say that a type judgment holds if it can be derived from the following judgment rules:

$$\frac{}{\{\alpha \triangleright (\theta, m)^b\} \vdash \alpha \triangleright \theta} \text{ (T-Var)}$$

if $(\theta, m)^b \uparrow \Omega(\theta)$ is defined and equal to $(\theta, m)^{\mathtt{t}}$,

$$\frac{}{\emptyset \vdash a \triangleright (q_1, m_1) \to ... \to (q_k, m_k) \to q} \text{ (T-Const)}$$

if $a \in \Sigma$ and $q \xrightarrow{a} q_1, ..., q_k \in \delta_{\mathcal{A}}$ and for all i, $m_i = \max(\Omega(q_i), \Omega(q))$,

$$\frac{\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta \ \forall i \ \Gamma_i \uparrow m_i \vdash t_1 \triangleright \theta_i}{\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \vdash t_0 \ t_1 \triangleright \theta} \text{ (T-App)}$$

if for all i, $\Gamma_i \uparrow m_i$ is defined.

▶ Remark. Notice that if $\Gamma \vdash t \triangleright \theta$ holds then every binding $\alpha \triangleright (\theta, m)^b \in \Gamma$ needs to be used in the proof of $\Gamma \vdash t \triangleright \theta$. For example the judgement $\{\alpha \triangleright (\theta, m)^b, \alpha \triangleright (\theta', m')^{b'}\} \vdash \alpha \triangleright \theta$ with $(\theta, m)^b \neq (\theta', m')^{b'}$ does not hold, even if $\{\alpha \triangleright (\theta, m)^b\} \vdash \alpha \triangleright \theta$ holds. In particular if $\alpha \triangleright (\theta, m)^b \in \Gamma$ and $\Gamma \vdash t \triangleright \theta$ then necessarily there is an occurrence of α in t.

In the following, we may use the shorthand $\bigwedge_{j \in J}(\theta_j, m_j)$ for the sequence $(\theta_{j_1}, m_{j_1}) \wedge ... \wedge (\theta_{j_k}, m_{j_k})$ with $J = \{j_1, ..., j_k\}$, and $\alpha \triangleright \bigwedge_{j \in J}(\theta_j, m_j)^{b_j}$ for the environment $\alpha \triangleright (\theta_{j_1}, m_{j_1})^{b_{j_1}}, ..., \alpha \triangleright (\theta_{j_k}, m_{j_k})^{b_{j_k}}$.

Given a rewrite rule $r_F = F \ x_1 ... x_k \to e$, an environment Γ and an atomic mapping $\theta' = \bigwedge_{j \in J_1}(\theta_{1j}, m_{1j}) \to ... \to \bigwedge_{j \in J_k}(\theta_{kj}, m_{kj}) \to \theta$, we use the notation $\Gamma \vdash r_F \triangleright \theta'$ to denote the fact that for all i there exists $I_i \subseteq J_i$ such that $\Gamma, x_1 \triangleright \bigwedge_{j \in I_1}(\theta_{1j}, m_{1j}), ..., x_k \triangleright \bigwedge_{j \in I_k}(\theta_{kj}, m_{kj}) \vdash e \triangleright \theta$.

B.2.2 The Game

Now we define a game in which Eve's states will be triples made of a non terminal, an atomic mapping and a colour, and Adam's states will be environments. Eve chooses an environment that can judge the rewrite rule of the current nonterminal with the current atomic mapping, while Adam picks one binding in the current environment.

Formally, we define the max-parity game $\mathbb{G}_{\mathcal{A}} = \langle V_\forall, V_\exists, E, \Omega' \rangle$ as follow: $V_\exists = \{(F, \theta, m) \mid F \in \mathcal{N}, F : \tau, \theta : \tau\}$, $V_\forall = \{\Gamma \mid \forall \alpha \triangleright (\theta, m)^b \in \Gamma\ \alpha \in \mathcal{N} \wedge b = \mathtt{f}\}$, $E = \{\big((F, \theta, m), \Gamma\big) \mid \Gamma \vdash r_F \triangleright \theta\} \cup \{\big(\Gamma, (F, \theta, m)\big) \mid F \triangleright (\theta, m)^{\mathtt{f}} \in \Gamma\}$, For all $(F, \theta, m) \in V_\exists$, $\Omega'\big((F, \theta, m)\big) = m$ and for all $\Gamma \in V_\forall$, $\Omega'(\Gamma) = 0$.

Intuitively, Eve tries to show a well-typing of the terms with respect to the rewrite rules, that would induce a well-coloured run of the automaton, and Adam tries to show that she is wrong. From the state (F, θ, m), Eve has to find an environment Γ such that she can prove $\Gamma \vdash r_F : \theta$, then Adam picks a biding $F' \triangleright (\theta', m')^{\mathtt{f}}$ in Γ and asks Eve to prove that θ' is chosen correctly. If at some point of a play, Eve cannot find a correct environment, she loses the play; if she can choose the empty environment, Adam would have nothing to choose then she wins the play; if the play is infinite, Eve wins if and only if the greatest colour seen infinitely often is even.

▶ **Theorem 10** (Kobayashi, Ong 09). *The tree $\|\mathcal{G}\|$ is accepted by $\mathcal{A}$ from the state q if and only if Eve has a positional winning strategy from the vertex $(S, q, \Omega(q))$ in the game $\mathbb{G}_{\mathcal{A}}$.*

B.3 A Morphism from Kobayashi and Ong Construction, Proof of Theorem 6.

We recall the morphism $[\![\cdot]\!] : \mathcal{T}(\Sigma \uplus \mathcal{N}) \to \mathcal{D}$. The domain $\mathcal{D}$ contains the sets of atomic mappings of the same type: for all τ, $\mathcal{D}^\tau = 2^{\mathrm{Map}^\tau}$. Given a nonterminal F, $[\![F]\!] = \{\theta \mid \exists m \textit{ Eve wins from } (F, \theta, m)\}$, given $a \in \Sigma$, $[\![a]\!] = \{\theta \mid \emptyset \vdash a : \theta\}$, and given $d : \tau_1 \to \tau_2 \in \mathcal{D}$ and $d' : \tau_1 \in \mathcal{D}$:

$$d\ d' = \left\{\theta \;\middle|\; \exists (\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \to \theta \in d\ \forall i\ \theta_i \in d'\right\}.$$

We will show that for any ground term $t : o$, $[\![t]\!] = \{q \mid \|\mathcal{G}_t\| \models \mathcal{A}_q\}$, and that the morphism is stable by rewriting.

In the following we define an environment $\Gamma^{\mathtt{t}}$ and we show that given a term $t : o$ and a state q, there exists $\Gamma \subseteq \Gamma^{\mathtt{t}}$ such that $\Gamma \vdash t \triangleright q$ if and only if $\|G_t\|$ is recognised by $\mathcal{A}$. Then we show that given a term $t : \tau$ and an atomic mapping $\theta : \tau$, there exists $\Gamma \subseteq \Gamma^{\mathtt{t}}$ such that $\Gamma \vdash t \triangleright \theta$ if and only if $\theta \in [\![t]\!]$.

B.3.1 The Environment $\Gamma^{\mathtt{t}}$ Witnesses the Recognition of $\mathcal{A}$

We show that given a term $t : o$, $\mathcal{A}$ accepts $\|G_t\|$ from state q if and only if $q \in \Gamma^{\mathtt{t}}(t)$.

▶ **Lemma 11.** *If (F, θ, m) is winning for some m then, for all m', (F, θ, m') is winning.*

Proof. From (F, θ, m') Eve follows the exact same strategy. ◀

Let $\mathcal{S} = \{(F, \theta) \mid \textit{there exists a winning vertex } (F, \theta, m)\}$ (a winning vertex is a vertex from which Eve has a winning strategy). Note that Theorem 11 states that if $(F, \theta) \in \mathcal{S}$ then for all m, (F, θ, m) is winning. Let m_{max} be the maximum color. We define $\Gamma^{\mathtt{t}} = \{F \triangleright (\theta, m_{max})^{\mathtt{t}} \mid (F, \theta) \in \mathcal{S}\}$ and $\Gamma^{\mathtt{f}} = \{F \triangleright (\theta, m)^{\mathtt{f}} \mid (F, \theta) \in \mathcal{S}, m \textit{ a color}\}$.

In the following, we say that an environment Γ ***witnesses*** $t \triangleright \theta$ if there exists a subset Γ' of Γ such that $\Gamma' \vdash t \triangleright \theta$. We write $\Gamma(t)$ the set of atomic mappings such that Γ witnesses $t \triangleright \theta$.

We define the mapping Π from environments to sets of couple of nonterminal and type as $\Pi(\Gamma) = \{(F, \theta) \mid \exists m, b\ F \triangleright (\theta, m)^b \in \Gamma\}$. Note that $\Pi(\Gamma^{\mathtt{f}}) = \Pi(\Gamma^{\mathtt{t}}) = \mathcal{S}$. Finally note that a subset Γ of $\Gamma^{\mathtt{t}}$ is entirely defined by $\Pi(\Gamma)$ (indeed, $\Gamma = \{(F, \theta, m_{max}) \mid (F, \theta) \in \Pi(\Gamma)\}$).

The idea of the proof is the following : first we show that $\mathcal{A}$ accepts $\|G_t\|$ from state q if and only if $q \in \Gamma^f(t)$, and then we show that for all term t, $\Gamma^f(t) = \Gamma^t(t)$.

▶ **Lemma 12** ($\Gamma^{\mathtt{f}}$ Witnesses the Recognition of $\mathcal{A}$)**.** *Given a term $t : o$ and a state q of the automaton, then $\mathcal{A}$ accepts $\|G_t\|$ from state q if and only if there exists a subset Γ of $\Gamma^{\mathtt{f}}$ such that $\Gamma \vdash t : q$.*

Proof of theorem 12. First, as we pointed out before, notice that if for some Γ, s and θ, $\Gamma \vdash s : \theta$ holds then all the elements of Γ have to be used in the proof using rule (T-Var). In particular, if s does not contain any occurrence of a non-terminal F then there is no element of Γ of the form $F \triangleright (\theta', m)^b$.

