Light Multiset path ordering and Ptime - Two is better than one

Jean-Yves Marion

Light multiset path ordering and PTIME
Two is better than one

J.Y. Marion*

March 29, 1999

Abstract

We consider term rewriting systems as a general setting for studying algorithms. We construct a termination ordering, called *light multiset path ordering* (LMPO), which is a restriction of the *multiset path ordering*. We establish that the class of programs on words which is terminating by LMPO, characterises exactly the functions computable in polynomial time.

1 Introduction

The main issue is to express the computational complexity of a program defined by a term rewriting system from its termination proof. We consider here ways of specifying the computational complexity of algorithms which is an intentional property. The traditional meta-theory of program reasoning deals with the verification of extensional properties. The study of intentional properties of programs turns out to be much more challenging. For example, the class of all algorithms running, say, in polynomial time is not recursively enumerable with respect to an enumeration of partial recursive algorithms. But, imposing an intentional property on program constructions is a relevant question which is considered for example in Nuprl development as explained in [5, 2]. One might wonder how relevant is a program whose proof termination is certified but runs in doubly exponential time.

We construct an ordering based on *multiset path orderings* (MPO) introduced in [6], that we call *light multiset path ordering* (LMPO). We shall

*Loria, Calligramme project, B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France `Jean-Yves.Marion@loria.fr`. The author is very grateful to Adam Cichon for discussions.

demonstrate a LMPO-program which is terminating by LMPO, is computable in polynomial time. Inversely, each polynomial time functions, i.e. in PTIME, are represented by a LMPO-program. For this, we exploit traditional methods from termination proofs, and we incorporate ideas involved in predicative analysis of recursive definitions which can be found in the works of [16, 1, 14]. However, we are straying out of those minimal characterisations of PTIME, which mainly concern functions. In contrast, we shall investigate a syntactic characterisation of algorithms, by looking for a class which would capture "most of the good algorithms". As already explained, the definition of "good algorithms" is beyond the scope of this paper. Nevertheless, we can point out some significant algorithms in our characterisation. For example, we capture algorithms which have an exponential recursive definition, but for which a dynamic programming method reduces the complexity to polynomial time.

Related works are [3, 15]. There, rewriting systems admitting a polynomial interpretation termination proofs are considered. It is established that computational complexity of a program is given by the constructor interpretation involved in the termination proof. Caseiro, in the unpublished work [4], also attempts to provide syntactic criteria to control the time complexity of a program.

This paper is organised as follows. Section 2 defines functions computed by rewrite systems and multiset path orderings. The construction of light multiset path orderings is given in Section 3. The main result is presented in Theorem 3. The proof is established by Theorem 4 and 5. The proof of Theorem 5 is first outlined. We give the details of the proof, which is a bit tedious, in Appendices.

2 Programming with rewriting systems

2.1 Programs and functions over words

We shall concentrate over programs computed by a term rewriting system over words. The set of terms built up from a signature $\mathbb{S}$ and from a set of variables $\mathcal{V}$ is $\mathcal{T}(\mathbb{S}, \mathcal{V})$. A program, is defined by a quadruplet $\langle \rightarrow, \mathbb{W}, \mathbb{F}, f \rangle$ thus. The set of data of the program is the term algebra $\mathcal{T}(\mathbb{W})$ where symbols in $\mathbb{W}$ are called constructors. We shall always assume that $\mathbb{W}$ contains, at least, a 0-ary constructor, i.e. a constant, and all other constructors are unary, i.e. successors. For example, binary words will be represented by terms built up from $\{\epsilon, s_0, s_1\}$ where ϵ is a constant denoting the empty words, and s_0,s_1 are two successors. $\mathbb{F}$ is the set of function symbols of fixed

arity $ar(f)$. So, the full signature is $\mathbb{S} = \mathbb{W} \cup \mathbb{F}$. Rewrite rules are given by the binary relation $\rightarrow$. Each rewrite rule is of the form $g(t_1, \cdots, t_n) \rightarrow s$ where $g \in \mathbb{F}$ and the t_i's and s are terms of $\mathcal{T}(\mathbb{W} \cup \mathbb{F}, \mathcal{V})$. Moreover, $\mathrm{FV}(g(t_1, \cdots, t_n)) \subseteq \mathrm{FV}(s)$ where $\mathrm{FV}(s)$ is the set of variables in s. The function symbol $f \in \mathbb{F}$ is the main function symbol.

