Model Checking and Functional Program Transformations
Axel Haddad

To cite this version:
Axel Haddad. Model Checking and Functional Program Transformations. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, Dec 2013, Guwahati, India. hal-00865682v2

HAL Id: hal-00865682
https://hal.archives-ouvertes.fr/hal-00865682v2
Submitted on 10 Oct 2013

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

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Model Checking and Functional Program Transformations

Axel Haddad

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 lose 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

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 $\mu$-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 $\lambda$-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 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. Again, they relied on the equivalence with 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 inspired by denotational semantics, and we give a morphism for MSO. In order to prove the logical selection, we embed carefully a winning strategy of the model-checking 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 6, we show how to use morphisms to obtain a new proof of logical reflection. In Section 7, we deal with logical selection. In Section 5, 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 languages, 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: $\text{Apply } \varphi \ x \to \varphi \ x$. 
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. \( t_1 \to (t_2 \to t_3) \) can be written \( t_1 \to t_2 \to t_3 \)), any type \( \tau \) can be written uniquely as \( \tau_1 \to \cdots \to \tau_k \to o \). The integer \( k \) is called the **arity** of \( \tau \). We define the **order of a type** by \( \text{order}(o) = 0 \) and \( \text{order}(\tau_1 \to \tau_2) = \max(\text{order}(\tau_1) + 1, \text{order}(\tau_2)) \). For instance, \( o \to o \to o \to o \) is a type of order 1 and arity 3, \( (o \to o) \to (o \to o) \), that can also be written \( (o \to o) \to o \to o \) is a type of order 2.

We let \( \tau^\ell \to \tau' \) be a shorthand for \( \tau \to \cdots \to \tau \to \tau' \).

Terms. Let \( \Gamma \) be a finite set of typed symbols, and Let \( \Gamma^\tau \) denote the set of symbols of type \( \tau \). For all type \( \tau \), we define the set of **terms** of type \( \tau \), \( T^\tau(\Gamma) \) as the smallest set containing the symbols of type \( \tau \) and the application of a term of type \( \tau' \to \tau \) to a term of type \( \tau' \) for all \( \tau' \); formally: \( \Gamma^\tau \subseteq T^\tau(\Gamma) \) and \( \bigcup \{ t \, | \, t \in T^\tau(\Gamma), s \in T^\tau(\Gamma) \} \subseteq T^\tau(\Gamma) \).

We shall write \( T(\Gamma) \) for the set of terms of any type, and \( t : \tau \) if \( t \) has type \( \tau \). The arity of a term \( t \), \( \text{arity}(t) \), 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 \( \alpha \) is the **head** of the term \( t \).

For instance, let \( \Gamma = \{ F : (o \to o) \to o \to o, G : o \to o \to o, H : (o \to o), a : o \} \). Then \( F \) and \( G a \) are terms of type \( o \to o \); \( F(Ga) \) \((H \, H \, a)\) is a term of type \( o \); \( F a \) is not a term since \( F \) is expecting a first argument of type \( o \to o \) while \( a \) has type \( o \).

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

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

**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 \( \Gamma \) and \( \mathcal{V} \) as a pair of terms of \( T(\Gamma \uplus \mathcal{V}) \) 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 \( e \in T(\Gamma \uplus \{ x_1, \ldots, x_k \}) \). Given a set of rewrite rules \( \mathcal{R} \), we define the **rewriting relation** \( \to \in T(\Gamma)^2 \) as \( t \to t' \) if there exists a context \( C[\bullet] \), a rewrite rule \( F x_1 \ldots x_k \to e \), and a term \( t_1 \ldots t_k : o \) such that \( t = C[F \, t_1 \ldots t_k] \) and \( t' = C[e] \circ t_1 \ldots t_k \). 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**\(^1\) 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 \( \Sigma \) be a finite signature, \( m \) be the maximum arity in \( \Sigma \) 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 \( \{ j \mid u \cdot j \in dom^t \} = \{ 1, \ldots, \text{arity}(a) \} \). 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 \( v \in dom^{t_u} \), \( t_u(v) = t(u \cdot v) \). Given \( a : \alpha^\ell \to o \) 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 \( dom^{t'} = \bigcup \{ i \cdot dom^t \mid t_i \cdot e = a \land t_i(i \cdot u) = t_i(u) \} \). Note that there is a direct bijection between ground terms of \( T^o(\Sigma \uplus \bot) \) and finite trees. Hence we will freely allow ourselves to treat ground terms over \( \Sigma \uplus \bot \) as trees. We define the partial

\(^1\) Although the rules are deterministics, there may be several possible rules to apply in a term.
order $\sqsubseteq$ over trees as $t \sqsubseteq t'$ if $\text{dom}^t \sqsubseteq \text{dom}^{t'}$ and for all $u \in \text{dom}^t$, $t(u) = t'(u)$ 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 \textit{limit tree} of the sequence.

**Higher Order Recursion Schemes.** A \textit{higher order recursion scheme (HORS)} $G = \langle V, \Sigma, N, R, S \rangle$ is a tuple such that: $V$ is a finite set of typed symbols called \textit{variables}; $\Sigma$ is a finite signature, called the \textit{set of terminals}; $N$ is a finite set of typed symbols called \textit{non-terminals}; $R$ is a deterministic set of rewrite rules on $\Sigma \uplus N$ and $V$, such that there is exactly one rule per non terminal and no rule for terminals; $S \in N$ is the \textit{initial non-terminal}.

We define inductively the $\perp$ \textit{transformation} $(\cdot)^\perp : T^o (N \uplus \Sigma) \rightarrow T^o (\Sigma \uplus \{ \perp : o\})$ turning a ground term into a finite tree as: $(F t_1 \ldots t_k)^\perp = \perp$ for all $F \in N$ and $(a t_1 \ldots t_k)^\perp = a t_1^\perp \ldots t_k^\perp$ for all $a \in \Sigma$. We define a \textit{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 $t_0 = S \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots$ be a derivation, then one can check that $(t_0)^\perp \sqsubseteq (t_1)^\perp \sqsubseteq (t_2)^\perp \sqsubseteq \ldots$, hence it admits a limit. One can prove\footnote{The existence of a value tree is a consequence the confluence property of HORS.} that the set of all such limit trees has a greatest element that we denote $\|G\|$ and refer to as the \textit{value tree} of $G$. Note that $\|G\|$ is the supremum of $\{ t^\perp : S \rightarrow^* t \}$. 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, $\mathcal{G}_t = \langle V, \Sigma, N \uplus \{ S' \}, R \uplus \{ S' \rightarrow t \}, S' \rangle$. One can prove that if $t \Rightarrow t'$ then $\|\mathcal{G}_t\| = \|\mathcal{G}_{t'}\|$. We call $\|\mathcal{G}_t\|$ the value tree of $t$ in $G$.

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

\textbf{Example 1.} Let $G = \langle \Sigma, V, N, S, R \rangle$ with $\Sigma = \{ a, b : o \rightarrow o; c : o \rightarrow o \rightarrow o \rightarrow o \}$, $V = \{ x : o, \varphi : o \rightarrow o \}$, $N = \{ S : o, I, F : (o \rightarrow o) \rightarrow o, D, A : (o \rightarrow o) \rightarrow o \rightarrow o \}$ and $R = \{ S \rightarrow F b, D \rightarrow x \rightarrow \varphi (\varphi x), A \rightarrow \varphi (a x), I \rightarrow \varphi \rightarrow \varphi (I \varphi), F \rightarrow c (I (A \varphi)) (F (D \varphi)) \}$. 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 $D$ 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 $I a$ can be derived to obtain $a (a (a (\ldots )))$. The tree $\|G\|$ is depicted on the left, its branches are labelled by $c^o$, $c^3$, $(b^2)^{n-1} \cdot a^o$ for all $n \geq 1$.

**Parity tree automaton.** A \textit{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 $\Sigma$ a finite signature, $Q$ a finite set called the \textit{set of states}, $\delta \subseteq \{ q \xrightarrow{a} (q_1, \ldots, q_{\text{arity}(a)}) \mid a \in \Sigma, q_1, \ldots, q_{\text{arity}(a)} \in Q \}$ is called the \textit{transition relation}, $q_0 \in Q$ the \textit{initial state}, $\Omega : Q \rightarrow \{ 1, \ldots, m_{\text{max}} \}$ for some $m_{\text{max}} \in \mathbb{N}$ the \textit{colouring function}.

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

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 \( b = (a_0, q_{0_0}) \cdot j_0 \cdot (a_1, q_{1_1}) \cdot j_1 \cdot \ldots \) in \( r \), the greatest colour seen infinitely often in \( \Omega(q_{0_0}), \Omega(q_{1_1}), \Omega(q_{2_2}), \ldots \) is even. We define \( \mathcal{A}_q \) as the automaton obtained by changing in \( \mathcal{A} \) the initial state to \( q \) : \( \mathcal{A}_q = (\Sigma, Q, \delta, q, \Omega) \), and we say that \( \mathcal{A} \) accepts the tree \( t \) from state \( q \), if \( t \models \mathcal{A}_q \).

Example 2. Let \( \mathcal{A} = (\Sigma, Q, q_0, \delta, \Omega) \) be an automaton with \( \Sigma = \{a, b : o \rightarrow o,a,c : o \rightarrow o \rightarrow o\} \), \( Q = \{q_0, q_1, q_2\} \), \( \Omega = \{q_0 \rightarrow 0, q_1 \rightarrow 1, q_2 \rightarrow 2\} \) and \( \delta = \{q_0 \xrightarrow{c} (q_0, q_0), q_0 \xrightarrow{b} q_1, q_1 \xrightarrow{a} q_2, q_1 \xrightarrow{b} q_2, q_2 \xrightarrow{a} q_1\} \).

The automaton \( \mathcal{A} \) accepts the trees whose branches are labelled by \( c^0 \) or by \( c^0 \cdot b \cdot (b^* \cdot a)^\omega \). An accepting run of \( \mathcal{A} \) on the tree \( \| G \| \) 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 \( \Sigma \), a set \( X \) and a tree \( t \) on the signature \( \Sigma \), we define the signature \( \Sigma \times X = \{(a, x) : o^k \rightarrow o \mid a : o^k \rightarrow o \in \Sigma, x \in X\} \) and we say that the tree \( t' \) on the signature \( \Sigma \times X \) is an \( X \)-annotation of \( t \), if \( \text{dom} t' = \text{dom} t \) and if for all \( u \in \text{dom} t', t'(u) = (t(u), x) \) 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 \( U\text{lab}(t') \) as the tree on \( \Sigma \) 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 \( \{0,1\} \)-annotation \( t' \) of \( t \) such that for all nodes \( u, v \in S \) if and only if \( t'(u) = (t(u), 1) \). Given a \( \mu \)-calculus formula \( \varphi \), we say that \( t' \) is a \( \varphi \)-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 \( \varphi \). Given a monadic second-order logic (MSO) formula \( \varphi[x] \) with a first order free variable, we say that \( t' \) is a \( \varphi[x] \)-reflection of \( t \) if it is a marking of the set of nodes \( u \) satisfying \( \varphi[u] \). An MSO formula \( \varphi[X] \) with a second order free variable, we say that \( t' \) is a \( \varphi[X] \)-selection of \( t \) if it is a marking of a set of nodes \( S \) satisfying \( \varphi[S] \).