Recall that $\mathcal{G}_t = \langle \Sigma, \mathcal{N} \uplus I, \mathcal{V}, \mathcal{R} \uplus I \to t, I \rangle$. Let $\mathbb{G} = \langle V_A, V_E, E, \Omega \rangle$ be the game associated to $\mathcal{G}$ and $\mathcal{A}$, and $\mathbb{G}_t$ the game associated to $\mathcal{G}_t$ and $\mathcal{A}$. Notice that $\mathbb{G}_t = \langle V_A, V_E \uplus \{(I, q, m) \mid q \in Q, m \in \mathbb{N}\}, E \uplus \{(I, q, m) \to \Gamma \mid \Gamma \vdash t : q\}, \Omega\}$. In particular, any winning strategy in $\mathbb{G}_t$ is also a winning strategy in $\mathbb{G}$, and any winning strategy in $\mathbb{G}$ is a winning strategy in $\mathbb{G}_t$ on the vertices of V_E.

Assume that $\mathcal{A}$ accepts $\|\mathcal{G}_t\|$ from state q. Then Theorem 10 states that the vertex $(I, q, \Omega(q))$ is winning in the game, hence there is a positional winning strategy defined on $(I, q, \Omega(q))$, and let Γ be the choice of Eve of vertex $(I, q, \Omega(q))$ according to the strategy. Note that Γ does not contains any $I \triangleright (q', m)^{\mathtt{f}}$ since I does not appear in t. Furthermore, since it is a winning strategy, if $F : (\theta, m)^{\mathtt{f}} \in \Gamma$ then (F, θ, m) is winning in $\mathbb{G}_t$ hence it is winning in $\mathbb{G}$ then $\Gamma \subseteq \Gamma^{\mathtt{f}}$. From the definition of the game comes that $\Gamma \vdash t \triangleright q$.

On the other hand, assume that there is a subset Γ of $\Gamma^{\mathtt{f}}$ such that $\Gamma \vdash t \triangleright q$. Then the strategy of playing Γ on state $(I, q, \Omega(q))$ and $\Gamma_{(F,\theta,m)}$ on (F, θ, m) for all winning vertices (F, θ, m) is winning, hence $\mathcal{A}$ accepts $\|G_t\|$ from state q. ◀

▶ **Lemma 13.** *Let Γ be a subset of $\Gamma^{\mathtt{t}}$, and m a color, then $\Gamma \uparrow m$ is defined and is equal to Γ. Furthermore for any environment Γ, if $\Gamma \uparrow m$ is defined, then $\Pi(\Gamma \uparrow m) = \Pi(\Gamma)$.*

Proof. It is sufficient to notice that since $m_{max} \geq m$ $(\theta, m_{\max})^{\mathtt{t}} \uparrow m$ is defined and is equal to $(\theta, m_{\max})^{\mathtt{t}}$. The second point comes from the fact that the operator $\uparrow m$ does not change the colors. ◀

▶ **Lemma 14.** *(Technical, but needed to be formal) Given an environment Γ, a term t, and $F \in \mathcal{N}$. Given some θ' and m, assume that Γ does not contain neither the binding $F \triangleright (\theta', m)^f$ or $F \triangleright (\theta', m)^t$.*

If $\Gamma \uplus \{F \triangleright (\theta', m)^b\} \vdash t \triangleright \theta$ for some b then $\Gamma \cup \{F \triangleright (\theta', m')^t\} \vdash t \triangleright \theta$ for all $m' \geq m$.

Proof. We prove the result by induction on the structure of t.

We know that $t \neq a$ with $a \in \Sigma$ because if $\Gamma'' \vdash a : \theta$ then $\Gamma = \{\}$. We know that $t \neq H$ with $H \neq F$ a nonterminal because if $\Gamma'' \vdash H \triangleright \theta$ then $\Gamma'' = \{H \triangleright (\theta, m'')^{b''}\}$ for some m'' and b''.

If $t = F$, then $\Gamma = \{\}$, $\theta' = \theta$ and $(\theta, m)^b \uparrow \Omega(\theta) = (\theta, m)^{\mathtt{t}}$, hence $m \geq \Omega(\theta)$ therefore, since $m' \geq m \geq \Omega(\theta)$, $(\theta, m')^{\mathtt{t}} \uparrow \Omega(\theta) = (\theta, m')^{\mathtt{t}}$, thus $\{F \triangleright (\theta, m')^{\mathtt{t}}\} \vdash F \triangleright \theta$.

If $t = t_0\ t_1$ then there exists $\Gamma_0, \Gamma_1, ..., \Gamma_k$ such that $\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k = \Gamma \uplus \{F \triangleright (\theta', m')^{\mathtt{t}}\}$, $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ and for all $i \in \{1, ..., k\}$, $\Gamma_i \uparrow m_i \vdash t_1 \triangleright \theta_i$. If $\Gamma_0 = \Gamma_0' \uplus \{F \triangleright (\theta', m)^b\}$ then by induction hypothesis $\Gamma_0' \cup \{F \triangleright (\theta', m')^{\mathtt{t}}\} \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$. If for some $i \geq 1$, $\Gamma_i = \Gamma_i' \uplus \{F \triangleright (\theta', m)^b\}$ then $(\theta', m)^b \uparrow m_i$ is define and $(\theta', m)^b \uparrow m_i = (\theta', m)^{b'}$ and $\Gamma_i \uparrow m_i = (\Gamma_i' \uparrow m_i) \uplus \{F \triangleright ((\theta', m)^{b'}\}$. Then $m \geq m_i$ hence $m' \geq m_i$, thus $(\theta', m')^{\mathtt{t}} \uparrow m_i$ is define and is equal to $(\theta', m')^{\mathtt{t}}$ then $(\Gamma_i' \uplus \{F \triangleright (\theta', m)^b\}) \uparrow m_i = (\Gamma_i' \uparrow m_i) \cup \{F \triangleright ((\theta', m)^{b'}\} \vdash t_i : \theta_i$. Then $\Gamma \uplus \{F \triangleright (\theta', m')^{\mathtt{t}}\} \vdash t \triangleright \theta$.

◀

▶ **Corollary 15.** *Let* $\Gamma \subseteq \Gamma^f$ *such that* $\Gamma \vdash t : \theta$*. Let* $\Gamma' \subseteq \Gamma^t$ *such that* $\Pi(\Gamma') = \Pi(\Gamma)$ *then* $\Gamma^t \vdash t \triangleright \theta$*.*

Proof. The proof proceeds by induction on the number of non terminals in Γ. ◀

▶ **Lemma 16** ($\Gamma^{\mathtt{t}}(t) = \Gamma^{\mathtt{f}}(t)$)**.** *Given a term* $t : \tau$ *and an atomic mapping* $\theta : \tau$*, if there is a subset* Γ' *of* Γ^t *such that* $\Gamma' \vdash t \triangleright \theta$*, there there is a subset* Γ *of* Γ^f *such that* $\Gamma \vdash t \triangleright \theta$*.*

Proof. We prove this result by induction of the structure of t.

If $t = a$ with a a terminal then $\Gamma' = \Gamma = \{\}$.

If $t = F$ with F a nonterminal, then $\Gamma' = \{F \triangleright (\theta, m_{max})^{\mathtt{t}}\}$, we define $\Gamma = \{F \triangleright (\theta, \Omega(\theta))^{\mathtt{f}}\}$. Since $(\theta, \Omega(\theta))^{\mathtt{f}} \uparrow \Omega(\theta) = (\theta, \Omega(\theta))^{\mathtt{t}}$, we have $\Gamma \vdash F \triangleright \theta$.

If $t = t_0\ t_1$ then $\Gamma' \vdash t \triangleright \theta$ is obtained using rule (T-App). $\Gamma' = \Gamma'_0 \cup \Gamma'_1 \cup ... \cup \Gamma'_k$ with $\Gamma'_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ and for all $i \in \{1, ..., k\}$ $\Gamma'_i \vdash t_1 \triangleright \theta_i$. Then, by induction hypothesis let $\Gamma_0, \Gamma_1, ..., \Gamma_k \subseteq \Gamma^{\mathtt{f}}$ such that $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ and for all $i \in \{1, ..., k\}$ $\Gamma_i \vdash t_1 \triangleright \theta_i$. For all $i \in \{1, ..., k\}$ we define $\Gamma''_i \subseteq \Gamma^{\mathtt{f}}$ as $\Gamma''_i = \Gamma_i \Uparrow m_i$ (defined by $\Gamma''_i = \{F \triangleright (\theta', max(m_i, m'))^{\mathtt{f}} | F \triangleright (\theta, m')^{\mathtt{f}} \in \Gamma\}$). Then we have $\Gamma''_i \uparrow m_i = \{F \triangleright (\theta', m_i)^{\mathtt{t}} | F \triangleright (\theta, m')^{\mathtt{f}} \in \Gamma_i, m' \neq m_i\} \cup \{F \triangleright (\theta', m')^{\mathtt{f}} | F \triangleright (\theta, m')^{\mathtt{f}} \in \Gamma_i, m' > m_i\}$, thus, using Theorem 14 one can state $\Gamma''_i \uparrow m_i \vdash t_1 \triangleright \theta_i$, then if we define $\Gamma = \Gamma_0 \cup \Gamma''_1 \cup ... \cup \Gamma''_k \subseteq \Gamma^{\mathtt{f}}$, we have $\Gamma \vdash t \triangleright \theta$.

◀

▶ **Theorem 17** ($\Gamma^{\mathtt{t}}$ Witnesses the Recognition of $\mathcal{A}$)**.** *Given a term* $t : o$ *and a state* q *of the automaton, then* $\mathcal{A}$ *accepts* $\|G_t\|$ *from state* q *if and only if there exists a subset* Γ' *of* Γ^t *such that* $\Gamma' \vdash t : q$*.*

Proof. This is just a consequence of Lemma 12 and Lemma 16. ◀

B.3.2 The Environment $\Gamma^{\mathtt{t}}$ Witnesses the Morphism $[\![\cdot]\!]$

We show that for all term t, $\Gamma^{\mathtt{t}}(t) = [\![t]\!]$. Furthermore we show that the morphism is stable by rewriting.

▶ **Theorem 18.** *Given a term* t*,* $[\![t]\!] = \Gamma^t(t)$*.*

Proof. We prove this result by induction on the structure of t. If $t = a \in \Sigma$ then $\Gamma^{\mathtt{t}}(a) = \bigcup_{\Gamma \subseteq \Gamma^{\mathtt{t}}} \{\theta \mid \Gamma \vdash a \triangleright \theta\}$, for all $\Gamma \neq \emptyset$ we have $\{\theta \mid \Gamma \vdash a \triangleright \theta\} = \emptyset$, then $\Gamma^{\mathtt{t}}(a) = \{\theta \mid \emptyset \vdash a \triangleright \theta\} = [\![a]\!]$.