We define $u \Rightarrow v$ to say that the term v is obtained from u by applying a rewrite rule. The relation $\stackrel{+}{\Rightarrow}$ ($\stackrel{*}{\Rightarrow}$) denotes the transitive (reflexive-transitive) closure of $\Rightarrow$. We write $s \stackrel{!}{\Rightarrow} t$ to mean that $s \stackrel{*}{\Rightarrow} t$ and t is in normal form.

Say that a program is confluent if the relation $\rightarrow$ is confluent. To give a semantic to programs, we just consider, as meaningful, the normal forms which are in the data set $\mathcal{T}(\mathbb{W})$.

Definition 1. A confluent program $\langle \rightarrow, \mathbb{W}, \mathbb{F}, f \rangle$, where $ar(f) = n$, computes the function $\{f\} : \mathcal{T}(\mathbb{W})^n \mapsto \mathcal{T}(\mathbb{W})$ which is defined as follows. For all $u_1, \cdots, u_n \in \mathcal{T}(\mathbb{W})$, $\{f\}(u_1, \cdots, u_n) = v$ if $f(u_1, \cdots, u_n) \stackrel{!}{\Rightarrow} v$, otherwise $\{f\}(u_1, \cdots, u_n)$ is undefined.

A program is terminating if there is no infinite derivation, that is there is no infinite sequence of terms such that $t_0 \Rightarrow t_1 \Rightarrow t_2 \Rightarrow \cdots$ One might consult [7] for a survey about rewriting termination, and [8] about general references on rewriting.

2.2 Multiset Path Ordering

In this section we briefly describe *Multiset Path Orderings* (MPO) introduced by N. Dershowitz [6]. First of all, a multiset M of terms of $\mathcal{T}(\mathbb{S})$ is a finite mapping $M : \mathcal{T}(\mathbb{S}) \mapsto \mathbb{N}$ which associates to each term t the number $M(t)$ of terms t in M. Suppose that $\mathcal{T}(\mathbb{S})$ is ordered by $\prec$. Following [9], this ordering $\prec$ induces an ordering $\prec^{\mathrm{multi}}$ on term multisets.

Definition 2. $M \prec^{\mathrm{multi}} N$ iff $M \neq N$ and if there is $s \in \mathcal{T}(\mathbb{S})$ such that $M(s) > N(s)$ then there is $t \in \mathcal{T}(\mathbb{S})$ such that $s \prec t$ and $M(t) < N(t)$.

Take as the subterm ordering $\lhd$. Then we have $\{s(x), x, x\} \lhd^{\mathrm{multi}} \{s(x), s(x)\}$, for example.

We shall always consider an equivalence relation $\approx$ on symbols of $\mathbb{S}$ which respects symbol arities, that is if $f \approx g$, then we have $ar(f) = ar(g)$. A *permutative congruence* is an extension of $\approx$ to terms of $\mathbb{S}$ defined by $f(t_1, \cdots, t_n) \approx g(s_1, \cdots, s_n)$ if $f \approx g$ and $t_i \approx s_{\pi(i)}$ for some permutation π over $\{1, \cdots, n\}$. Define the pre-order $\preceq_{\mathbb{S}}$ by $\prec_{\mathbb{S}} \cup \approx$ where $\prec_{\mathbb{S}}$ is a strict order on $\mathbb{S}$. We shall say that $(\prec_{\mathbb{S}})$ $\preceq_{\mathbb{S}}$ is a (resp. strict) precedence on $\mathbb{S}$.