Finally, we define the \( \mu \)-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 \( \mu \)-calculus (resp. MSO) if for any \( \mu \)-calculus formula \( \varphi \) (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 \( \varphi \)-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 \( \| g' \| \) 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 \( \Sigma \) and an automaton \( A = (\Sigma, Q, q_0, \delta, \Omega) \), we define an \( A \)-reflection of \( A \) on \( t \) as a \( 2^Q \)-annotation \( t' \) of \( t \), such that for all nodes \( u \), \( t'(u) = (t(u), Q') \) with \( Q' \subseteq Q \) being the set of states \( q \) such that \( A_q \) accepts the subtree of \( t \) rooted on \( u \); we define an \( A \)-selection of \( A \) on \( t \) as an accepting run of \( A \) on \( t \). We say that a class \( R \) of generators of trees is reflective with respect to automata if for any automaton \( A \) and any generator \( g \in R \), one can construct another generator \( g' \) such that \( ||g'|| \) is an \( A \)-reflection of \( ||g|| \). We say that \( R \) is selective with respect to automata if for any automaton \( A \) and any generator \( g \in R \) such that \( ||g|| \models A \), one can construct another generator \( g' \) such that \( ||g'|| \) is an \( A \)-selection of \( ||g|| \).

From the equivalence between logics and automata [21] we have that schemes are reflective with respect to automata. In Appendix A, we transform \( \mathcal{G} \) into \( \mathcal{G}' \) which, while it is producing a derivation, evaluates \( [t'] \) for any subterm \( t' \) of the current term and annotates the term with all these \( D \)-values. The new symbols of \( \mathcal{G}' \) are symbols of \( \mathcal{G} \) annotated with elements of the domain \( D \), and we define a transformation \( \cdot \mapsto \cdot' \) from terms of \( \mathcal{G} \) to terms of \( \mathcal{G}' \) such that the transformation \( t' \) of \( t \) will be annotated with the \( D \)-values of the subterms of \( t \). The tree generated by \( \mathcal{G}' \) will be annotated and when one removes these annotations, one retracts back the tree generated by \( \mathcal{G} \). More precisely we show the following.
Theorem 4 (Embedding a morphism). Given two terms \( t, t' : o \in T(\Sigma \cup N) \), if \( t \Rightarrow G t' \), then \( t^+ \Rightarrow_{G_+} t'^+ \). In particular \( \text{Unlab}(\|G_+\|) = \|G\| \) (where \( \text{Unlab} \) 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.

4.3 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 embedded into a scheme, one can add some new rewrite rules that deal explicitly with wheter the property is valid or not. For example one could replace all forbidden subtrees of the value tree by a special symbol, as illustrated in Section 5.

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 one 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 6 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. \( \|t\| \neq \bot \)). Reflecting the productivity of terms into a scheme \( \|G\| \) allows one to create a scheme \( \mathcal{G}' \) on the signature \( \Sigma \cup \{\bot\} \) such that \( \|\mathcal{G}'\| = \|G\| \), but such that no derivation will create some non productive terms. In [11] we developed this idea to compare evaluation policies.

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

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

\[
\begin{align*}
S & \rightarrow a (F H) (F J) & H x & \rightarrow a x (H x) \\
J x & \rightarrow a (J x) (J x) & F y & \rightarrow a (y b) (y c).
\end{align*}
\]

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

\[
\varphi(b) = 0 \quad \varphi(c) = 1 \quad \varphi(S) = 1 \quad \varphi(a) \ u\ v = u \lor v.
\]

One can check that the morphism \( \varphi \) is stable by rewrite. Furthermore it recognises the property "\( t \) has type \( o \), and its value tree contains a \( c \)", with the subset \( \Sigma' = \{ 0 \} \).

We construct a scheme \( G' = \{ V', \Sigma', \mathcal{N}', S, \mathcal{R} \} \) that consists of an embedding of the morphism \( \varphi \) inside the scheme \( G \). \( V' = \{ x : o, \ y^0, y^1 : o \rightarrow o \} \), \( \Sigma' = \{ a^{0,0}, a^{0,1}, a^{1,0}, a^{1,1} : o^2 \rightarrow o, \ b, c : o \} \), \( \mathcal{N}' = \{ S : o, \ H^0, H^1, J^0, J^1 : o, \ F^{[0,0]}, F^{[0,1]}, F^{[1,0]}, F^{[1,1]} : (o \rightarrow o)^2 \rightarrow o \} \), \( \mathcal{R}' \) contains the following rewrite rules:

\[
\begin{align*}
S & \rightarrow a^{1,0} (F^{[0,1]} H^0 H^1) (F^{[0,0]} J^0 J^1) \\
H^0 x & \rightarrow a^{0,0} x (H^0 x) \\
H^1 x & \rightarrow a^{1,1} x (H^0 x) \\
J^0 x & \rightarrow a^{0,0} (J^0 x) (J^0 x) \\
J^1 x & \rightarrow a^{0,0} (J^1 x) (J^1 x) \\
F^{[0,0]} y^0 y^1 & \rightarrow a^{0,0} (y^0 b) (y^1 c) \\
F^{[0,1]} y^0 y^1 & \rightarrow a^{0,1} (y^0 b) (y^1 c) \\
F^{[1,0]} y^0 y^1 & \rightarrow a^{1,0} (y^0 b) (y^1 c) \\
F^{[1,1]} y^0 y^1 & \rightarrow a^{1,1} (y^0 b) (y^1 c).
\end{align*}
\]

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

The first occurrence of \( y \) is applied to \( b \), therefore it should be annotated with \( \varphi(b) = 0 \). Similarly, the second occurrence of \( y \) should be annotated with \( \varphi(c) = 1 \). 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 \( y \ b \) is equal to \([0, 1]\ \varphi(b) = [0, 1] 0 = 0 \) and by a similar reasoning the evaluation of \( y \ c \) is equal to \( 1 \). Therefore the occurrence of \( a \) 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 \( c \). For instance, the occurrence of \( a \ c \) 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 \( F^{[0,0]} y_1 y_2 \rightarrow \text{FORBIDDEN} \), where \( \text{FORBIDDEN} : o \) is a new terminal added to the scheme. Here is the whole set of rewrite rules transformed this way.

![Figure 1](The value tree of the scheme of Section 5.)
6 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 4.2, 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 → pushdown automaton → reflective pushdown automaton → reflective scheme) lose the structure of the scheme. In our construction, the structure of the scheme is the preserved, in the sense of Remark 4.2. Since our proof of the logical reflection, as well as the one of logical selection in Section 7, 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.

6.1 Kobayashi-Ong result

We fix a non deterministic parity tree automaton \( \mathcal{A} = (\Sigma, Q, q_1, \delta, \Omega) \) and a scheme \( \mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{V}, S, \mathcal{R}) \). We let \( \text{arity}_{\text{max}}, \text{order}_{\text{max}}, \) and \( m_{\text{max}} \) be the maximum arity in \( \Sigma \uplus \mathcal{N} \), order in \( \Sigma \uplus \mathcal{V} \), 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 a set of 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 \( (\theta_1, \theta_1) \wedge \ldots \wedge (\theta_k, \theta_k) \to \theta \) with for all \( i \), \( \theta_i \) is a mapping of type \( \tau \), \( m_i \) is a color, and \( \theta \) is a mapping of type \( \tau' \). The judgements are of the form \( \Gamma \vdash t \theta \theta \) meaning that under the environment \( \Gamma \), one can judge \( t \) with the mapping \( \theta \). The environment \( \Gamma \) associates some mapping and colors to non terminal and variables, and gives 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 judged as \( \emptyset \vdash a \theta (q_1, m_1) \to \ldots \to (q_k, m_k) \to q \) 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 \((F, \theta, m)\), Eve has to find an environment \(\Gamma\) such that she can prove \(\Gamma \vdash r_F : \theta\), then Adam picks a \(F\) and \(\theta'\) in \(\Gamma\) and asks Eve to prove that \(\theta'\) 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.

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

6.2 A morphism for automata reflection

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

\[\text{Theorem 6. The morphism } [\cdot] \text{ recognises the states of the automaton, i.e. for each state } q \in Q \text{ of the automaton, it recognises the set } T_q = \{t : o \mid \|G_t\| \models A_q\} \text{ which is the set of ground terms whose associated value tree is recognised by the automaton from state } q. \text{ Furthermore, it is stable by rewriting.}\]

Using the construction of Section 4.2, we have the following result.

\[\text{Corollary 7 (Automata Reflection). Higher order recursion schemes are reflective with respect to automata (hence they are reflective with respect to } \mu\text{-calculus, and to MSO).}\]

7 Selection

Given a signature \(\Sigma\), 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 \(\|G\|\) 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 \(\|G'\|\) is a \(S\)-marking of \(\|G\|\). 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 4.2. The construction is also based on the 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.

\[\text{Theorem 8 (Automata Selection). Higher order recursion schemes are selective with respect to automata (hence to MSO).}\]
Proof 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 $G$ and an automaton $A$ such that $\|G\| \models A$. We want to construct a scheme $G'$ such that $\|G'\|$ is an accepting run of $A$ on $\|G\|$. The first observation is that there are some great similarities between the structure of the proof trees in the type 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 $t_0 : o \rightarrow o$ and $t_1 : o$ and assume that we have a proof $\mathcal{P}_0$ of $t_0 \triangleright (\theta_1, m_1) \rightarrow \theta$ 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 $\triangleright t_0 \triangleright t_1 \triangleright \theta$. 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 $t^P$ where each symbol is annotated by an atomic mapping and a color, verifying that if a non terminal $F$ is annotated by $(\theta, m)$ then $\Gamma$ associate $F$ to $(\theta, m)$. The term $t^P$ is somehow a trace of the proof $\mathcal{P}$. Now given a rewrite rule $F \ x_1 \ldots x_k \rightarrow e$ in the original scheme, we want to define for all annotated versions $F^{(\theta, m)}$ an associated rewrite rule in the transformed scheme. If $(F, \theta, m)$ is a winning vertex of the game, then Eve can choose an environment $\Gamma$ such that $\Gamma \vdash e \triangleright \theta$. We take a proof $\mathcal{P}$ of this judgement and we define $F^{(\theta, m)}(x_1 \ldots x_k) \rightarrow e^P$. Since Eve has chosen the environment $\Gamma$ with respect to her winning strategy, then for all annotated non terminals $H^{(\theta', m')}$ appearing in $e^P$, $(H, \theta', m')$ is winning in the game. In particular, if the initial non terminal of the transformed scheme is $S^{(\theta, m)}$ then $S^{(\theta, m)}$ 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 $(F, \theta, m)$ is not winning.