If $t = F \in \mathcal{N}$ then $\Gamma^{\mathtt{t}}(a) = \bigcup_{\Gamma \subseteq \Gamma^{\mathtt{t}}} \{\theta \mid \Gamma \vdash a \triangleright \theta\}$, for all $\Gamma \neq \{F \triangleright (\theta', m_{max})^{\mathtt{t}}\}$ for some θ' we have $\{\theta \mid \Gamma \vdash a \triangleright \theta\} = \emptyset$, then $\Gamma^{\mathtt{t}}(a) = \bigcup_{(F,\theta') \in \mathcal{S}} \{\theta \mid \{F \triangleright (\theta', m_{max})^{\mathtt{t}}\} \vdash a \triangleright \theta\} = \{\theta' \mid (F, \theta') \in \mathcal{S}\} = [\![F]\!]$.

If $t = t_1\ t_2$, take $\theta \in \Gamma^{\mathtt{t}}(t)$. There exists $\Gamma \subseteq \Gamma^{\mathtt{t}}$ such that $\Gamma \vdash t \triangleright \theta$. Therefore there exists $\Gamma_0, \Gamma_1, .., \Gamma_k$, $\theta_1, ..., \theta_k$, $m_1, ..., m_k$ such that

- $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$,
- for all $i \geq 1$, $\Gamma_i \uparrow m_i \vdash t_1 \triangleright \theta_i$,
- $\Gamma_0 \cup \Gamma_1 \uplus ... \cup \Gamma_k = \Gamma$.

From $\Gamma_0 \cup \Gamma_1 \uplus ... \cup \Gamma_k = \Gamma$ we get that all Γ_i are subsets of $\Gamma^{\mathtt{t}}$. Then from $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ we get that $(\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta \in \Gamma^{\mathtt{t}}(t_0)$, then by induction hypothesis $(\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta \in [\![t_0]\!]$. Using Lemma 13 we have that for all i $\Gamma_i \uparrow m_i = \Gamma_i$, then $\Gamma_i \vdash t_1 \triangleright \theta_i$ hence $\theta_i \in \Gamma^{\mathtt{t}}(t_0)$ therefore by induction hypothesis $\theta_i \in [\![t_0]\!]$. According to the definition of $[\![\cdot]\!]$ we have $\theta \in [\![t_0]\!]\ semt_1 = [\![t_0\ t_1]\!]$.

On the other hand, if $\theta \in [\![t_0\ t_1]\!] = [\![t_0]\!]\ semt_1$, then there exist θ_1,...,θ_k and m_1,...,m_k such that $(\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta \in [\![t_0]\!]$ and for all i $\theta_i \in [\![t_1]\!]$. Then by induction hypothesis there exists $\Gamma_0, .., \Gamma_k \subseteq \Gamma^{\mathsf{t}}$ such that $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ and for all $i \geq 1$ $\Gamma_i \vdash t_1 \triangleright \theta_i$. Then again, using Lemma 13 we have that for all i $\Gamma_i \uparrow m_i = \Gamma_i$, then for all $i \geq 1$ $\Gamma_i \uparrow m_i \vdash t_1 \triangleright \theta_i$. Therefore $\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \vdash t_1\ t_2 \triangleright \theta$, and since $\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \subseteq \Gamma^{\mathsf{t}}$, $\theta \in \Gamma^{\mathsf{t}}(t_1\ t_2)$. ◀

▶ **Theorem 19.** *Given two terms t, t', if $t \to t'$ then $[\![t]\!]_{\Gamma^t} = [\![t']\!]_{\Gamma^t}$.*

Proof. Since $t \to t'$, there exist two terms $r, r' : o$ and a context $C[\bullet]$ such that $r \to r'$, therefore we have $\|G_r\| = \|G_{r'}\|$. Theorem 17 states that $[\![r]\!] = \{q \mid \mathcal{A} \text{ accepts } \|G_r\| \text{ from state } q\}$, then we have $[\![r]\!] = [\![r']\!]$, hence $[\![t]\!] = [\![C[r]]\!] = [\![C[r']]\!] = [\![C[t]]\!]$. ◀

B.4 MSO reflection

In order to prove MSO reflexivity, we use the following theorem from [3].

▶ **Theorem 20** (Broadbent, Carayol, Ong, Serre)**.** *Let R be a class of generators of trees. If R is reflective with respect to modal μ-calculus and with respect to regular paths, then it is also reflective with respect to* MSO*.*

We need to define what means the reflexivity with respect to regular paths. Given a regular language L in $(\Sigma \cdot \{1, ..., \text{arity}_{\max}\})^*$, a scheme $\mathcal{G}$ on Σ, one can reflect L on $\mathcal{G}$ if one can construct a scheme $\mathcal{G}'$ on $\Sigma \times \{0, 1\}$ such that $\|\mathcal{G}'\|$ is an S_L-marking (as defined in Section 3), S_L being the set of node $u = j_0 \cdot j_1 \cdot j_2 \cdot ... \cdot j_k$ such that the word $Path(u) = \|\mathcal{G}\|(\varepsilon) \cdot j_0 \cdot \|\mathcal{G}\|(j_0) \cdot j_1 \cdot \|\mathcal{G}\|(j_0 j_1) \cdot j_2 \cdot ... \cdot \|\mathcal{G}\|(j_0 ... j_{k-1}) \cdot j_k$ is in the language L.

▶ Remark. The construction of [3] uses the two reflexivity construction one after the other, and no other transformation, in particular if both reflexivity transformations keep the structure of the scheme, then the resulting MSO transformation does too.

▶ **Lemma 21.** *Higher order recursion schemes are reflective with respect to regular paths.*

B.5 Schemes are Reflective With Respect to Regular Paths, Proof of Lemma 21.

Take a regular language L in $(\Sigma \cdot \{1, ..., \text{arity}_{\max}\})^*$. There is a finite word automaton $\mathcal{B} = \langle \Sigma \uplus \{1, ..., \text{arity}_{\max}\}, Q, q_0, F \subseteq Q, \delta_{\mathcal{B}} \rangle$ that recognises exactly L. We construct a parity tree automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \Omega = \{q \mapsto 0 \mid q \in Q\} \rangle$ such that $Path(u) \in L$ if and only if there is a state $q_f \in F$ that can be reached by $\mathcal{A}$ on u. We define the transitions in δ as $q \xrightarrow{a} q_1, ..., q_k \in \delta$ if for all j there exists q' such that $q \xrightarrow{a} q' \in \delta_{\mathcal{B}}$ and $q' \xrightarrow{j} q_j \in \delta_{\mathcal{B}}$. A simple induction shows that $\mathcal{A}$ satisfies the property we wanted. Now we show that we can embed the reachable states of $\mathcal{A}$ in a scheme, which would conclude the proof, since we can always turn a terminal labelled by a set of state $a^{Q' \subseteq Q}$ into a nonterminal, and make it transforms into the terminal $(a, 0)$ if $Q' \cap F = \emptyset$ and $(a, 1)$ otherwise.

▶ **Lemma 22** (Reachable States Reflexion)**.** *Given an automaton $\mathcal{A}$ and a scheme $\mathcal{G}$ one can create a scheme $\mathcal{G}'$ on $\Sigma \times 2^Q$ such that the projection on the first component of $\|\mathcal{G}'\|$ is $\|\mathcal{G}\|$ and such that for all u, if $\|\mathcal{G}'\|(u) = (a, Q')$ then Q' is the set of reachable states at the node u.*

Proof. We use a similar construction as in Section 4.0.2. In the following let $N = Card(2^Q)$, and let $Q_1, ..., Q_N$ be an enumeration of 2^Q. We define inductively the type $\tau^+ = (\tau_1^+)^N \to ... \to (\tau_k^+)^N \to o$.

To a terminal $a : o^k \to o$ (resp. a nonterminal $F : \tau$ variable $x : \tau$)and a set of states $Q' \subseteq Q$, we associate the terminal $\overline{a}^{Q'} : (o^k \to o)^+ \in \Sigma'$ (resp. a nonterminal $F^{Q'} : \tau$ variable $x^{Q'} : \tau$).

Given a term $t : \tau \in \mathcal{T}(\mathcal{V}' \uplus \Sigma' \uplus \mathcal{N}')$ and a set $Q' \subseteq Q$, we define inductively the term $t^Q : \tau^+ \in \mathcal{T}(\mathcal{V}' \uplus \Sigma' \uplus \mathcal{N}')$ f. If $t = a \in \Sigma$ a symbol, $t^{Q'} = \overline{a}^{Q'}$, if $t = \alpha \in \Sigma \uplus \mathcal{V}$, $t^{Q'} = \alpha^{Q'}$, if $t = t_1\ t_2$ we define $(t_1\ t_2)^Q = {t_1}^Q\ {t_2}^{Q_1} ...\ {t_2}^{Q_N}$. Note that since this transformation is only duplicating and annotating, given a term $t^{Q'}$ we can uniquely find the unique term t associated to it.

Let $F : \tau_1 \to ... \to \tau_k \to o \in \mathcal{N}$, $Q' \subseteq Q$. If $F\ x_1 ... x_k\ \to\ e \in \mathcal{R}$, we define in $\mathcal{R}'$ the rule

$$F^Q\ x_1^{Q_1} ...\ x_1^{Q_N}\ ...\ \ x_k^{Q_1} ...\ x_k^{Q_N}\ \to\ e^Q.$$

Finally, We let $\mathcal{G}' = \langle \mathcal{V}', \Sigma', \mathcal{N}', \mathcal{R}', S^{\{q_0\}} \rangle$.