As we said, the initial non terminal is $(S, q_0, \Omega(q_0))$ which is winning in the game since $A$ accepts $\|G\|$ from state $q_0$. Any terminal $a$ will be annotated by some $(q_1 \rightarrow \cdots \rightarrow q_k \rightarrow q, m)$. Then to obtain elements of $\Sigma \times Q$, we just turn any $a^{(q_1 \rightarrow \cdots \rightarrow q_k \rightarrow q, m)}$ into a non terminal and we add a rewrite rule transforming it into $a^\theta$.

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 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 $G'$ the greatest colour seen infinitely often is even, hence it is an accepting run.

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. 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

A. Haddad 13

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 \( D \)-value of their arguments, for example in the term \( F \ t_1 \ldots t_k \), we will label \( F \) with the \( k \)-tuple \((\llbracket t_1 \rrbracket \ldots \llbracket t_k \rrbracket)\).

A problem may appear if some of the arguments are not fully applied. For example, imagine we want to annotate \( F \ H \), where \( F : (a \rightarrow o) \rightarrow o \) and \( H : o \rightarrow o \): we will annotate \( F \) with \( \llbracket H \rrbracket \), 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 \( D \)-value of its arguments, i.e. one per element of \( D^o \). For example assume that there are 4 of them: \( D_o = \{d_1, d_2, d_3, d_4\} \) 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 \((o \rightarrow o) \rightarrow o \), but \( F^{[H]} \) has type \((o \rightarrow o)^4 \rightarrow o \). Also note that, even if \( F \) has 4 arguments, it only has to be labelled with a single \( D \)-value since all four arguments represent different labellings of the same term. We now formalise these notions.

Types

to a type \( \tau = \tau_1 \rightarrow \ldots \rightarrow \tau_k \rightarrow o \) we associate the integer
\[
n_{\tau} = \text{Card}(\{(d_1, \ldots, d_k) \mid \forall i \ d_i \in D^{\tau_i}\}),
\]
and we fix an ordering \((d^1, \ldots, d^n)\) over the set \((\{d_1, \ldots, d_k\} \mid \forall i \ d_i \in D^{\tau_i}\})\). We define inductively the type \( \tau^+ \) as \( \tau^+ = (\tau_1^+)^{n_1} \rightarrow \ldots \rightarrow (\tau_k^+)^{n_k} \rightarrow o \). Note that, as \( n_o = 1 \), \((o^k \rightarrow o)^+ = o^k \rightarrow o \).

Symbols

to a terminal \( a : o^k \rightarrow o \) and a tuple \( d_1, \ldots, d_k \in D^o \), we associate the terminal \( a^{d_1, \ldots, d_k} : o^k \rightarrow o \in \Sigma' \).

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

Terms

given a mapping \( \rho : \mathcal{W} \rightarrow \mathcal{D} \) with \( \mathcal{W} \subseteq \mathcal{V} \), such that if \( x : \tau, \rho(x) : \tau \), let \( \llbracket \cdot \rrbracket_\rho \) be the morphism from \( \mathcal{T}(\Sigma \cup \Sigma' \cup \mathcal{W}) \) to \( \mathcal{D} \) defined by letting \( \llbracket \alpha \rrbracket_\rho = \llbracket \alpha \rrbracket \) for \( \alpha \in \Sigma \cup \Sigma' \) and \( \llbracket x \rrbracket_\rho = \rho(x) \) for \( x \in \mathcal{W} \).

given a term \( t : \tau = \tau_1 \rightarrow \ldots \rightarrow \tau_k \rightarrow o \in \mathcal{T}(\mathcal{V} \cup \Sigma' \cup \mathcal{W}) \) and a mapping \( \rho : \mathcal{W} \rightarrow \mathcal{D} \) such that \( \mathcal{W} \subseteq \mathcal{V} \) contains all the variables appearing in \( t \), we define inductively the term \( t^{\rho^{d_1, \ldots, d_k}} : \tau^+ \in \mathcal{T}(\mathcal{V} \cup \Sigma' \cup \mathcal{W}) \) for all \( d_1 : \tau_1, \ldots, d_k : \tau_k \). If \( t = F \in \Sigma' \) (resp. \( t = x \in \mathcal{V} \), \( t^{\rho^{d_1, \ldots, d_k}} = F^{d_1, \ldots, d_k} \) (resp. \( t^{\rho^{d_1, \ldots, d_k}} = x^{d_1, \ldots, d_k} \)), if \( t = a \in \Sigma \), \( t^{\rho^{d_1, \ldots, d_k}} = a^{d_1, \ldots, d_k} \). Finally consider the case where \( t = t_1 t_2 \) with \( t_1 : \tau' \rightarrow \tau \) and \( t_2 : \tau' \). Let \( d = \llbracket t_2 \rrbracket_\rho \). Remark that \( t^{\rho^{d,d_1, \ldots, d_k}} \) has type \((\tau')^{n'} \rightarrow \tau^+ \). We define
\[
(t_1 t_2)^{\rho^{d_1, \ldots, d_k}} = t_1^{\rho^{d_1, \ldots, d_k}} \ L^{\rho_{12}},
\]
where $\bar{t}_p = \bar{t}_1^{+\alpha_{1r_1}}, \ldots, \bar{t}_p^{+\alpha_{pr_p}}$. 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 $\Sigma'$ is simply an annotated tree on $\Sigma$.

**Rewrite Rules**

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

$$F^{d_1, \ldots, d_k} x_1 \ldots x_k \rightarrow e^+,$$

where for all $i$, $\bar{x}_i = x_1^{d_{1i}}, \ldots, x_1^{d_{pi}}$. Finally, we let $\mathcal{G}' = (\mathcal{V}', \Sigma', \mathcal{N}', \mathcal{R}', S)$.

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

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

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

**Proof of Theorem 4.2.** First we need the following lemma.

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

$$(e_{[\forall i, x_i \mapsto t_i]})^+ = e^+[\forall i, j, x_i^{t_{ji}} \mapsto t_{ji}^+].$$

**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 $t, t' : o \in T(\Sigma \sqcup \mathcal{N})$, and $d_1 : \tau_1', \ldots, d_k : \tau_k' \in \mathcal{D}$, if $t \Rightarrow t'$, then $t^{+d_1, \ldots, d_k} \Rightarrow t'^{+d_1, \ldots, d_k}$. We prove this result by induction on the structure of $t$.

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

$$t^{+d_1, \ldots, d_k} = \alpha^{[t_1], \ldots, [t_k], d_1, \ldots, d_k} t_1^{+d_{1i}} \cdots t_k^{+d_{ki}} = t_1^{+d_{1i}^\tau} \cdots t_k^{+d_{ki}^\tau}.$$

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

$$s = \alpha^{[t_1], \ldots, [t_k], d_1, \ldots, d_k} s_1^{d_{1i}} \cdots s_k^{d_{ki}}$$

with for all $i, j$, $t_i^{+\bar{x}_{ji}} \Rightarrow s_i^j$. Then by induction hypothesis for all $i, j$, $s_i^j = t_i^{+\bar{x}_{ji}}$. Then $s = t^{+d_1, \ldots, d_k}$.

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

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

$$s = (e_{[\forall i, x_i \mapsto t_i]})^+ = (t')^+.$$
We define inductively the typed set $\text{Map}$ of atomic mappings $\sigma$. We say that $A$ strategy $\varphi$ is winning if there exists a mapping $\sigma$ such that for all $i$, $(v_i, v_{i+1}) \in E$, and if $p = v_1 \ldots v_k$ 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 $\varphi$ 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 $v_0 \cdot v_1 \ldots$ we say that Eve respects $\varphi$ if for all $v \in V_E$, $v_{i+1} = \varphi(v_i)$. We say that the strategy is winning from a vertex $v_0$ if Eve wins all plays where she respects $\varphi$ and that starts from $v_0$. A strategy $\varphi$ 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 $\varphi$ if for all $v \in V_E$, $v_{i+1} = \varphi(v_i)$. We say that the strategy is winning if Eve wins all play $p$ where she plays according to $\varphi$. A strategy $\varphi$ is positional (or memoryless) if there exists a mapping $f : V_E \to V_A$ such that for 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 $\text{NP} \cap \text{coNP}$.

B.2 A formal presentation of Kobayashi Ong result

In the following we fix a non deterministic parity tree automaton $A = (\Sigma, Q, q_0, \delta, \Omega)$ and a scheme $\mathcal{G} = (\Sigma, N, V, S, R)$. We let $\text{arity}_{\text{max}}$, $\text{order}_{\text{max}}$, and $m_{\text{max}}$ be the maximum arity in $\Sigma \cup N$ (resp. the maximum order in $\Sigma \cup N$, 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 $\text{Map}$ of atomic mappings $\sigma$ as $\text{Map} = Q$ and

$$
\text{Map}^\tau \to \tau' = \left\{ \sigma : \theta \to \theta' \mid \theta' \in \text{Map}^\tau, \right. \\
\left. \sigma \subseteq \{ (\theta, m) \mid \theta \in \text{Map}^\tau, m \geq \text{max}(\Omega(\theta), \Omega(\theta')) \} \right\},
$$

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 $\tau$ we choose and fix a complete ordering of the finite set $\text{Map}^\tau$, 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 $\theta$ to range over atomic mappings.

An environment $\Gamma$ is a set of tuples $\alpha \triangleright (\theta, m)^b$ (called bindings) with $\alpha \in N \cup V$, $\theta$ an atomic mapping, $m \leq m_{\text{max}}$ a colour, and $b \in \{ \text{f}, \text{t} \}$, called the flag.
Intuitively, \( \alpha \triangleright (\ell, m)^b \) means "an occurrence of \( \alpha \) can be typed by \( \ell \) if the greatest colour seen along the path from the root of the term to the occurrence of \( \alpha \) is \( m \)." The binding \( \alpha \triangleright (\ell, m)^b \) is less restrictive, it means "an occurrence of \( \alpha \) can be typed by \( \ell \) if no colour seen in the path from the root of the term to the occurrence of \( \alpha \) is greater or equal than \( m \)."

In the type system the flag will switch from \( \bar{x} \) to \( t \) whenever the colour \( m \) is seen.

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

\[
(\ell, m)^b \uparrow m' = \begin{cases} 
(\ell, m)^b & \text{if } m' < m \\
(\ell, m)^b & \text{if } m' = m \\
\text{undefined} & \text{if } m' > m
\end{cases}
\]

Intuitively, \( (\ell, m)^b \uparrow m' \) is the way to inform \( (\ell, m)^b \) that the colour \( m' \) has been seen. We define \( \{\alpha \triangleright (\ell_1, m_1)^b_1, ..., \alpha \triangleright (\ell_k, m_k)^b_k\} \uparrow m' = \{\alpha \triangleright (\ell_1, m_1)^b_1 \uparrow m', ..., \alpha \triangleright (\ell_k, m_k)^b_k \uparrow m'\} \).