▶ **Lemma 23.** *Given two terms $t, t' : o \in \mathcal{T}(\Sigma \uplus \mathcal{N})$, if $t \Rightarrow t'$, then $t^Q \Rightarrow t'^Q$.*

Proof. The proof is entirely similar to the one of Section 4.0.2. ◀

Now we define the scheme $\mathcal{G}''$ in which symbols in Σ' are turned into non terminals and where the new set of terminals is $\Sigma \times 2^Q$. The rules associated to $\overline{a}^Q$ is :

$$\overline{a}^Q\ x_1^{Q_1} ...\ x_1^{Q_N}\ ...\ \ x_k^{Q_1} ...\ x_k^{Q_N}\ \to a^Q\ x_1^{Q_{i_1}} ... x_k^{Q_{i_k}}$$

with for all j, $Q_{i_j} = \{q_i \mid \exists q \in Q, q \xrightarrow{a} q_1, ..., q_i, ..., q_k\}$.

▶ **Lemma 24.** *Given a term $t : o$, let $t_\star^{\{q_0\}}$ be the term obtained by applying all redex whose head is in Σ' in $t^{\{q_0\}}$. We have that the projection of $(t_\star^{\{q_0\}})^\perp$ on its first component is equal to $t^\perp$, furthermore each node is labelled by the reachable sets by the automaton.*

Proof. A simple structural induction proves the result. ◀

Let $S = t_0 \Rightarrow t_1 \Rightarrow t_2 \Rightarrow ...$ be the parallel derivation in $\mathcal{G}$. We know that the limit tree of $t_0^\perp, t_1^\perp, t_2^\perp, ...$ is equal to $\|\mathcal{G}\|$, we get by the previous lemma that $\|\mathcal{G}\| \sqsubseteq \|\mathcal{G}''\|_{|1}$ ($\|\mathcal{G}''\|_{|1}$ denote the projection on the first component of $\|\mathcal{G}''\|$). Since we can assume without loss of generality that there are no occurrences of $\perp$ in $\|\mathcal{G}\|$, we get $\|\mathcal{G}\| = \|\mathcal{G}''\|_{|1}$, which concludes the proof. ◀

C Proof of Theorem 8: Automata Selection

C.1 The Construction

In the following we fix a scheme $\mathcal{G} = \langle \mathcal{V}, \Sigma, \mathcal{N}, S, \mathcal{R} \rangle$ and a non deterministic parity tree automaton $\mathcal{A}$ such that $\|\mathcal{G}\|$ is accepted by $\mathcal{A}$ from state q_0. For all vertex (F, θ, m) in the associated game, let $\Gamma_{(F,\theta,m)}$ be the choice of Eve in a positional winning strategy. We construct a scheme $\mathcal{G}' = \langle \mathcal{V}', \Sigma', \mathcal{N}', S^{(q_0,\Omega(q_0))}, \mathcal{R}' \rangle$ that would consists in annotated version of $\mathcal{G}$.

We write $\text{arity}_{\max}$ the maximum arity of $\Sigma \uplus \mathcal{N}$, $\text{order}_{\max}$ the maximum order, and $m_{\max}$ the maximum color of $\Omega(Q)$. We define $\text{Types}_{\mathcal{G}}$ as the finite set of types of order and arity less or equal than $\text{order}_{\max}$ and $\text{arity}_{\max}$.

Types

To a type τ we associate the integer

$$n_\tau = Card\,\{(\theta, m) \mid \forall i\ \theta : \tau, m \leq m_{\max}\}\,.$$

We define inductively the type $\tau^+ = (\tau_1^+)^{n_{\tau_1}} \to ... \to (\tau_k^+)^{n_{\tau_k}} \to o$. For example $(o^k \to o)^+ = o^{k\cdot|Q|\cdot m_{\max}} \to o$.

Symbols

We define

$$\Sigma' = \{a^{(q_1\to...\to q_k\to q,m)} : (o^k \to o)^+ \mid \quad a : o^k \to o \in \Sigma, q_1, ..., q_k, q \in Q, m \leq m_{\max}\},$$

$$\mathcal{V}' = \{v_{\tau,i} | \tau \in \text{Types}_{\mathcal{G}}, i \leq \text{arity}_{\max}\} \ \uplus \quad \{x^{(\theta,m)} : \tau^+ \mid x : \tau \in \mathcal{V}, \theta : \tau, m \leq m_{\max}\},$$

$$\mathcal{N}' = \{Void_\tau | \tau \in \text{Types}_{\mathcal{G}}\} \ \uplus \qquad \{F^{(\theta,m)} : \tau^+ \mid F : \tau \in \mathcal{N}, \theta : \tau, m \leq m_{\max}\}.$$

Terms

Given a term $e = \tau \in \mathcal{T}(\mathcal{V} \uplus \Sigma \uplus \mathcal{N})$, a mapping θ and an environment Γ such that $\Gamma \vdash e \triangleright \theta$ we define the term $e^{\Gamma,\theta} : \tau^+$ by induction on the structure of e.

- If $e = a \in \Sigma$, then $\Gamma = \emptyset$ and $\theta = q_1 \to ... \to q_k \to q \in Q$, we define $e^{\Gamma,q} = a^{(q_1\to...\to q_k\to q,\Omega(q))}$.
- If $e = F \in \mathcal{N}$, then $\Gamma = \{F \triangleright (\theta, m)^b\}$ for some m and b, we define $e^{\Gamma,\theta} = F^{(\theta,m)}$.
- If $e = x \in \mathcal{V}$, then $\Gamma = \{x \triangleright (\theta, m)^b\}$ for some m and b, we define $e^{\Gamma,\theta} = x^{(\theta,m)}$.
- If $e = e_1\ e_2$ with $e_2 : \tau_2$, then there exist (θ_1, m_1),...,(θ_k, m_k) and $\Gamma_0, \Gamma_1, ..., \Gamma_k$ such that $\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k = \Gamma$, $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ and for all i, $\Gamma_i \uparrow m_i \vdash t_1 \triangleright \theta_i$. We define

$$e^{\Gamma,\theta} = e_1^{\Gamma_0,(\theta_1,m_1)\wedge...\wedge(\theta_k,m_k)\to\theta}\ e_2^{\Gamma\uparrow m_1,\theta_1} ...\ e_2^{\Gamma\uparrow m_k,\theta_k} \qquad Void_{\tau_2}\ ...\ Void_{\tau_2}.$$

Here $Void_{\tau_2}$ is used to fill the missing arguments.

Rewrite Rules

Given a rewrite rule $F\ x_1...x_k \to e$ with $F : \tau_1 \to ... \to \tau_k \to o$ and a couple (θ, m) with

$$\theta = \bigwedge_i^{\ell_1}(\theta_{1,i}, m_{1,i}) \to ... \to \bigwedge_i^{\ell_k}(\theta_{k,i}, m_{k,i}) \to \theta'$$

such that (F, θ, m) is winning in the game. For all $1 \leq j \leq k$ we define the tuple $\vec{x_j} = (x_j^{(\theta_{j,1}, m_{j,1})}, ..., x_j^{(\theta_{j,\ell_j}, m_{j,\ell_j})}, v_{\tau_j,...}, ..., v_{\tau_j,...})$, that contains n_{τ_i} elements of type τ_i^+ (the first elements consists in x_j annotated with the couples $(\theta_{j,i}, m_{j,i})$, and the remaining of the tuple is filled with some variables $v_{\tau_j,i}$).

Let $\Gamma = \Gamma_{(F,\theta,m)}, x_1 \triangleright \bigwedge_{j\in I_1}(\theta_{1j}, m_{1j}), ..., x_k \triangleright \bigwedge_{j\in I_k}(\theta_{kj}, m_{kj})$ such that $\Gamma \vdash e \triangleright \theta$.

To the nonterminal $F^{(\theta,m)} \in \mathcal{N}'$ we associate the rewrite rule

$$F^{(\theta,m)}\ \vec{x_1}...\vec{x_k} \to e^{\Gamma,\theta}.$$

Notice that there is no use of variables $v_{\tau_j,i}$ in $e^{\Gamma,\theta}$ they are only used to match the correct type.

For other non terminals $H^{(\theta',m')} \in \mathcal{N}'$ such that (H, θ', m') is not winning in the game, we define the rule $H\ x_1...x_{k'} \to Void_o$ but as we will see, such non terminals will never appears in derivations starting from $S^{(q_0,\Omega(q_0))}$.

We also add the rule $Void_\tau\ x_1...x_{\text{arity}(\tau)} \to Void_o$.

Remarks

By definitions of $e^{\Gamma,\theta}$, given a reachable term $S \to^* t$, all terminal symbols occuring in t are of the form $a^{(q_1\to...\to q_k\to q,\Omega(q))}$ with $q \xrightarrow{a} q_1, ..., q_k \in \delta$.

A simple induction shows that in t, every fully applied terminal $a^{(q_1\to...\to q_k\to q,\Omega(q))}$ will have the structure:

$$a^{(q_1\to...\to q_k\to q,\Omega(q))}\vec{t_1}...\vec{t_k}$$

with

$$\vec{t_k} = \underbrace{t_k, Void_o, ..., Void_o}_{n_o \text{ arguments}}.$$

It means that every subtree of $\|\mathcal{G}'\|$ will have the structure

$$a^{(q_1\to...\to q_k\to q,\Omega(q))}\ \underbrace{t_1\ \bot...\bot}_{n_o \text{ arguments}}\ ...\ \underbrace{t_k\ \bot...\bot}_{n_o \text{ arguments}}.$$

Finally we define the scheme $\mathcal{G}''$ obtained from $\mathcal{G}'$ by turning the terminals $a^{(q_1\to...\to q_k\to q,m)}$ as non terminals with the rewrite rules

$$a^{(q_1\to...\to q_k\to q,m)}\ \vec{x}_1\ ...\ \vec{x}_k \to (a, q)\ x_1...x_k.$$

We state precisely the Automata Selection Theorem we want to prove in this section.

▶ **Theorem 25.** *The tree $\|\mathcal{G}''\|$ is an accepting run of $\mathcal{A}$ on $\|\mathcal{G}\|$.*

The proof of this theorem is based on the proof of the soundness of the construction of Kobayashi and Ong.

In the following we only consider the OI parallel derivation, which can be shown to lead to the value tree (using a well known λ-calculus theorem called the Standardisation Theorem).

Given a term $t = \alpha\ t_1 ... t_k$ with $\alpha \in \Sigma \uplus \mathcal{N}$, we define inductively the OI parallel rewriting $t^\star_{OI}$ of t:

- if $\alpha \in \Sigma$ or $k < \text{arity}(\alpha)$, then $t^\star = \alpha\ t_1^\star ... t_k^\star$,
- if $\alpha \in \mathcal{N}$ and $k = \text{arity}(\alpha)$, let $\alpha\ x_1 ... x_k \to e$ be the rewrite rule associated to α, we have $t^\star = e_{[\forall i\ x_i \mapsto t_i]}$.

Given t, t', we write $t \Rightarrow_{OI} t'$ the relation "$t' = t^\star$".

C.2 Occurrences

In the following we put some formalism on the notion of occurrence of some symbols in terms. We define a conservation relation and a creation relation that matches the ideas that after a rewriting $t_1 \to t_2$ such occurrence in t_2 is the same as such other occurrence in t_1, or that such occurrence in t_2 has been created during the rewriting $t_1 \to t_2$.

For example let's look at a term $t_1 = \mathtt{a}\ (F\ H)$ with the rewriting rule $F\ x \to J\ x\ x$. The term t_1 can be rewritten in $t_2 = \mathtt{a}\ (J\ H\ H)$. We want to formalise the fact that both occurrences of H in t_2 comes from the one in t_1 or the fact that the head $\mathtt{a}$ of t_2 is the same as the one in t_1. We also want to express the fact that the occurrence of J in t_2 has been created by the occurrence of F in t_1.

For technical reason, to each rewriting $t \to t'$ we associate a canonical context $C[\bullet]$ where the rewriting take place. This would avoid some ambiguity in the definitions.

▶ **Definition 26** (Occurrences). Given a set of typed symbols Γ and a term $t : \tau \in \mathcal{T}(\Gamma)$, an ***occurrence*** in t is a couple $(C[\bullet], \alpha)$ with $C[\bullet : \tau'] : \tau$ a context and $\alpha : \tau'$ a symbol in Γ such that $t = C[\bullet]$.

Notation. We may use subscript to denote occurrence of a symbol, e.g. α_i would denote an occurrence of α. When we do not want to precise the symbol, we use the notation *occ* or occ_i for some i. Given an occurrence *occ* we write $C_{occ}[\bullet]$ the associated context. We write Occ_t the set of occurrences in the term t.

▶ **Definition 27** (Conservation Relation). Given t_1, t_2 such that $t_1 \to t_2$ and let $C[\bullet]$ be the context associated to $t_1 \to t_2$, $F \in \Sigma$, $F\ x_1, ..., x_k \in \mathcal{R}$, and $s_1, ..., s_k$ be such that $t_1 = C[F\ s_1, ..., s_k]$ and $t_2 = C[e_{[\forall i\ x_1 \mapsto s_i]}]$. Given $\alpha \in \Sigma \uplus \mathcal{N}$ and α_1 (resp. α_2) an occurrence of α in t_1 (resp. in t_2). We say that α_1 ***gives*** α_2 in the rewriting $t \to t'$, written $\alpha_1 \gg_{t \to t'} \alpha_2$ (or simply $\alpha_1 \gg \alpha_2$ when the rewriting is clear from the context), if:

- Either there exist a two holes context $\mathbb{C}[\bullet_1][\bullet_2]$ such that

$$C_{\alpha_1}[\bullet] = \mathcal{C}[\bullet][F\ s_1, ..., s_k] \quad C_{\alpha_2}[\bullet] = \mathbb{C}[\bullet][e_{[\forall i\ x_1 \mapsto s_i]}] \quad C[\bullet] = \mathbb{C}[\alpha][\bullet].$$

- Or there exists j and a context $C'[\bullet]$ such that $s_j = C'[\alpha]$, $C_{\alpha_1}[\bullet] = C[F\ s_1 ... C'[\bullet] ... s_k]$ and there exist an occurrence $occ = (C_{occ}[\bullet], x_j)$ of x_j in e such that

$$C_{\alpha_2}[\bullet] = \Big(C_{occ}\big[C'[\bullet]\big]\Big)_{[\forall i\ x_1 \mapsto s_i]}$$

Remark. Given α_2, there exists at most one α_1 such that $\alpha_1 \gg \alpha_2$.

▶ **Definition 28** (Creation Relation). Given t_1, t_2 such that $t_1 \to t_2$ and let $C[\bullet]$ be the context associated to $t_1 \to t_2$, $F \in \Sigma$, $F\ x_1, ..., x_k \in \mathcal{R}$, and $s_1, ..., s_k$ be such that $t_1 = C[F\ s_1, ..., s_k]$ and $t_2 = C[e_{[\forall i\ x_1 \mapsto s_i]}]$. Given $\alpha \in \Sigma \uplus \mathcal{N}$ and α_2 (resp. α_2) an occurrence of α in t_2. Let F_1 be the occurrence $(C_{F_1} = C[\bullet\ s_1 ... s_k], F)$ of F in t_1, we say that α_1 ***creates*** α_2 in the rewriting $t \to t'$, written $F_1 \succ_{t \to t'} \alpha_2$ (or simply $F_1 \succ \alpha_2$ when the rewriting is clear from the context), if there exists a context $C'[\bullet]$ such that $e = C'[\alpha]$ and

$$C_{\alpha_2}[\bullet] = C\Big[\big(C'[\bullet]\big)_{[\forall i\ x_i \mapsto s_i]}\Big].$$

▶ **Proposition 29.** *Given $t_1 \to t_2$ and occ_2 an occurrence in t_2 there there exist a unique occurrence occ_1 in t_1 such that either $occ_1 \gg occ_2$ or $occ_1 \succ occ_2$ (but not both).*

Proof. Let $C[\bullet]$ be the context associated to $t_1 \to t_2$. The proof comes from the fact that for any occurrence α_2 in t_2, there are three disjoint possibilities.

1. Either there exist a two holes context $\mathbb{C}[\bullet_1][\bullet_2]$ such that

   $$C_{\alpha_2}[\bullet] = \mathbb{C}[\bullet][e_{[\forall i\ x_1 \mapsto s_i]}]$$

   and $C[\bullet] = \mathbb{C}[\alpha][\bullet]$, in which case there is an occurrence $(\mathbb{C}[\bullet][F\ s_1 ... s_k], \alpha)$ in t_1 that gives α_2.
2. Or there exist an occurrence $occ = (C_{occ}[\bullet], x_j)$ of x_j in e and a context $C'[\bullet]$ such that

   $$C_{\alpha_2}[\bullet] = \Big(C_{occ}\big[C'[\bullet]\big]\Big)_{[\forall i\ x_1 \mapsto s_i]}$$

   in which case there is an occurrence $(C[F\ s_1 ... C'[\bullet] ... s_k], \alpha)$ in t_1 that gives α_2.
3. Or there exists a context $C'[\bullet]$ such that $e = C'[\alpha]$ and

   $$C_{\alpha_2}[\bullet] = C\Big[\big(C'[\bullet]\big)_{[\forall i\ x_i \mapsto s_i]}\Big]$$

   in which case the occurrence $(C[\bullet s_1 ... s_k], F)$ creates α_2.

◀

Now we want to look at occurrences of terms inside a derivation. To specify the term from which comes an occurrence we say that an occurrence in a derivation is an occurrence of a term inside the derivation along with the index of the term. This allows to avoid ambiguity if two terms in the derivation are the same.

In the following we let $d = t_0 \to t_1 \to t_2 \to ...$ be a derivation.

▶ **Definition 30** (Occurrences in derivation)**.** We define the set Occ_d of occurrences in der as:

$$Occ_d = \{(i, occ_i) \mid occ_i \in t_i\}$$

▶ **Definition 31** (Conservation relation in Occ_d)**.** We define the relation $\gg_d$ (or simply $\gg$ when the derivation is clear from the context) on $Occ_d \times Occ_d$ as the smallest reflexive and transitive relation satisfying that if occ_i and occ_{i+1} are occurrences of respectively t_i and t_{i+1} such that $occ_i \gg_{t_i \to t_{i+1}} occ_{i+1}$ then $(i, occ_i) \gg_{der} (i+1, occ_{i+1})$.

▶ **Definition 32** (Creation relation in Occ_d)**.** We define the relation $\succ_d$ (or simply $\succ$ when the derivation is clear from the context) on $Occ_d \times Occ_d$ as $(i, occ_i) \succ_d (j, occ_j)$ if occ_i (resp. occ_j) is an occurrence of t_i (resp. t_j), and if there exists an occurrence occ_{i+1} in t_{i+1} such that $occ_i \succ_{t_i \to t_{i+1}} occ_{i+1}$ and $(i+1, occ_{i+1}) \gg_d (j, occ_j)$.

C.3 Definition of $(\Sigma \times Q)$-contexts

▶ **Definition 33** ($(\Sigma \times Q)$-context)**.** A $(\Sigma \times Q)$-context is a context $C[\bullet : o] : o$ such that either $C[\bullet] = \bullet$ or

$$C[\bullet] = (a, q)\ t_1\ ...\ C'[\bullet]\ ...\ t_k$$

with $(a, q) \in \Sigma \times Q$ and $C'[\bullet]$ a Σ-context.

We associate to $C[\bullet]$ a finite sequence $b_{C[\bullet]}$ of $(\Sigma \times Q \cdot \mathbb{N})^*$ as if $C[\bullet] = \bullet$, $b_{C[\bullet]} = \varepsilon$, if

$$C[\bullet] = (a, q)\ t_1\ ...t_{j-1}\ C'[\bullet]\ t_{j+1}\ ...t_k$$

then

$$b_{C[\bullet]} = (a, q), j, b_{C'[\bullet]}.$$

▶ **Definition 34** (Color of a sequence b)**.** Given a finite sequence b of $(\Sigma \times Q \cdot \mathbb{N})^*$ and a state q (which represent the state of the node b is pointing on), we define the color of (b, q) by induction on b: if $b = \varepsilon$ then $\Omega(b, q) = q$, if $b = (a_0, q_0)jb'$ then $\Omega(b, q) = \max(q_0, \Omega(b', q))$.

▶ **Definition 35** (Approximation)**.** Given a tree over $\Sigma \times Q$ and an infinite branch $b = (a_0, q_0), j_0, (a_1, q_1), j_1, ...$ We say that a $(\Sigma \times Q)$-context $C[\bullet]$ ***approximates*** b if $b_{C[\bullet]}$ is a prefix of b.