Note that it is undefined if one of the \( (\ell_i, m_i)^b_i \uparrow m' \) is undefined.

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

\[
\frac{}{\{\alpha \triangleright (\ell, m)^b\} \vdash \alpha \triangleright \theta} \quad \text{(T-VAR)}
\]

if \( (\ell, m)^b \uparrow \Omega(\theta) \) is defined and equal to \( (\ell, m)^{b'} \),

\[
\frac{}{\emptyset \vdash a \triangleright (q_1, m_1) \rightarrow ... \rightarrow (q_k, m_k) \rightarrow q} \quad \text{(T-CONST)}
\]

if \( a \in \Sigma \) and \( q \overset{a}{\rightarrow} q_1, ..., q_k \in \delta_A \) and for all \( i, m_i = \max(\Omega(q_i), \Omega(q)) \),

\[
\frac{\Gamma_0 \vdash t_0 \triangleright (\ell_1, m_1) \land ... \land (\ell_k, m_k) \rightarrow \theta \quad \forall i \Gamma_i \vdash m_i \vdash t_1 \triangleright \theta_i}{\Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \vdash t_0 \triangleright \theta} \quad \text{(T-APP)}
\]

if for all \( i, \Gamma_i \vdash m_i \) is defined.

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

In the following, we may use the shorthand \( \bigwedge_{j \in J}(\ell_j, m_j) \) for the sequence \( (\ell_{j_1}, m_{j_1}) \land ... \land (\ell_{j_k}, m_{j_k}) \) with \( J = \{j_1, ..., j_k\} \), and \( \alpha \triangleright \bigwedge_{j \in J}(\ell_j, m_j)^b \) for the environment \( \alpha \triangleright (\ell_{j_1}, m_{j_1})^{b_1}, ..., \alpha \triangleright (\ell_{j_k}, m_{j_k})^{b_k} \).

Given a rewrite rule \( r_F = F \ x_1...x_k \rightarrow e \), an environment \( \Gamma \) and an atomic mapping \( \theta' = \bigwedge_{j \in J}(\ell'_{j_1}, m_{j_1}) \rightarrow ... \rightarrow \bigwedge_{j \in J}(\ell'_{j_k}, m_{j_k}) \rightarrow \theta \), 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 \( \Gamma, x_1 \triangleright \bigwedge_{j \in I_i}(\ell_{j_1}, m_{j_1}), ..., x_k \triangleright \bigwedge_{j \in I_k}(\ell_{j_k}, m_{j_k}) \vdash e \triangleright \theta \).

\section{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 $G_A = \langle V_A, V_3, E, \Omega \rangle$ as follow: $V_3 = \{ (F, \theta, m) \mid F \in \mathcal{N}, \ F : \tau, \theta : \tau \}$, $V_A = \{ \Gamma \mid \alpha \vdash (\theta, m) \} \subseteq \Gamma \in \mathcal{N} \land b \neq z \}$, $E = \{ (F, \theta, m), \Gamma \} \mid \Gamma \vdash r_F \vdash \theta \} \cup \{ (\Gamma, (F, \theta, m)) \mid F \vdash (\theta, m)^f \in \Gamma \}$. For all $(F, \theta, m) \in V_3$, $\Omega((F, \theta, m)) = m$ and for all $\Gamma \in V_A$, $\Omega(\Gamma) = 0$.

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 $(F, \theta, m)$, Eve has to find an environment $\Gamma$ such that she can prove $\Gamma \vdash r_F : \theta$, then Adam picks a biding $F' \vdash (\theta', m')^d \in \Gamma$ and asks Eve to prove that $\theta'$ is chosen correctly. If at some point of the 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 $\| G \|$ is accepted by $A$ from the state $q$ if and only if Eve has a positional winning strategy from the vertex $(S, q, \Omega(q))$ in the game $G_A$.

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

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

$$d d' = \{ \theta \mid \exists(\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \rightarrow \theta \in d \forall i \theta_i \in d' \}.$$  

We will show that for any ground term $t : o$, $[t] = \{ q \mid \| G_t \| = A_q \}$, and that the morphism is stable by rewriting.

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

#### B.3.1 The Environment witnesses the recognition of the automaton

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

**Lemma 11.** If $(F, \theta, m)$ is winning for some $m$, then, for all $m'$, $(F, \theta, m')$ is winning.

**Proof.** From $(F, \theta, m')$ Eve follows the exact same strategy. □

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

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

We define the mapping $\Pi$ 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 $\Pi(\Gamma^e) = \Pi(\Gamma^F) = S$. Finally note that a subset $\Gamma$ of $\Gamma^e$ is entirely defined by $\Pi(\Gamma)$ (indeed, $\Gamma = \{ (F, \theta, m_{\text{max}}) \mid (F, \theta) \in \Pi(\Gamma) \}$).
The idea of the proof is the following: first we show that \( A \) accepts \( \|G_t\| \) from state \( q \) if and only if \( q \in \Gamma^t(t) \), and then we show that for all term \( t, \Gamma^t(t) = \Gamma^t(t) \).

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

**Proof of Theorem 12.** First, as we pointed out before, notice that if for some \( \Gamma, s \) and \( \theta \), \( \Gamma \vdash s : \theta \) holds then all the elements of \( \Gamma \) 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 \( \Gamma \) of the form \( \Gamma \vdash (\theta', m)^\theta \).

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

Assume that \( A \) accepts \( \|G_t\| \) 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 \( \Gamma \) be the choice of Eve of vertex \((I, q, \Omega(q))\) according to the strategy. Note that \( \Gamma \) does not contains any \( I \vdash (q', m)^\theta \) since \( I \) does not appear in \( t \). Furthermore, since it is a winning strategy, if \( F : (\theta, m)^\theta \in \Gamma \) then \( (F, \theta, m) \) is winning in \( G_t \) hence it is winning in \( G \) then \( \Gamma \subseteq \Gamma^t \). From the definition of the game comes that \( \Gamma \vdash t \vdash q \).

On the other hand, assume that there is a subset \( \Gamma \) of \( \Gamma^t \) such that \( \Gamma \vdash t \vdash q \). Then the strategy of playing \( \Gamma \) on state \((I, q, \Omega(q))\) and \( \Gamma(F, \theta, m) \) on \((F, \theta, m)\) for all winning vertices \((F, \theta, m)\) is winning, hence \( A \) accepts \( \|G_t\| \) from state \( q \).

**Lemma 13.** Let \( \Gamma \) be a subset of \( \Gamma^t \), and \( m \) a color, then \( \Gamma \vdash m \) is defined and is equal to \( \Gamma \). Furthermore for any environment \( \Gamma \), if \( \Gamma \vdash m \) is defined, then \( \Pi(\Gamma \vdash m) = \Pi(\Gamma) \).

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

**Lemma 14.** (Technical, but needed to be formal) Given an environment \( \Gamma \), a term \( t \), and \( F \in N \). Given some \( \theta' \) and \( m \), assume that \( \Gamma \) does not contain neither the binding \( F : (\theta', m)^\theta \) nor \( F : (\theta', m)^\theta \).

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

**Proof.** We prove the result by induction on the structure of \( t \).

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

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

If \( t = t_0 \), then there exists \( \Gamma_0, \Gamma_1, ..., \Gamma_k \) such that \( \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k = \Gamma \) \( \{F : (\theta', m')^\theta\} \) , \( \Gamma_0 \vdash t_0 \vdash (\theta_1, m_1) \wedge ... \wedge (\theta_n, m_n) \rightarrow \theta \) and for all \( i \in \{1, ..., k\} \), \( \Gamma_i \cup \Gamma_i \vdash \Gamma_i \vdash t_i \vdash \theta_i \). If \( \Gamma_0 = \Gamma_0 \cup \{F : (\theta', m')^\theta\} \vdash t_0 \vdash (\theta_1, m_1) \wedge ... \wedge (\theta_n, m_n) \rightarrow \theta \). If for some \( i \geq 1 \), \( \Gamma_i = \Gamma_i \cup \{F : (\theta', m')^\theta\} \) then \( (\theta', m')^\theta \vdash m_i \) is define and \( (\theta', m')^\theta \vdash m_i = (\theta', m')^\theta \) and \( \Gamma_i \vdash m_i = (\Gamma_i \vdash m_i) \vdash \{F : ((\theta', m')^\theta\} \). Then \( m \geq m_i \) hence \( m' \geq m_i \), thus \( (\theta', m')^\theta \vdash m_i \) is define and is equal to \( (\theta', m')^\theta \) then \( (\Gamma_i \vdash ((\theta', m')^\theta\} \vdash m_i = (\Gamma_i \vdash m_i) \cup \{F : ((\theta', m')^\theta\} \vdash t_i \vdash \theta_i \). Then \( \Gamma \vdash \{F : (\theta', m')^\theta\} \vdash t \vdash \theta \).
Corollary 15. Let $\Gamma \subseteq \Gamma'$ such that $\Gamma \vdash t : \theta$. Let $\Gamma' \subseteq \Gamma$ such that $\Pi(\Gamma') = \Pi(\Gamma)$ then $\Gamma' \vdash t \triangleright \theta$.

Proof. The proof proceeds by induction on the number of non terminals in $\Gamma$.

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

Proof. This proof results by induction of the structure of $t$.

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

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

Since $(\theta, \Omega(\theta))^f \triangleright \Omega(\theta) = (\theta, \Omega(\theta))^f$, 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). $\Gamma' = \Gamma_0 \cup \Gamma_1 \cup \ldots \cup \Gamma_k$ with $\Gamma_0 \vdash t_0 \triangleright (\{0\}, m_1) \wedge \ldots \wedge (\theta_k, m_k) \rightarrow \theta$ and for all $i \in \{1, \ldots, k\}$ $\Gamma_i \vdash t_1 \triangleright \theta_i$. Then, by induction hypothesis let $\Gamma_0, \Gamma_1, \ldots, \Gamma_k \subseteq \Gamma^\tau$ such that $\Gamma_0 \vdash t_0 \triangleright (\{0\}, m_1) \wedge \ldots \wedge (\theta_k, m_k) \rightarrow \theta$ and for all $i \in \{1, \ldots, k\}$ $\Gamma_i \vdash t_1 \triangleright \theta_i$. For all $i \in \{1, \ldots, k\}$ we define $\Gamma_i'' \subseteq \Gamma^\tau$ as $\Gamma_i'' = \Gamma_i \uparrow m_i$ (defined by $\Gamma_i'' = \{F \triangleright (\theta', \text{max}(m_i, m')^f | F \triangleright (\theta, m')^f \in \Gamma_i\}$). Then we have $\Gamma_i'' \uparrow m_i = \{F \triangleright (\theta', m_i)^f | F \triangleright (\theta', m')^f \in \Gamma_i, m_i' \geq m_i\}$, thus, using Theorem 14 one can state $\Gamma_i'' \uparrow m_i \vdash t_1 \triangleright \theta_i$, then if we define $\Gamma = \Gamma_0 \cup \Gamma_1'' \cup \ldots \cup \Gamma_k'' \subseteq \Gamma^\tau$, we have $\Gamma \vdash t \triangleright \theta$.