C.4 Correction of the Labelling of Non Terminals

▶ **Lemma 36.** *Given a reachable term $S^{(q_0,\Omega(q_0))} \to^* t$, all non terminal $F^{(\theta,m)}$ occurring in t satisfies that (F, θ, m) is winning in the game.*

Proof. First given a rewrite rule $H^{(\theta',m')}x_1...x_k \to e$ with (H, θ', m') winning in the game, we prove by induction on the structure of e that all non terminal $F^{(\theta,m)}$ occurring in e satisfies that (F, θ, m) is winning in the game.

Then a simple induction on the size of the derivation $S^{(q_0,\Omega(q_0))} \to^* t$ proves the lemma. ◀

▶ **Lemma 37.** *Given a derivation $S^{(q_0,\Omega(q_0))} = t_0 \to t_1 \to ...$ and two occurrences of non terminals $(F_i^{(\theta,m)}, i)$ and $(H_j^{(\theta',m')}, j)$ with $i < j$. If $(F_i^{(\theta,m)}, i) \succ (H_j^{(\theta',m')}, j)$, then $H \triangleright (\theta', m')^f \in \Gamma_{(F,\theta,m)}$.*

Proof. Same kind of simple proof as the previous lemma. ◀

C.5 Main Proof

▶ **Lemma 38.** *Given a derivation $S^{(q_0,\Omega(q_0))} = t_0 \to t_1 \to ...$, two indexes $i < j$, two occurrences $H_i^{(\theta_0,m_0)}$ in t_i and $F_j^{(\theta,m)}$ in t_j such that:*

- $(H_i^{(\theta_0,m_0)}, i) \succ (F_j^{(\theta,m)}, j)$,
- *There exists two Σ-contexts $C_1[\bullet]$ and $C_2[\bullet]$ such that:*

$$C[\bullet]_{H^{(\theta_0,m_0)}} = C_1[\bullet\ s_1...s_{arity(H^{(\theta_0,m_0)})}],$$

$$C[\bullet]_{F^{(\theta,m)}} = C_2[\bullet\ t_1...t_{arity(F^{(\theta,m)})}],$$

- $b_{C_2[\bullet]} = b_{C_1[\bullet]} \cdot b'$.

Then $\Omega(b', q) = m$ with $\theta = \bigwedge_i(\theta_{1,i}, m_{1,i}) \to ... \to \bigwedge_i(\theta_{k,i}, m_{k,i}) \to q$.

▶ **Theorem 39** (Kobayashi, Ong)**.** *Given an infinite derivation $S^{(q_0,\Omega(q_0))} = t_0 \to t_1 \to ...$ and an infinite branch b in the limit tree of this derivation. there exist an infinite strictly increasing sequence of indexes $i_0 = 0 < i_1 < i_2 < ...$ and an infinite sequence of occurrences $(occ_0, i_0), (occ_1, i_1), ...$ such that:*

- *for all j, $(occ_j, i_j) \succ (occ_{j+1}, i_{j+1})$,*
- *for all j $C_{occ_j}[\bullet] = C_j[\bullet\ s_1...s_{arity(\bullet)}]$ for some Σ-context $C_j[\bullet]$and terms $s_1, ..., s_{arity(\bullet)}$,*
- *for all j, let $b_j = b_{C_j[\bullet]}$, then b_j is a prefix of b,*
- *the sequence $b_0, b_1, b_2, ...$ is increasing and its limit is b.*
- *The greatest colour appearing infinitely often in b is equal to the greatest colour appearing infinitely often in $\Omega(b'_0), \Omega(b'_1), \Omega(b'_2), ...$ where for all i, $b_{i+1} = b_i \cdot b'_i$.*

Proof. See [18]. ◀

Now we can prove the main theorem.

Proof of Theorem 25. Proving that the value tree is indeed a run of the automaton is easy and similar to the proof of Theorem 4, so we focus on proving that it is an accepting run. Take a derivation leading to $\|\mathcal{G}''\|$ and an infinite branch b in $\|\mathcal{G}''\|$. Take an infinite sequence of occurrences $(occ_0, i_0), (occ_1, i_1), ...$ defined as in Theorem 39. We know that the maximum colour seen infinitely often in b is the greatest colour appearing infinitely often in $\Omega(b'_0), \Omega(b'_1), \Omega(b'_2), ...$ Let $F_i^{(\theta_i,m_i)}$ be the non terminal associated to the occurrence occ_i. By definition, $F_{i+1} \triangleright \{\theta_{i+1}, m_{i+1}\}^{\mathtt{f}}$ is in the environment picked by Eve from (F_i, θ_i, m_i), then, since Eve wins in the game when she respects her strategy, the infinite colour seen infinitely often in $m_0, m_1, ...$ is even. Since for all i, $m_i = \Omega(b'_i)$, it means that the greatest colour seen infinitely often along b is even, which concludes the proof. ◀

C.6 Substitution Lemma

The following section consists in a proof of the following proposition.

▶ **Proposition 40** (Substitution). *Given* $e \in \mathcal{T}(\Sigma \uplus \mathcal{N} \uplus \mathcal{V})$, $x : \tau \in \mathcal{V}$, $t : \tau \in \mathcal{T}(\Sigma \uplus \mathcal{N} \uplus \mathcal{V})$, *if*

$$\Gamma_x = \Gamma_0, x \triangleright (\theta_1, m_1)^{\mathtt{f}}, ..., x \triangleright (\theta_k, m_k)^{\mathtt{f}} \vdash e \triangleright \theta,$$

$$\forall i \ \Gamma'_i = \Gamma_i \uparrow m_i \vdash t \triangleright \theta_i$$

then

$$\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \vdash e_{[x \mapsto t]} \triangleright \theta,$$

and if we let $\Gamma = \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k$ *we have*

$$\left(e^{\Gamma_x, \theta}\right)_{\left[\forall i \ x^{(\theta_i, m_i)} \mapsto t^{\Gamma'_i, \theta_i}\right]} = (e_{[x \mapsto t]})^{\Gamma, \theta}$$

▶ **Lemma 41.** *If* $\Gamma \uparrow m$ *is well-defined, and if* $\Gamma \vdash t \triangleright \theta$ *holds then* $\Gamma \uparrow m \vdash t : \theta$ *and* $t^{\Gamma, \theta} = t^{\Gamma \uparrow m, \theta}$.

Proof. Straightforward induction on the derivation of $\Gamma \vdash t \triangleright \theta$. ◀

▶ **Lemma 42.** *If* $\Gamma \vdash t \triangleright \theta$ *then* $\Gamma \uparrow \Omega(\theta)$ *is well defined. Furthermore, if* $\Gamma \uparrow \Omega(\theta) = \Gamma' \uparrow \Omega(\theta)$ *then* $\Gamma' \vdash t \triangleright \theta$ *and* $t^{\Gamma', \theta} = t^{\Gamma, \theta}$.

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash t \triangleright \theta$.

- If the last rule applied is (T-Var), then $t = \alpha \in \mathcal{V} \uplus \mathcal{N}$ and $\Gamma = \alpha \triangleright (\theta, m)^b$ with $(\theta, m)^b \uparrow \Omega(\theta) = (\theta, m)^{\mathtt{t}}$. Thus $\Gamma \uparrow \Omega(\theta)$ is well defined. If $\Gamma \uparrow \Omega(\theta) = \Gamma' \uparrow \Omega(\theta)$ then $\Gamma' = \alpha \triangleright (\theta, m)^{b'}$ and $(\theta, m)^b \uparrow \Omega(\theta) = (\theta, m)^{\mathtt{t}}$ therefore $\Gamma' \vdash t \triangleright \theta$. We also have that $t^{\Gamma', \theta} = \alpha^{(\theta, m)} = t^{\Gamma, \theta}$.
- If the last rule applied is (T-Const) then $t = a \in \Sigma$, $\Gamma = \emptyset$ and the result is trivial.
- If the last rule applied is (T-App) then $t = t_0 \ t_1$ $\Gamma = \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k$ with $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ and for all i, $\Gamma_i \uparrow m_i \vdash t_i \triangleright \theta_i$. By induction hypothesis $\Gamma_0 \uparrow \theta$ is well defined. Furthermore by the well formedness of $(\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$ we have for all $m_i \geq \Omega(\theta)$ for all i then since $\Gamma_i \uparrow m_i$ is well defined, we have that $\Gamma_i \uparrow \Omega(\theta)$ is well defined, therefor $\Gamma \uparrow \Omega(\theta) = \Gamma_0 \uparrow \Omega(\theta) \cup \Gamma_1 \uparrow \Omega(\theta) \cup ... \cup \Gamma_k \uparrow \Omega(\theta)$ is well defined.
  If $\Gamma \uparrow \Omega(\theta) = \Gamma' \uparrow \Omega(\theta)$ then there exists $\Gamma'_0 \cup \Gamma'_1 \cup ... \cup \Gamma'_k = \Gamma'$ such that for all i $\Gamma'_i \uparrow \Omega(\theta) = \Gamma_i \uparrow \Omega(\theta)$. Since $m_i \geq \Omega(\theta)$ we have for all $i > 0$ $\Gamma_i \uparrow m_i = \Gamma_i \uparrow m_i \uparrow \Omega(\theta) = \Gamma_i \uparrow \Omega(\theta) \uparrow m_i = \Gamma'_i \uparrow \Omega(\theta) \uparrow m_i = \Gamma'_i \uparrow m_i \uparrow \Omega(\theta) = \Gamma'_i \uparrow m_i$ thus $\Gamma'_i \uparrow m_i \vdash t_1 \triangleright \theta_i$.
  Furthermore by induction hypothesis, since $\Omega((\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta) = \Omega(\theta)$ we have $\Gamma'_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta$, hence $\Gamma' \vdash t \triangleright \theta$.
  Also by induction hypothesis,

  $$t_0^{\Gamma'_0, (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta} = t_0^{\Gamma_0, (\theta_1, m_1) \wedge ... \wedge (\theta_k, m_k) \to \theta}$$

  and since $\Gamma_i \uparrow m_i = \Gamma'_i \uparrow m_i$, for all i, $t_1^{\Gamma'_i \uparrow m_i, \theta_i} = t_1^{\Gamma_i \uparrow m_i, \theta_i}$ then by definition $t^{\Gamma, \theta} = t^{\Gamma', \theta}$.

◀

Given $b \in \{\mathtt{f}, \mathtt{t}\}$, we define $\Gamma \uparrow_b m$ by:

$$\Gamma \uparrow_b m = \begin{cases} \Gamma \uparrow m & \text{if } b = \mathtt{f} \\ \Gamma & \text{if } b = \mathtt{t} \text{ and } \Gamma \uparrow m \text{ is well defined.} \end{cases}$$

To conclude, we prove the following Lemma which is a generalisation of Proposition 40.