Theorem 17 $(\Gamma^\tau$ Witnesses the Recognition of $A)$. Given a term $t : o$ and a state $q$ of the automaton, then $A$ accepts $\|G_s\|$ from state $q$ if and only if there exists a subset $\Gamma'$ of $\Gamma^\tau$ such that $\Gamma' \vdash t : q$.

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

B.3.2 The Environment witnesses the morphism

We show that for all term $t$, $\Gamma^\tau(t) = \llbracket t \rrbracket$. Furthermore we show that the morphism is stable by rewriting.

Theorem 18. Given a term $t$, $\llbracket t \rrbracket = \Gamma^\tau(t)$.

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

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

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

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

From $\Gamma_0 \cup \Gamma_1 \cup \ldots \cup \Gamma_k = \Gamma$ we get that all $\Gamma_i$ are subsets of $\Gamma^\tau$. Then from $\Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \rightarrow \theta$ we get that $(\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \rightarrow \theta \in \Gamma^\tau(t_0)$, then by induction hypothesis $(\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \rightarrow \theta \in \llbracket t_0 \rrbracket$. Using Lemma 13 we have that for all $i \geq 1$, $\Gamma_i \vdash m_i \rightarrow \Gamma_i$, hence $\theta_i \in \Gamma^\tau(t_0)$ therefore by induction hypothesis $\theta_i \in \llbracket t_0 \rrbracket$. According to the definition of $\llbracket \cdot \rrbracket$ we have $\theta \in \llbracket t_0 \rrbracket$ semti = [t_0 t_1].
On the other hand, if \( \theta \in [t_0, t_1] = [t_0] \) \( \text{sent} \), then there exist \( \theta_1, ..., \theta_k \) and \( m_1, ..., m_k \) such that \( (\theta_1, m_1) \land ... \land (\theta_k, m_k) \rightarrow \theta \in [t_0] \) and for all \( i \), \( \theta_i \in [t_1] \). Then by induction hypothesis there exists \( \Gamma_0, ..., \Gamma_k \subseteq \Gamma^* \) such that \( \Gamma_0 \uparrow t_0 \triangleright (\theta_1, m_1) \land ... \land (\theta_k, m_k) \rightarrow \theta \) and for all \( i \geq 1 \), \( \Gamma_i \uparrow t_1 \triangleright \theta_i \). Then again, using Lemma 13 we have that for all \( i \), \( \Gamma_i \uparrow m_i = \Gamma_i \), then for all \( i \geq 1 \), \( \Gamma_i \uparrow t_i \triangleright \theta_i \). Therefore \( \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \uparrow t_1 t_2 \triangleright \theta \), and since \( \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k \subseteq \Gamma^r \), \( \theta \in \Gamma^r(t_1 t_2) \).

\begin{itemize}
\item \textbf{Theorem 19.} \textit{Given two terms \( t, t' \), if \( t \rightarrow t' \) then \( [t^r] = [t'^r] \).}
\end{itemize}

\begin{proof}
Since \( t \rightarrow t' \), there exist two terms \( r, r' : o \) and a context \( C[\bullet] \) such that \( r \rightarrow r' \), therefore we have \( ||G_r|| = ||G_r'|| \). Theorem 17 states that \( \{r \} = \{q \mid A \text{ accepts } ||G_r|| \text{ from state } q\} \), then we have \( [r] = [r'] \), hence \( [t] = [C[r]] = [C[r']] = [C[t]] \).
\end{proof}

\textbf{B.4 MSO reflection}

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

\begin{itemize}
\item \textbf{Theorem 20 (Broadbent, Carayol, Ong, Serre).} \textit{Let \( R \) be a class of generators of trees. If \( R \) is reflective with respect to modal \( \mu \)-calculus and with respect to regular paths, then it is also reflective with respect to MSO.}
\end{itemize}

We need to define what means the reflexivity with respect to regular paths. Given a regular language \( L \) in \( \langle \Sigma \cdot \{1, ..., \text{arity}_{\max}\} \rangle^* \), a scheme \( S \) on \( \Sigma \), one can reflect \( L \) on \( S \) if one can construct a scheme \( S' \) on \( \Sigma \times \{0, 1\} \) such that \( ||S'|| \) 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 \( \text{Path}(u) = ||S|| \cdot (\varepsilon \cdot j_0 \cdot ||S'||(j_0) \cdot j_1 \cdot ||S'||(j_0 j_1) \cdot j_2 \cdot ... \cdot ||S'||(j_0 j_1 ... k_{j-1}) \cdot j_k \) is in the language \( L \).

\begin{itemize}
\item \textbf{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.
\end{itemize}

\begin{itemize}
\item \textbf{Lemma 21.} \textit{Higher order recursion schemes are reflective with respect to regular paths.}
\end{itemize}

\textbf{B.5 Schemes are reflective with respect to regular paths, proof of Lemma 21.}

Take a regular language \( L \) in \( \langle \Sigma \cdot \{1, ..., \text{arity}_{\max}\} \rangle^* \). There is a finite word automaton \( B = \langle \Sigma \cdot \{1, ..., \text{arity}_{\max}\}, Q, q_0, F \subseteq Q, \delta_B \rangle \) that recognises exactly \( L \). We construct a parity tree automaton \( A = \langle \Sigma, Q, q_0, \delta, \Omega = \{q \rightarrow 0 \mid q \in Q\} \rangle \) such that \( \text{Path}(u) \in L \) if and only if there is a state \( q_f \in F \) that can be reached by \( A \) on \( u \). We define the transitions in \( \delta \) as \( q \xrightarrow{a} q_1, q_2, q_3 \in \delta \) if for all \( j \) there exists \( q' \) such that \( q \xrightarrow{a} q' \in \delta_B \) and \( q' \xrightarrow{b} q_j \in \delta_B \). A simple induction shows that \( A \) satisfies the property we wanted. Now we show that we can embed the reachable states of \( A \) in a scheme, which would conclude the proof, since we can always turn a terminal labelled by a set of state \( aQ \subseteq Q \) into a nonterminal, and make it transforms into the terminal \( (a, 0) \) if \( Q' \cap F = \emptyset \) and \( (a, 1) \) otherwise.

\begin{itemize}
\item \textbf{Lemma 22 (Reachable States Reflexion).} \textit{Given an automaton \( A \) and a scheme \( S \) one can create a scheme \( S' \) on \( \Sigma \times 2^Q \) such that the projection on the first component of \( ||S'|| \) is \( ||S|| \) and such that for all \( u, ||S'||(u) = (a, Q') \) then \( Q' \) is the set of reachable states at the node \( u \).}
\end{itemize}
Proof. We use a similar construction as in Section 4.2. 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 \( \tau^+ = (\tau^+_k)^N \rightarrow \) ...

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

Given a term \( t : \tau \in T(\Sigma' \cup \Sigma' \cup N') \) and a set \( Q' \subseteq Q \), we define inductively the term \( t^{Q'} : \tau^+ \in T(\Sigma' \cup \Sigma' \cup N') \) f. If \( t = a \in \Sigma \) a symbol, \( t^{Q'} = \bar{Q} \), if \( t = a \in \Sigma \cup V \), \( t^{Q'} = \alpha^{Q'} \), if \( t = t_1 t_2 \) we define \((t_1 t_2)^{Q'} = t_1^{Q'} t_2^{Q'} ... t_2^{Q'} \). 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 \( F : \tau \rightarrow \rightarrow \tau_k \rightarrow o \in N', Q' \subseteq Q \). If \( F x_1 ... x_k \rightarrow e \in R \), we define in \( R' \) the rule \( F^{Q'} x_1^{Q'} ... x_k^{Q'} \rightarrow e^{Q'} \).

Finally, We let \( G' = (V', \Sigma', N', R', S^{(e)}) \).

Lemma 23. Given two terms \( t, t' : o \in T(\Sigma \cup N) \), if \( t \Rightarrow t' \), then \( t^Q \Rightarrow t'^Q \).

Proof. The proof is entirely similar to the one of Section 4.2. \( \triangleright \)

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

\[
\bar{Q}_1 x_1^{Q_1} ... x_k^{Q_k} \rightarrow a^{Q_1} x_1^{Q_1} ... x_k^{Q_k}
\]

with for all \( j, Q_{ij} = \{ q_i | \exists q \in Q, q \xrightarrow{a} q_1, ..., q_i, ..., q_k \} \).

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

Proof. A simple structural induction proves the result. \( \triangleright \)

Let \( S = t_0 \Rightarrow t_1 \Rightarrow t_2 \Rightarrow \ldots \) be the parallel derivation in \( G \). We know that the limit tree of \( t_0, t_1, t_2, \ldots \) is equal to \( ||G|| \), we get by the previous lemma that \( ||G|| \subseteq ||G''||_1 (||G''||_1 \) denote the projection on the first component of \( ||G''|| ) \). Since we can assume without loss of generality that there are no occurrences of \( \perp \) in \( ||G|| \), we get \( ||G|| = ||G''||_1 \), which concludes the proof. \( \triangleright \)
C Proof of Theorem 8: automata selection

C.1 The construction

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

We write \( \text{arity}_{\text{max}} \) the maximum arity of \( \Sigma \cup N \), \( \text{order}_{\text{max}} \) the maximum order, and \( m_{\text{max}} \) 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 \( \text{order}_{\text{max}} \) and \( \text{arity}_{\text{max}} \).

Types

To a type \( \tau \) we associate the integer

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

We define inductively the type \( \tau^+ = (\tau_1^+)^{n_1} \rightarrow ... \rightarrow (\tau_k^+)^{n_k} \rightarrow o \). For example \((o^k \rightarrow o)^+ = o^{k\cdot|Q| \cdot m_{\text{max}}} \rightarrow o\).