▶ **Lemma 43** (Generalised Substitution). *Given* $e \in \mathcal{T}(\Sigma \uplus \mathcal{N} \uplus \mathcal{V})$, $x : \tau \in \mathcal{V}$, $t : \tau \in \mathcal{T}(\Sigma \uplus \mathcal{N} \uplus \mathcal{V})$, *if*

$$\Gamma_0, x \triangleright (\theta_1, m_1)^b_1, ..., x \triangleright (\theta_k, m_k)^b_k \vdash e \triangleright \theta,$$

$$\forall i \ \Gamma_i \uparrow_{b_i} m_i \vdash t \triangleright \theta_i$$

then

$$\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \vdash e_{[x \mapsto t]} \triangleright \theta,$$

and if we let $\Gamma_x = \Gamma_0, x \triangleright (\theta_1, m_1)^{\mathtt{f}}, ..., x \triangleright (\theta_k, m_k)^{\mathtt{f}}$, $\Gamma = \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k$ *and for all* i, $\Gamma'_i = \Gamma_i \uparrow_{b_i} m_i$ *then*

$$\left(e^{\Gamma_x, \theta}\right)_{\left[\forall i \ x^{(\theta_i, m_i)} \mapsto t^{\Gamma'_i, \theta_i}\right]} = (e_{[x \mapsto t]})^{\Gamma, \theta}$$

Proof. The proof proceeds by induction on the derivation of $\Gamma_0, x \triangleright (\theta_1, m_1)^b_1, ..., x \triangleright (\theta_k, m_k)^b_k \vdash e \triangleright \theta$.

- If the last rule applied is (T-Const), the result is trivial.
- If the last rule applied is (T-Var), and $e = \alpha \neq x$ the result is trivial.
- If the last rule applied is (T-Var), and $e = x$, then $\Gamma_0 = \emptyset$, $k = 1$, $\theta = \theta_1$, $(\theta_1, m_1)^b_1 \uparrow \Omega(\theta) = (\theta_1, m_1)^{\mathtt{t}}$, and $\Gamma_1 \uparrow_{b_1} m_1 \vdash t \triangleright \theta_1$.
  If $b_1 = \mathtt{t}$ then $\Gamma_1 \uparrow_{b_1} m_1 = \Gamma$ therefore $\Gamma_1 \vdash t \triangleright \theta$. Furthermore $e^{\{(\theta_1, m_1)^{b_1}\}, \theta} = x^{(\theta_1, m_1)}$ therefore

  $$\left(e^{\{(\theta_1, m_1)^{b_1}\}, \theta}\right)_{\left[x^{(\theta_1, m_1)} \mapsto t^{\Gamma_1 \uparrow m_1, \theta_1}\right]} = t^{\Gamma_1, \theta} = (e_{[x \mapsto t]})^{\Gamma_1, \theta}.$$

  If $b_1 = \mathtt{f}$ then since $(\theta_1, m_1)^b_1 \uparrow \Omega(\theta) = (\theta_1, m_1)^{\mathtt{t}}$, $m_1 = \Omega(\theta)$. Then since $\Gamma_1 \uparrow_{b_1} m_1 \vdash t \triangleright \theta_1$ and $\Gamma_1 \uparrow_{b_1} m_1 = \Gamma_1 \uparrow \Omega(\theta_1)$, Lemma 42 states that $\Gamma_1 \vdash t \triangleright \theta_1$. It also states that $t^{\Gamma_1 \uparrow \Omega(\theta), \theta} = t^{\Gamma_1, \theta}$ then

  $$\left(e^{\{(\theta_1, m_1)^{b_1}\}, \theta}\right)_{\left[x^{(\theta_1, m_1)} \mapsto t^{\Gamma_1 \uparrow m_1, \theta_1}\right]} = t^{\Gamma_1 \uparrow \Omega(\theta), \theta} = t^{\Gamma_1, \theta} = (e_{[x \mapsto t]})^{\Gamma_1, \theta}.$$

- If the last rule applied is (T-App), we have:

  $$e = e_0 \ e_1$$

  $$\Gamma_0 = \Delta_0 \cup \Delta_1 \cup ... \cup \Delta_\ell$$

  $$S_0 \cup S_1 \cup ... \cup S_\ell = \{1, ..., k\}$$

  $$\theta' = (\theta'_1, m'_1) \wedge ... \wedge (\theta'_\ell, m'_\ell) \to \theta$$

  $$\Delta^x_0 = \Delta_0 \cup \{x \triangleright (\theta_i, m_i)^{b_i} \mid i \in S_0\} \vdash e_0 \triangleright \theta'$$

  $$\Delta^x_j = \left(\Delta_j \cup \{x \triangleright (\theta_i, m_i)^{b_i} \mid i \in S_j\}\right) \uparrow m'_j \vdash e_1 \triangleright \theta'_j$$

  $$\Gamma_i \uparrow_{b_i} m_i \vdash t \triangleright \theta_i$$

From the induction hypothesis,we have

$$\Delta'_0 = \Delta_0 \cup \bigcup_{i \in S_0} \Gamma_i \vdash e_0 \triangleright (\theta'_1, m'_1) \wedge ... \wedge (\theta'_\ell, m'_\ell) \to \theta,$$

$$\left(e_0^{\Delta_0^x, \theta'}\right)_{\left[\forall i \in S_0 \ x^{(\theta_i, m_i)} \mapsto t^{\Gamma_i \uparrow_{b_i}, \theta_i}\right]} = (e_{0[x \mapsto t]})^{\Delta'_0, \theta'}.$$

Now we want to show that for all j,

$$\Delta'_j = \left(\Delta_j \cup \bigcup_{i \in S_j} \Gamma_i\right) \uparrow m'_j \vdash e_{1[x \mapsto t]} \triangleright \theta'_j,$$

and

$$\left(e_0^{\Delta_j^x, \theta'_j}\right)_{\left[\forall i \in S_j \ x^{(\theta_i, m_i)} \mapsto t^{\Gamma_i \uparrow_{b_i}, \theta_i}\right]} = (e_{1[x \mapsto t]})^{\Delta'_j, \theta'_j},$$

which by simple application would conclude.
For $i \in S_j$, we define $(\theta_i, m_i)^{b_{i,j}} = (\theta_i, m_i)^{b_i} \uparrow m'_j$. In particular we have:

(1) $\Delta_j \uparrow m'_j \cup \{x \triangleright (\theta_i, m_i)^{b_{i,j}} \mid i \in S_j\} = \Delta_j^x \vdash e_1 \triangleright \theta'_j$.

Since $(\theta_i, m_i)^{b_i} \uparrow m'_j$ is well define, we have $m'_j \leq m_i$, and since $\Gamma_i \uparrow_{b_i} m_i$ is well define, we get that m'_j is less or equal than any color in Γ_i.
Furthermore since $\Gamma_i \uparrow_{b_i} m_i \vdash t \triangleright \theta_i$, Lemma 41 states that $\Gamma_i \uparrow_{b_i} m_i \uparrow m'_j = \Gamma_i \uparrow m'_j \uparrow_{b_i} m_i \vdash t \triangleright \theta_i$ (indeed, one can commute the $\uparrow$), and $t^{\Gamma_i \uparrow_{b_i}, \theta_i} = t^{\Gamma_i \uparrow m'_j \uparrow_{b_i} m_i, \theta_i}$.
Notice that $\Gamma_i \uparrow m'_j \uparrow_{b_i} m_i = \Gamma_i \uparrow m'_j \uparrow_{b_{i,j}} m_i$, then

(2) $\Gamma_i \uparrow m'_j \uparrow_{b_{i,j}} m_i \vdash t \triangleright \theta_i$,

and

(3) $t^{\Gamma_i \uparrow_{b_i}, \theta_i} = t^{\Gamma_i \uparrow m'_j \uparrow_{b_i} m_i, \theta_i} = t^{\Gamma_i \uparrow m'_j \uparrow_{b_{i,j}} m_i, \theta_i}$.

Then by induction hypothesis on (1) and (2), and using (3), one can state that

$$\Delta'_j = \left(\Delta_j \cup \bigcup_{i \in S_j} \Gamma_i\right) \uparrow m'_j \vdash e_{1[x \mapsto t]} \triangleright \theta'_j,$$

and

$$\left(e_0^{\Delta_j^x, \theta'_j}\right)_{\left[\forall i \in S_j \ x^{(\theta_i, m_i)} \mapsto t^{\Gamma_i \uparrow_{b_i}, \theta_i}\right]} = (e_{1[x \mapsto t]})^{\Delta'_j, \theta'_j}.$$

◀

C.7 Proof of Lemma 38

Take the infinite OI parallel derivation in the initial scheme $S = t_0 \Rightarrow t_1 \Rightarrow ....$

▶ **Definition 44** (Painting of a term)**.** Given a term t a painting of t is a mapping that associates to every Σ-context $C[\bullet]$ such that there exist a term $t_{C[\bullet]} : o$ satisfying $C[t_{C[\bullet]}] = t$, a state $q_{i,C[\bullet]}$, and if $t_{C[\bullet]}$ is a redex an environment $\Gamma_{C[\bullet]}$ such that:

- If t is a redex, $\Gamma_{C[\bullet]} \subseteq \{F \triangleright (\theta, m)^b \mid (F, \theta, m)$ winning$, b \in \{\mathtt{f}, \mathtt{t}\}\}$,
- if t is a redex, $\Gamma_{C[\bullet]} \vdash t_{C[\bullet]} \triangleright q$ for some $q_{C[\bullet]} \in Q$,
- If $t_{C[\bullet]} = a\ s_1...s_k$ let $q = q_{C[\bullet]}$ and for all j, let $q_j = q_{C[a\ s_1...s_{j-1} \bullet s_{j+1}...s_k]}$, we have

  $$q \xrightarrow{a} (q_1, ..., q_k) \in \delta.$$

  In the following we define a painting of all terms of the derivation.

▶ **Claim 45.** *Given a term $t : o$ if we have $\Gamma \vdash t \triangleright q$ with*

$$\Gamma \subseteq \{F \triangleright (\theta, m)^b \mid (F, \theta, m)\ \textit{winning}, b \in \{\mathtt{f}, \mathtt{t}\}\}$$

then we can construct a painting of t.