Symbols

We define

\[
  \Sigma' = \{ a^{n_k \rightarrow ... \rightarrow n_q \rightarrow n_m} : (o^k \rightarrow o)^+ \mid a : o^k \rightarrow o \in \Sigma, q_1, ..., q_k, q \in Q, m \leq m_{\text{max}} \},
\]

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

\[
  N' = \{ \text{Void}_x \mid \tau \in \text{Types}_{\mathcal{G}} \} \uplus \{ F^{(\theta, m)} : \tau^+ \mid F : \tau \in N, \theta : \tau, m \leq m_{\text{max}} \}.
\]

Terms

Given a term \( e = \tau \in T(V \cup \Sigma \cup N) \), a mapping \( \theta \) and an environment \( \Gamma \) 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 \( \theta = q_1 \rightarrow ... \rightarrow q_k \rightarrow q \in Q \), we define \( e^\Gamma, q = a^{q_1 \rightarrow ... \rightarrow q_k \rightarrow q_\Omega(q)} \).
- If \( e = F \in N \), then \( \Gamma = \{ F \triangleright (\theta, m)^b \} \) for some \( m \) and \( b \), we define \( e^\Gamma, \theta = F^{(\theta, m)} \).
- If \( e = x \in V \), then \( \Gamma = \{ x \triangleright (\theta, m)^b \} \) for some \( m \) and \( b \), we define \( e^\Gamma, \theta = x^{(\theta, m)} \).
- If \( e = e_1 \cdot e_2 \) with \( e_2 : \tau_2 \), then there exist \((\theta_1, m_1), ..., (\theta_k, m_k)\) and \( \Gamma_0, \Gamma_1, ..., \Gamma_k \) such that \( \Gamma_0 \cup \Gamma_1 \cup ... \cup \Gamma_k = \Gamma \), \( \Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \land ... \land (\theta_k, m_k) \rightarrow \theta \) and for all \( i \), \( \Gamma_i \vdash m_i \vdash t_i \triangleright \theta_i \). We define

\[
  e^\Gamma, \theta = e_1^{\Gamma_0, (\theta_1, m_1) \land ... \land (\theta_k, m_k) \rightarrow \theta} e_2^{\Gamma \rightarrow m_1, \theta_1} ... e_k^{\Gamma \rightarrow m_k, \theta_k} \text{Void}_{\tau_2} \ldots \text{Void}_{\tau_2}.
\]

Here \( \text{Void}_{\tau_2} \) is used to fill the missing arguments.
Rewrite Rules

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

\[ \theta = \bigwedge_i (\theta_{1,i}, m_{1,i}) \rightarrow ... \rightarrow \bigwedge_i (\theta_{k,i}, m_{k,i}) \rightarrow \theta' \]

such that \((F, \theta, m)\) is winning in the game. For all \(1 \leq j \leq k\) we define the tuple \(x_j' = (x_j^{(\theta_{j,i}, m_{j,i})}, ..., x_j^{(\theta_{j,i}, m_{j,i})}, v_{\tau_j}, ..., v_{\tau_j})\), that contains \(n_{\tau_j}\) elements of type \(\tau_j\) (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 \(\Gamma = \Gamma(F, \theta, m), x_1 \triangleright \bigwedge_i (\theta_{1,i}, m_{1,i}), ..., x_k \triangleright \bigwedge_i (\theta_{k,i}, m_{k,i})\) such that \(\Gamma \vdash e \triangleright \theta\). To the nonterminal \(F(\theta, m) \in N'\) we associate the rewrite rule

\[ F(\theta, m) \ x_1...x_k \rightarrow e^{\Gamma, \theta}. \]

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 \(H'(\theta', m') \in N'\) such that \((H, \theta', m')\) is not winning in the game, we define the rule \(H \ x_1...x_k' \rightarrow Void_o\) 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 \(Void_o \ x_1...x_{arity(\tau)} \rightarrow Void_o\).

Remarks

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

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

\[ a(q_1\rightarrow...\rightarrow q_k \rightarrow q, \Omega(q)) \ x_1...x_k \]

with

\[ t_k = t_k, Void_o, ..., Void_o \]

It means that every subtree of \(\|G'\|\) will have the structure

\[ a(q_1\rightarrow...\rightarrow q_k \rightarrow q, \Omega(q)) \ x_1...x_k \]

Finally we define the scheme \(G''\) obtained from \(G'\) by turning the terminals \(a(q_1\rightarrow...\rightarrow q_k \rightarrow q, m)\) as non terminals with the rewrite rules

\[ a(q_1\rightarrow...\rightarrow q_k \rightarrow q, m) \ x_1...x_k \rightarrow (a, q) \ x_1...x_k. \]

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

▶ Theorem 25. The tree \(\|G''\|\) is an accepting run of \(A\) on \(\|G'\|\).
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 \( t = \alpha \, t_1 \ldots t_k \) with \( \alpha \in \Sigma \cup \mathcal{N} \), we define inductively the OI parallel rewriting \( t^*_\text{OI} \) of \( t \):

\[ t^*_\text{OI} \]

- if \( \alpha \in \Sigma \) or \( k < \text{arity}(\alpha) \), then \( t^* = \alpha \, t^*_1 \ldots t^*_k \);

- if \( \alpha \in \mathcal{N} \) and \( k = \text{arity}(\alpha) \), let \( \alpha \, x_1 \ldots x_k \rightarrow e \) be the rewrite rule associated to \( \alpha \), we have \( t^* = e[\forall_i \, x_i \mapsto t_i] \).

Given \( t, t' \), we write \( t \Rightarrow_{\text{OI}} t' \) the relation “\( t' = t^* \)”. 

| 25 | A. Haddad |
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 \( t_1 \rightarrow t_2 \) 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 \( t_1 \rightarrow t_2 \).

For example let’s look at a term \( t_1 = a (F \ H) \) with the rewriting rule \( F \ x \rightarrow J \ x \ x \). The term \( t_1 \) can be rewritten in \( t_2 = a (J \ H \ H) \). 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 \( a \) 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 \( t \rightarrow t’ \) we associate a canonical context \( C[\ast] \) where the rewriting take place. This would avoid some ambiguity in the definitions.

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

  **Notation.** We may use subscript to denote occurrence of a symbol, e.g. \( \alpha_i \) would denote an occurrence of \( \alpha \). 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 \( C_{occ}[\ast] \) the associated context. We write \( Occ \) the set of occurrences in the term \( t \).

- **Definition 27 (Conservation Relation).** Given \( t_1, t_2 \) such that \( t_1 \rightarrow t_2 \) and let \( C[\ast] \) be the context associated to \( t_1 \rightarrow t_2 \). \( F \ in \Sigma, \ F \ x_1, ..., x_k \in \mathcal{R}, \) and \( s_1, ..., s_k \) be such that \( t_1 = C[F \ s_1, ..., s_k] \) and \( t_2 = C[\ast] \).

  Either there exist a two holes context \( C[\ast_1][\ast_2] \) such that

  \[
  C_{\alpha_1}[\ast] = C[\ast][F \ s_1, ..., s_k] \quad C_{\alpha_2}[\ast] = C[\ast][e[\forall_1 \ x_1 \rightarrow s_1]] \quad C[\ast] = C[\ast][\ast],
  \]

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

  \[
  C_{\alpha_2}[\ast] = \left(C_{occ}[C’[\ast]]\right)_{[\forall_1 \ x_1 \rightarrow s_1]}
  \]

  **Remark.** Given \( \alpha_2 \), there exists at most one \( \alpha_1 \) such that \( \alpha_1 \gg \alpha_2 \).

- **Definition 28 (Creation Relation).** Given \( t_1, t_2 \) such that \( t_1 \rightarrow t_2 \) and let \( C[\ast] \) be the context associated to \( t_1 \rightarrow t_2 \). \( F \ in \Sigma, \ F \ x_1, ..., x_k \in \mathcal{R}, \) and \( s_1, ..., s_k \) be such that \( t_1 = C[F \ s_1, ..., s_k] \) and \( t_2 = C[\ast] \).

  Given \( \alpha \ in \Sigma \uplus \mathcal{N} \) and \( \alpha_2 \) (resp. \( \alpha_2 \)) an occurrence of \( \alpha \) in \( t_2 \). Let \( F_1 \) be the occurrence \( (C_{F_1} = C[\ast, ..., s_k], F) \) of \( F \) in \( t_1 \), we say that \( \alpha_1 \ creates \ \alpha_2 \ in \ the \ rewriting \ t \rightarrow t’ \), written \( F_1 \gg t \rightarrow t’ \ \alpha_2 \) (or simply \( F_1 \gg \alpha_2 \) when the rewriting is clear from the context), if there exists a context \( C’[\ast] \) such that \( e = C’[\alpha] \) and

  \[
  C_{\alpha_2}[\ast] = C’[\ast]_{[\forall_1 \ x_1 \rightarrow s_1]}
  \]

  **Proposition 29.** Given \( t_1 \rightarrow t_2 \) and \( occ_2 \) an occurrence in \( t_2 \) there exist a unique occurrence \( occ_1 \) in \( t_1 \) such that either \( occ_1 \gg occ_2 \) or \( occ_1 \gg occ_2 \) (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 \( \alpha_2 \) in \( t_2 \), there are three disjoint possibilities.

1. Either there exist a two holes context \( C[\bullet][\bullet'] \) such that
   \[
   C_{\alpha_2}[\bullet] = C[\bullet][\{\forall i \ x_i \mapsto s_i\}]
   \]
   and \( C[\bullet] = C[\alpha][\bullet] \), in which case there is an occurrence \( (C[\bullet][F \ s_1...s_k], \alpha) \) in \( t_1 \) that gives \( \alpha_2 \).

2. Or there exist an occurrence \( \text{occ} = (C_{\text{occ}}[\bullet], x_j) \) of \( x_j \) in \( e \) and a context \( C'[\bullet] \) such that
   \[
   C_{\alpha_2}[\bullet] = \left( C_{\text{occ}}[C'[\bullet]] \right)_{\{\forall i \ x_i \mapsto s_i\}}
   \]
   in which case there is an occurrence \( (C[F \ s_1...C'[\bullet]...s_k], \alpha) \) in \( t_1 \) that gives \( \alpha_2 \).

3. Or there exists a context \( C'[\bullet] \) such that \( e = C'[\alpha] \) and
   \[
   C_{\alpha_2}[\bullet] = C[\left( C'[\bullet] \right)_{\{\forall i \ x_i \mapsto s_i\}}]
   \]
   in which case the occurrence \( (C'[s_1...s_k], F) \) creates \( \alpha_2 \).

\( \Box \)

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.

\( \triangleright \) **Definition 30 (Occurrences in derivation).** We define the set \( \text{Occ}_d \) of occurrences in \( \text{der} \) as:

\[
\text{Occ}_d = \{(i, \text{occ}_i) \mid \text{occ}_i \in t_i\}
\]

\( \triangleright \) **Definition 31 (Conservation relation in \( \text{Occ}_d \)).** We define the relation \( \triangleright_d \) (or simply \( \triangleright \) when the derivation is clear from the context) on \( \text{Occ}_d \times \text{Occ}_d \) as the smallest reflexive and transitive relation satisfying that if \( \text{occ}_i \) and \( \text{occ}_{i+1} \) are occurrences of respectively \( t_i \) and \( t_{i+1} \) such that \( \text{occ}_i \triangleright_{t_i \to t_{i+1}} \text{occ}_{i+1} \) then \( (i, \text{occ}_i) \triangleright_{\text{der}} (i + 1, \text{occ}_{i+1}) \).

\( \triangleright \) **Definition 32 (Creation relation in \( \text{Occ}_d \)).** We define the relation \( \triangleright_d \) (or simply \( \triangleright \) when the derivation is clear from the context) on \( \text{Occ}_d \times \text{Occ}_d \) as \( (i, \text{occ}_i) \triangleright_d (j, \text{occ}_j) \) if \( \text{occ}_i \) (resp. \( \text{occ}_j \)) is an occurrence of \( t_i \) (resp. \( t_j \)), and if there exists an occurrence \( \text{occ}_{i+1} \) in \( t_{i+1} \) such that \( \text{occ}_i \triangleright_{t_i \to t_{i+1}} \text{occ}_{i+1} \) and \( (i + 1, \text{occ}_{i+1}) \triangleright_d (j, \text{occ}_j) \).
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
\[C[\bullet] = (a, q) \quad t_1 \ldots \quad C'[\bullet] \quad t_k\]
with \((a, q) \in \Sigma \times Q\) and \(C'[\bullet]\) a \(\Sigma\)-context.

We associate to \(C[\bullet]\) a finite sequence \(b_{C[\bullet]}\) of \((\Sigma \times Q \cdot N)\) as if \(C[\bullet] = \bullet\), \(b_{C[\bullet]} = \varepsilon\), if 
\[C[\bullet] = (a, q) \quad t_1 \ldots \quad t_j - 1 \quad C'[\bullet] \quad t_{j+1} \ldots \quad t_k\]
then 
\[b_{C[\bullet]} = (a, q), j, b_{C'[\bullet]}\].

Definition 34 (Color of a sequence \(b\)). Given a finite sequence \(b\) of \((\Sigma \times Q \cdot N)\) and a state \(q\) (which represents 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 \(b = (a_0, q_0) j b'\) then \(\Omega(b, q) = \max(q_0, \Omega(b', q))\).

Definition 35 (Approximation). Given a tree over \(\Sigma \times Q\) and an infinite branch \(b = (a_0, q_0), j_0, (a_1, q_1), j_1, \ldots\). 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 \(S(q_0, \Omega(q_0)) \rightarrow^* t\), all non terminal \(F(\theta, m)\) occurring in \(t\) satisfies that \((F, \theta, m)\) is winning in the game.

Proof. First given a rewrite rule \(H(\theta', m') x_1 \ldots x_k \rightarrow e\) with \((H, \theta', m')\) 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 \((F, \theta, m)\) is winning in the game.

Then a simple induction on the size of the derivation \(S(q_0, \Omega(q_0)) \rightarrow^* t\) proves the lemma.

Lemma 37. Given a derivation \(S(q_0, \Omega(q_0)) = t_0 \rightarrow t_1 \rightarrow \ldots\) 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 
\[H \triangleright (\theta', m') \in \Gamma(F, \theta, m)\].

Proof. Same kind of simple proof as the previous lemma.
C.5 Main Proof

Lemma 38. Given a derivation \( S^{(q_0,\Omega(q_0))} = t_0 \rightarrow t_1 \rightarrow \ldots \), 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 exist two \( \Sigma \)-contexts \( C_1[\bullet] \) and \( C_2[\bullet] \) such that:
  \[
  C_1[H^{(\theta_0,m_0)}] = C_1[\bullet s_1 \ldots s_{\text{arity}(H^{(\theta_0,m_0)})}],
  \]
  \[
  C_2[F^{(\theta,m)}] = C_2[\bullet t_1 \ldots t_{\text{arity}(F^{(\theta,m)})}],
  \]
  \[
  b_{C_2[\bullet]} = b_{C_1[\bullet]} \cdot b'.
  \]
Then \( \Omega(b',q) = m \) with \( \theta = \Lambda_i(\theta_1,i,m_{1,i}) \rightarrow \ldots \rightarrow \Lambda_i(\theta_{k,i},m_{k,i}) \rightarrow q \).

Theorem 39 (Kobayashi, Ong). Given an infinite derivation \( S^{(q_0,\Omega(q_0))} = t_0 \rightarrow t_1 \rightarrow \ldots \) and an infinite branch \( b \) in the limit tree of this derivation, there exist an infinite strictly increasing sequence of indexes \( i_0 = 0 < i_1 < i_2 < \ldots \) and an infinite sequence of occurrences \((\text{occ}_{i_0},i_0),(\text{occ}_{i_1},i_1),\ldots\) such that:
- for all \( j \), \( (\text{occ}_{j},i_j) \succ (\text{occ}_{j+1},i_{j+1}) \),
- for all \( j \) \( \text{occ}_{\text{arity}(\bullet)} = C_j[\bullet s_1 \ldots s_{\text{arity}(\bullet)}] \) for some \( \Sigma \)-context \( C_j[\bullet] \) and terms \( s_1,\ldots,s_{\text{arity}(\bullet)} \),
- for all \( j \), let \( b_j = b_{C_j[\bullet]} \), then \( b_j \) is a prefix of \( b \),
- the sequence \( b_0,b_1,b_2,\ldots \) 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 \( \Omega(b'_0),\Omega(b'_1),\Omega(b'_2),\ldots \) 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 \( \Vert G'' \Vert \) and an infinite branch \( b \) in \( \Vert G'' \Vert \). Take an infinite sequence of occurrences \((\text{occ}_{0},0),(\text{occ}_{1},i_1),\ldots\) defined as in Theorem 39. We know that the maximum colour seen infinitely often in \( b \) is the greatest colour appearing infinitely often in \( \Omega(b'_0),\Omega(b'_1),\Omega(b'_2),\ldots \). Let \( F_i^{(\theta,m)} \) be the non terminal associated to the occurrence \( \text{occ}_i \). By definition, \( F_{i+1} \triangleright \{\theta_{i+1},m_{i+1}\}^t \) is in the environment picked by Eve from \( (F_i,\theta,i,m_i) \), then, since Eve wins in the game when she respects her strategy, the infinite colour seen infinitely often in \( m_0,m_1,\ldots \) 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.

\textbf{Proposition 40 (Substitution).} Given \( e \in T(\Sigma \cup \mathcal{N} \cup \mathcal{V}) \), \( x : \tau \in \mathcal{V} \), \( t : \tau \in T(\Sigma \cup \mathcal{N} \cup \mathcal{V}) \), if

\[
\Gamma_x = \Gamma_0, x \triangleright (\theta_1, m_1)^\tau, \ldots, x \triangleright (\theta_k, m_k)^\tau \vdash e \triangleright \theta,
\]

\[
\forall i \Gamma'_i = \Gamma_i \uparrow m_i \vdash t \triangleright \theta_i,
\]

then

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

and if we let \( \Gamma = \Gamma_0 \cup \Gamma_1 \cup \ldots \cup \Gamma_k \) we have

\[
\left( e_{[x \mapsto t]} \right)^{\Gamma, \theta}_{x(i, m_i) \mapsto t'_i, s_i} = (e_{[x \mapsto \theta]})^{\Gamma, \theta}
\]

\textbf{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

\[
t^{\Gamma, \theta} = t^{\Gamma'_m, \theta}.
\]

\textbf{Proof.} Straightforward induction on the derivation of \( \Gamma \vdash t \triangleright \theta \).

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

\textbf{Proof.} The proof proceeds by induction on the derivation of \( \Gamma \vdash t \triangleright \theta \).

- If the last rule applied is (T-VAR) then \( t = a \in \Sigma \), \( \Gamma = \emptyset \) and the result is trivial.

- If the last rule applied is (T-Const) then \( t = a \in \mathcal{N} \), \( \Gamma = \emptyset \) and the result is trivial.

- If the last rule applied is (T-App) then \( t = t_0 \Gamma_0 \Gamma_1 \Gamma_2 \ldots \Gamma_k \) with \( \Gamma_0 \vdash t_0 \triangleright (\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \rightarrow \theta \) 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 \( (\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \rightarrow \theta \) 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 \( \Gamma_1 \uparrow \Omega(\theta) \) is well defined, therefore \( \Gamma \uparrow \Omega(\theta) = \Gamma_0 \uparrow \Omega(\theta) \cup \Gamma_1 \uparrow \Omega(\theta) \cup \ldots \cup \Gamma_k \uparrow \Omega(\theta) \) is well defined.

If \( \Gamma \uparrow \Omega(\theta) = \Gamma' \uparrow \Omega(\theta) \) then there exists \( \Gamma'_0 \cup \Gamma'_1 \cup \ldots \cup \Gamma'_k = \Gamma' \) 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 \( i \) \( \Gamma_i \uparrow m_i = \Gamma_i \uparrow m_i \uparrow \Omega(\theta) = \Gamma_i \uparrow \Omega(\theta) \uparrow m_i = \Gamma'_i \uparrow \Omega(\theta) \uparrow m_i = \Gamma'_i \uparrow m_i \uparrow \Omega(\theta) = \Gamma'_i \uparrow m_i \uparrow t_i \triangleright \theta_i \).

Furthermore by induction hypothesis, since \( \Omega((\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \rightarrow \theta) = \Omega(\theta) \) we have \( \Gamma'_0 \vdash t_0 \triangleright (\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \rightarrow \theta \), hence \( \Gamma' \vdash t \triangleright \theta \).

Also by induction hypothesis,

\[
t^{\Gamma'_0, \theta}_{(\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \rightarrow \theta} = t^{\Gamma_0, (\theta_1, m_1) \land \ldots \land (\theta_k, m_k) \rightarrow \theta}
\]

and since \( \Gamma_i \uparrow m_i = \Gamma'_i \uparrow m_i \), for all \( i \), \( t^{\Gamma_i, \theta}_i = t^{\Gamma'_i, \theta}_i \) then by definition \( t^{\Gamma, \theta} = t^{\Gamma', \theta} \).
Given $b \in \{\mathsf{t, f}\}$, we define $\Gamma \uparrow_b m$ by:

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

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

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

$$
(\Gamma_0 \searrow \mathcal{N}) \hat{} \ldots \hat{} \mathcal{N}, (\Gamma_k \searrow \mathcal{N}) \hat{} \ldots \hat{} \mathcal{N} \vdash e \triangleright \theta,
$$

then $\Gamma_0 \cup \Gamma_1 \cup \ldots \cup \Gamma_k \vdash e_{[x \mapsto t]} \triangleright \theta$, and if we let $\Gamma_x = \Gamma_0 \searrow \mathcal{N}, ..., \searrow \mathcal{N}, (\Gamma_k \searrow \mathcal{N}) \hat{} \ldots \hat{} \mathcal{N}$, $\Gamma = \Gamma_0 \cup \Gamma_1 \cup \ldots \cup \Gamma_k$ and for all $i, \Gamma_i = \Gamma_i \uparrow_{b_i} m_i \vdash e \triangleright \theta_i$.

**Proof.** The proof proceeds by induction on the derivation of $\Gamma_0 \searrow \mathcal{N}, ..., \searrow \mathcal{N}, (\Gamma_k \searrow \mathcal{N}) \hat{} \ldots \hat{} \mathcal{N}$.