Proof. Simple induction on the derivation. ◀

▶ **Proposition 46.** *If we have a painting of a redex $red = F\ s_1...s_k$, with $F\ x_1...x_k \to e$, then we can construct a painting of $e_{[\forall i\ x_i \mapsto s_i]}$.*

Proof. Take $\Gamma \vdash F\ s_1...s_k \triangleright q_\bullet$.

We know that there exists

$$\theta = \bigwedge_{p \in J_1} (\theta_{1,p}, m_{1,p}) \to ... \to \bigwedge_{p \in J_k} (\theta_{k,p}, m_{k,p}) \to q,$$

$(\theta, m)^b$ and $\Gamma_{j,p}$ for all j and $p \in J_j$ such that

$$\{F \triangleright (\theta, m)^b\} \vdash F \triangleright \theta,$$

for all j and $p \in J_j$

$$\Gamma_{j,p} \uparrow m_{j,p} \vdash s_j \triangleright \theta_{j,p},$$

and

$$\Gamma = \{F \triangleright (\theta, m)^b\} \cup \bigcup_{j,p} \Gamma_{j,p}.$$

We have (F, θ, m) wining in the game, then for all j there exists $I_j \subseteq J_j$ such that

$$\Gamma_0 = \Gamma_{(F,\theta,m)} \cup \{x_j \triangleright (\theta_{1,p}, m_{1,p})^{\mathtt{f}} \mid p \in I_j\} \vdash e \triangleright q.$$

Then using substitution property we know that

$$\Gamma' = \Gamma_0 \cup \bigcup_{j, p \in I_j} \Gamma_{j,p} \vdash e_{[\forall j\ x_j \mapsto s_j]} \triangleright q_\bullet,$$

And the previous claim concludes. ◀

▶ **Proposition 47.** *If we have a painting of t, we can create a painting of t' with $t \Rightarrow t'$.*

Proof. By induction, if t is a redex then we get the painting from the previous Proposition. If $t = a\ t_1...t_k$ then $t' = a\ t'_1...t'_k$ we define $q_\bullet$ in t' as $q_\bullet$ in t and the rest of the painting by induction. ◀

▶ **Proposition 48.** *We construct the painting of t_i for all i in the parallel derivation.*

Proof. For t_0 we define the painting of $\bullet$ as $\{S \triangleright (q_0, \Omega(q_0))\} \vdash S \triangleright q_0$ and we define the other paintings by induction on i. ◀

▶ **Definition 49** (Term Transformation)**.** Given a term $t : o$ painted by π. We define the term t^{++} in $\mathcal{G}'$ by induction. If t is a redex then $t^{++} = t^{\Gamma,q}$ with $\pi(\bullet) = (\Gamma, q)$. If $t = a\ t_1...t_k$ then

$$t^{++} = a^{(q_1,m_1)\to...\to(q_k,m_k)\to q}\ \vec{t_1^{++}}...\vec{t_k^{++}},$$

with q the painting of $\bullet$ and q_i the painting of $a\ t_1...t_{i-1}\bullet t_{i+1}...t_k$, and $\vec{t_1^{++}} = t_1^{++}Void_o...Void_o$.

Given a term $t : o$ painted by π. We define the term t^{+} in $\mathcal{G}''$ by induction. If t is a redex then $t^{+} = t^{\Gamma,q}$ with $\pi(\bullet) = (\Gamma, q)$. If $t = a\ t_1...t_k$ then $t^{+} = (a, q)\ t_1^{+}...t_k^{+}$.

▶ **Proposition 50.** *Let $t = F\ s_1...s_k$ and a painting π of t, with $F\ x_1...x_k \to e$, and $t' = e_{[\forall i\ x_i \mapsto s_i]}$, then $t'^{++} = t'^{\Gamma',q}$ with Γ' defined as in Proposition 46.*

Proof. A simple induction. ◀

▶ **Definition 51** (The rewritting $\Rightarrow_\Sigma$)**.** Given a term t in $\mathcal{G}''$ we define the term $t \Rightarrow_\Sigma t'$ as the one obtained by rewriting at once all the a^θ.

▶ **Proposition 52.** *Given a term t, then $t^{++} \Rightarrow_\Sigma t^{+}$.*

Proof. A simple induction ◀

▶ **Definition 53** (Canonic rewriting $\Rightarrow_+$)**.** Given a term t in $\mathcal{G}''$ we define its canonic rewriting t' by $t \Rightarrow_{OI} t'' \Rightarrow_\Sigma t'$.

▶ **Proposition 54.** *Given a term $t : o$ painted by π. If $t \Rightarrow t'$ then $t^{+} \Rightarrow_+ (t')^{+}$.*

Proof. We just need to show the case where t is a redex. Take $t = F\ s_1...s_k$ and $\Gamma \vdash F\ t_1...t_k \to q$ with $\Gamma \subseteq \{H \triangleright (\theta, m)^b \mid (H, \theta, m)$ winning$\}$. We know that there exists

$$\theta = \bigwedge_{p\in J_1} (\theta_{1,p}, m_{1,p}) \to ... \to \bigwedge_{p\in J_k} (\theta_{k,p}, m_{k,p}) \to q,$$

$(\theta, m)^b$ and $\Gamma_{j,p}$ for all j and $p \in J_j$ such that

$$\{F \triangleright (\theta, m)^b\} \vdash F \triangleright \theta,$$

for all j and $p \in J_j$

$$\Gamma_{j,p} \uparrow m_{j,p} \vdash s_j \triangleright \theta_{j,p},$$

and

$$\Gamma = \{F \triangleright (\theta, m)^b\} \cup \bigcup_{j,p} \Gamma_{j,p}.$$

We have (F, θ, m) wining in the game, then for all j there exists $I_j \subseteq J_j$ such that

$\Gamma_0 = \Gamma_{(F,\theta,m)} \cup \{x_j \triangleright (\theta_{j,p}, m_{j,p})^{\mathtt{f}} \mid p \in I_j\} \vdash e \triangleright q.$

Then using substitution property we know that

$\Gamma' = \Gamma_{(F,\theta,m)} \cup \bigcup_{j,p\in I_j} \Gamma_{j,p} \vdash e_{[\forall j\ x_j \mapsto s_j]} \triangleright q_\bullet,$

and

$\left(e^{\Gamma_0,\theta}\right)_{\left[\forall j,p\in I_j\ x_j^{(\theta_{j,p},m_{j,p})} \mapsto t^{\Gamma'_{j,p},\theta_{j,p}}\right]} = (e_{[x\mapsto t]})^{\Gamma',\theta}.$

Since

$(t')^{++} = (e_{[x\mapsto t]})^{\Gamma',\theta}$

and

$t^+ \Rightarrow \left(e^{\Gamma_0,\theta}\right)_{\left[\forall j,p\in I_j\ x_j^{(\theta_{j,p},m_{j,p})} \mapsto t^{\Gamma'_{j,p},\theta_{j,p}}\right]},$

we have

$t^+ \Rightarrow (t')^{++} \Rightarrow_\Sigma (t')^+.$

◀

▶ **Proposition 55.** *The derivation $S^{(q_0,\Omega(q_0))} \Rightarrow \cdot \Rightarrow_\Sigma t_1 \Rightarrow \cdot \Rightarrow_\Sigma t_2 \Rightarrow \cdot \Rightarrow_\Sigma ...$ is maximal.*

Proof. A very simple way to prove it is to suppose that the tree $\|\mathcal{G}\|$ does not contain $\perp$ and then since we know that this derivation produce $\|\mathcal{G}\|$ which is maximal, then the derivation is maximal. ◀

Take the derivation $S = t_0 \Rightarrow t_1 \Rightarrow t_2 \Rightarrow ....$ Take i and $j = i + \ell$ such that $t_i = C[s]$ with s a redex and $C[\bullet]$ a Σ-context. Assume that there is an occurrence (F_i, i) of a non terminal F in s and another one (F_j, j) in t_j such that $C_{F_j}[\bullet] = C'[\bullet\ s_1...s_k]$ with $C'[\bullet]$ a Σ-context and $(F_i, i) \gg (F_j, j)$. Notice that $b_{C'[\bullet]} = b_{C[\bullet]} \cdot b'$ for some b'.

▶ **Lemma 56.** *Given a term $t : o$, such that $\Gamma \vdash t \triangleright q$ and let π be the painting on t induced by $\Gamma \vdash t \triangleright q$. Let $C[\bullet]$ be a Σ-context such that $t = C[F\ s_1...s_k]$ and let $F \triangleright (\theta, m)^b$ be the head binding of $\Gamma' \vdash F\ s_1...s_k \triangleright q'$ with $\pi(C[\bullet]) = (\Gamma', q')$. Then*

- *either $(\theta, m)^{\mathtt{f}} \in \Gamma$ and, $\Omega(b_{C[\bullet]}) = m$,*
- *or $(\theta, m)^{\mathtt{t}} \in \Gamma$ and, $\Omega(b_{C[\bullet]}) \leq m$.*

With $\Omega(b_{C[\bullet]})$ is the maximum colour appearing in the sequence of states painting the term.

Proof. Simple Induction. ◀

The following theorem allow to concludes the proof of Lemma 38.

▶ **Theorem 57.** *Let (Γ_0, q_0) be the painting of $C[\bullet]$ in t_i and (Γ, q) be the painting of $C'[\bullet]$, And let $F \triangleright (\theta, m)^b$ be the binding in Γ used for (F_j, j) in the proof of $\Gamma \vdash F\ s_1...s_k \triangleright q$. then $F \triangleright (\theta, m)^{b_0}$ in Γ_0 and*

- *if $b_0 = \mathtt{f}$, $\Omega(b', q) = m$,*
- *if $b_0 = \mathtt{t}$, $\Omega(b', q) \leq m$.*

Proof. The proof proceeds by induction on ℓ and on b'. If $\ell = 0$, we have $q_0 = q$, $\Gamma_0 = \Gamma$ and $b' = \varepsilon$. Thus looking at the proof of $\Gamma \vdash F\ s_1...s_k \triangleright q$ concludes. If $s = a\ s_1...s_k$ with $a \in \Sigma$, we can add a to b' and looking at the s_i that contains (F_i, i). If s is a redex, then combining Proposition 46 and Lemma 56 concludes. ◀