- 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 $\Gamma_0 = \emptyset, k = 1, \theta = \theta_1, (\theta_1, m_1)^t \uparrow \Omega(\theta) = (\theta_1, m_1)^x$, and $\Gamma_1 \uparrow_{b_1} m_1 \vdash t \triangleright \theta_1$.

If $b_1 = \mathsf{t}$ then $\Gamma_1 \uparrow_{b_1} m_1 = \Gamma$ therefore $\Gamma_1 \vdash t \triangleright \theta$. Furthermore $e_{[\theta_1, m_1^x]}(\theta_1, m_1^x) \rightarrow x(\theta_1, m_1^x)$ therefore

$$
eq \bar{i}(\{\theta, m_1^x\}) \triangleright \theta_x \rightarrow x(\theta_1, m_1^x)
$$

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

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

$$
eq e_0 e_1
$$

$$
\Gamma_0 = \Delta_0 \cup \Delta_1 \cup \ldots \cup \Delta_k
$$

$$
S_0 \cup S_1 \cup \ldots \cup S_k = \{1, ..., k\}
$$

$$
\theta' = (\theta'_1, m'_1) \wedge \ldots \wedge (\theta'_k, m'_k) \rightarrow \theta
$$

$$
\Delta_0^x = \Delta_0 \cup \{x \triangleright (\theta_1, m_1^x) \mid x \in S_0\} \vdash e_0 \triangleright \theta'
$$

$$
\Delta_j^x = (\Delta_j \cup \{x \triangleright (\theta_1, m_1^x) \mid x \in S_j\}) \uparrow m^x_j \vdash e_1 \triangleright \theta_j
$$

$$
\Gamma_i \uparrow_{b_i} m_i \vdash t \triangleright \theta_i
$$
From the induction hypothesis, we have
\[ \Delta'_0 = \Delta_0 \cup \bigcup_{i \in S_0} \Gamma_i \vdash e_0 \triangleright (\theta'_1, m'_1) \land \ldots \land (\theta'_t, m'_t) \rightarrow \theta, \]
\[ \left( e_0^{\Delta'_0, \theta'} \right) = (e_0^{\Delta_0, \theta'})^{\Delta'_0, \theta'}. \]

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
\[ \left( e_0^{\Delta'_j, \theta'_j} \right) = (e_0^{\Delta_j, \theta_j})^{\Delta'_j, \theta'_j}, \]
which by simple application would conclude.

For \( i \in S_j \), we define \((\theta_i, m_i)^{\theta_i, j} = (\theta_i, m_i)^{\theta_i} \uparrow m'_j \). In particular we have:

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

Since \((\theta_i, m_i)^{\theta_i} \uparrow m'_j \) is well defined, we have \( m'_j \leq m_i \), and since \( \Gamma_i \uparrow b_i \) \( m_i \) is well defined, we get that \( m'_j \) is less or equal than any color in \( \Gamma_i \).

Furthermore since \( \Gamma_i \uparrow b_i \) \( m_i \uparrow t \triangleright \theta_i \), Lemma 41 states that \( \Gamma_i \uparrow b_i \) \( m_i \uparrow m'_j \) = \( \Gamma_i \uparrow m'_j \uparrow b_i \) \( m_i \uparrow t \triangleright \theta_i \) (indeed, one can commute the \( \uparrow \)) and \( t^{\Gamma_i \uparrow b_i \theta_i} = t^{\Gamma_i \uparrow m'_j \uparrow b_i \theta_i}. \)

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. \( \Gamma_i \uparrow m'_j \uparrow b_i, j \) \( m_i \uparrow t \triangleright \theta_i, \)

and

3. \( t^{\Gamma_i \uparrow b_i \theta_i} = t^{\Gamma_i \uparrow m'_j \uparrow b_i \theta_i} = t^{\Gamma_i \uparrow m'_j \uparrow b_i, j \theta_i}. \)

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
\[ \left( e_0^{\Delta'_j, \theta'_j} \right) = (e_0^{\Delta_j, \theta_j})^{\Delta'_j, \theta'_j}. \]
### C.7 Proof of Lemma 38

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

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

- If $t$ is a redex, $\Gamma_{C[\_]} \subseteq \{ F \triangleright (\theta, m)^b | (F, \theta, m) \text{ winning}, b \in \{ f, t \}\}$,
- if $t$ is a redex, $\Gamma_{C[\_]} \vdash t_{C[\_]} \triangleright q$ for some $q_{C[\_]} \in Q$,
- If $t_{C[\_]} = a s_1 \ldots s_k$ let $q = q_{C[\_]}$ and for all $j$, let $q_j = q_{C[\_][a s_1 \ldots s_{j-1} s_{j+1} \ldots s_k]}$, we have
  
  $$q \xrightarrow{a} (q_1, \ldots, q_k) \in \delta.$$

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

$$\Gamma \subseteq \{ F \triangleright (\theta, m)^b | (F, \theta, m) \text{ winning}, b \in \{ f, t \}\}$$

then we can construct a painting of $t$.

**Proof.** Simple induction on the derivation. \hfill ◀

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

**Proof.** Take $\Gamma \vdash F s_1 \ldots s_k \triangleright q \ast$. We know that there exists

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

$(\theta, m)^b$ 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 $(F, \theta, m)$ winning in the game, then for all $j$ there exists $I_j \subseteq J_j$ such that

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

Then using substitution property we know that

$$\Gamma' = \Gamma_0 \cup \bigcup_{j,p \in I_j} \Gamma_{j,p} \vdash e[\forall x_i \mapsto s_i] \triangleright q \ast,$$

And the previous claim concludes. \hfill ◀

**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 \( q_* \) in \( t' \) as \( q_* \) 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 \( \{ S \vdash (q_0, \Omega(q_0)) \} \vdash S \vdash q_0 \) and we define the other paintings by induction on \( i \).

▶ Definition 49 (Term Transformation). Given a term \( t : o \) painted by \( \pi \). We define the term \( t^{++} \) in \( G' \) 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

\[
t^{++} = a(q_1,m_1)\vdash...\vdash(q_k,m_k)\rightarrow q t_1^{++}...t_k^{++},
\]

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 \( t_i^{++} = t_i^{++} Void_o...Void_o \).

Given a term \( t : o \) painted by \( \pi \). We define the term \( t^+ \) in \( G'' \) 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 \( \pi \) of \( t \), with \( F x_1...x_k \rightarrow e \), and \( t' = c_{[1 x_1\rightarrow s_1]} \), then \( t'^{++} = t^{\pi',q} \) with \( \pi' \) defined as in Proposition 46.

Proof. A simple induction.

▶ Definition 51 (The rewriting \( \Rightarrow_\Sigma \)). Given a term \( t \) in \( G'' \) we define the term \( t \Rightarrow_\Sigma t' \) as the one obtained by rewriting at once all the \( a^0 \).

▶ 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 \( G'' \) we define its canonic rewriting \( t' \) by \( t \Rightarrow_+ t' \Rightarrow_\Sigma t' \).

▶ Proposition 54. Given a term \( t : o \) painted by \( \pi \). 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 \( \Gamma \vdash F t_1...t_k \rightarrow q \) with \( \Gamma \subseteq \{ H \vdash (\theta, m)^b \mid (H, \theta, m) \) winning\}. We know that there exists

\[
\theta = \bigwedge_{p \in J_1} (\theta_1,p,m_1,p) \rightarrow ... \rightarrow \bigwedge_{p \in J_d} (\theta_k,p,m_k,p) \rightarrow q,
\]

\((\theta, m)^b \) and \( \Gamma_j,p \) for all \( j \) and \( p \in J_j \) such that

\[
\{ F \vdash (\theta, m)^b \} \vdash F \vdash \theta,
\]

for all \( j \) and \( p \in J_j \)

\[
\Gamma_j,p \vdash m_j,p \vdash s_j \vdash \theta_j,p,
\]

and

\[
\Gamma = \{ F \vdash (\theta, m)^b \} \cup \bigcup_{j,p} \Gamma_j,p.
\]
We have \((F, \theta, m)\) winning in the game, then for all \(j\) there exists \(I_j \subseteq J_j\) such that
\[\Gamma_0 = \Gamma_{(F, \theta, m)} \cup \{x_j \triangleright (\theta, p, m, \theta, m) \mid p \in I_j\} \vdash e \triangleright q.\]

Then using substitution property we know that
\[\Gamma' = \Gamma_{(F, \theta, m)} \cup \bigcup_{j, p \in I_j} \Gamma_{j, p} \vdash e_{[x_j \triangleright s_j]} \triangleright q_\ast,\]
and
\[(e^{\Gamma_{j, p}, \theta})_{[x_j \triangleright s_j]} = (e_{[x_j \triangleright s_j]}^{\Gamma_{j, p}, \theta}) = (e^{\Gamma_{j, p}, \theta}).\]

Since \((t')^{+ +} = (e_{[x_j \triangleright s_j]})^{\Gamma_{j, p}, \theta}\)
and
\[t' \Rightarrow (e^{\Gamma_{j, p}, \theta})_{[x_j \triangleright s_j]},\]
we have
\[t' \Rightarrow (t')^{+ +} \Rightarrow_{\Sigma} (t')^{+}.\]

\begin{proposition}
The derivation \(S^{(q_0, R(q_0))} \Rightarrow \ast \Rightarrow_{\Sigma} t_1 \Rightarrow \ast \Rightarrow_{\Sigma} t_2 \Rightarrow \ast \Rightarrow_{\Sigma} \ldots\) is maximal.
\end{proposition}

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

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

\begin{lemma}
Given a term \(t : a\), such that \(\Gamma \vdash t \triangleright q\) and let \(\pi\) be the painting on \(t\) induced by \(\Gamma\).
\end{lemma}

\textbf{Proof.} Simple Induction.

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

\begin{theorem}
Let \((\Gamma_0, q_0)\) be the painting of \(C[\ast]\) in \(t_i\) and \((\Gamma, q)\) be the painting of \(C'[\ast]\).
And let \(F \triangleright (\theta, m)^{b_0}\) be the binding in \(\Gamma\) used for \((F_j, j)\) in the proof of \(\Gamma \vdash F \circ s_1 \ldots s_k \triangleright q\). Then \(F \triangleright (\theta, m)^{b_0}\) in \(\Gamma_0\) and
\[\begin{array}{ll}
\text{if } b_0 = f, \Omega(b', q) = m, \\
\text{if } b_0 = x, \Omega(b', q) \leq m.
\end{array}\]
\end{theorem}

\textbf{Proof.} The proof proceeds by induction on \(\ell\) and on \(b'\). If \(\ell = 0\), we have \(q_0 = q\), \(\Gamma_0 = \Gamma\) and \(b' = e\). Thus looking at the proof of \(\Gamma \vdash F \circ s_1 \ldots s_k \triangleright q\) concludes. If \(s = a s_1 \ldots s_k\) 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.