Definition 3. Let $\preceq_{\mathbb{S}}$ be a precedence on the signature $\mathbb{S}$. The multiset path ordering $\prec_{mpo}$ is defined recursively by

1. If $s \preceq_{mpo} t_i$ then $s \prec_{mpo} f(\cdots, t_i, \cdots)$.
2. If $g \prec f$ and for all $i \leq m$, $s_i \prec_{mpo} f(t_1, \cdots, t_n)$, then $g(s_1, \cdots, s_m) \prec_{mpo} f(t_1, \cdots, t_n)$.
3. If $g \approx f$ and if $\{s_1, \cdots, s_n\} \prec_{mpo}^{\text{multi}} \{t_1, \cdots, t_n\}$, then $g(s_1, \cdots, s_n) \prec_{mpo} f(t_1, \cdots, t_n)$.

where $\preceq_{mpo} = \prec_{mpo} \cup \approx$.

A confluent program $\langle \rightarrow, \mathbb{W}, \mathbb{F}, f \rangle$ is terminating by MPO if there is a precedence $\preceq_{\mathbb{S}}$ on $\mathbb{S} = \mathbb{W} \cup \mathbb{F}$ such that for each ground substitution σ and each rule $l \rightarrow r$, we have $r\sigma \prec_{mpo} l\sigma$. Hofbauer has shown in [10] that

Theorem 1. *Functions computed by confluent programs which are terminating by* MPO*, are exactly the primitive recursive functions.*

3 Light multiset path ordering

We shall now describe a restriction of MPO that we call *Light multiset path ordering* (LMPO)[1]. We shall all along make a clear distinction between function symbols of $\mathbb{F}$ and constructors of $\mathbb{W}$. In particular, we shall always assume that there is a strict precedence $\prec_{\mathbb{F}}$ on $\mathbb{F}$. On the contrary, constructors of $\mathbb{W}$ will be incomparable and that the only equivalence relation $\mathbb{W}$ will be the syntactic equality.

3.1 Valence and congruence.

We define the valence of a function symbol f of arity n as a mapping $\nu(f) : \{1, \cdots, n\} \mapsto \{0, 1\}$. The valence of a signature $\mathbb{F}$ is given by the valence of each function symbol in $\mathbb{F}$. Valences will give the ability of combining two kind of orderings to prove the termination of a program. Valences are somehow related to Kamin and Levy [12] notion of functionals on orders. Indeed, a functional on orderings can be defined as a status function on $\mathbb{F}$ which indicates how to compare terms, either in a lexicographic or in a multiset way.

Above all, the notion of valences resembles to the tiering of data which was introduced by Leivant in [13, 14]. But data tiering discipline enforces

[1] The terminology 'Light' is taken from the Light Linear Logic of J-Y Girard

that the type of terms are also tiered. Actually, function valences are much more like normal and safe position arguments as it is presented by Bellantoni and Cook in [1]. Function valences generalise this concept to functions defined by means of equations. For easier readability, we use a notational convention similar to that of [1], and write $f(x_1, \cdots, x_n; y_1, \cdots, y_m)$, with a semi-colon separating two lists of arguments, to indicate that $\nu(f, i) = 1$ for $i \in \{1, \cdots, n\}$, and $\nu(f, n + j) = 0$ for $j \in \{1, \cdots, m\}$.

Notice that we shall consider throughout valences on function symbols of $\mathbb{F}$, but we shall never assign valences to constructors of $\mathbb{W}$. Constructors of $\mathbb{W}$ will be always treated on their own.

Let $\approx$ be an equivalence relation on $\mathbb{F}$. A permutative congruence $\approx_0$, on the set of terms $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$, which respects the valence ν, is a permutative congruence based on the extension of $\approx$ which satisfies:

1. $c(t) \approx_0 c(t')$ if $c \in \mathbb{W}$ and $t \approx_0 t'$.
2. $f(t_1, \cdots, t_n) \approx_0 g(s_1, \cdots, s_n)$ if there is a permutation π such that f, g are in $\mathbb{F}$ and $f \approx g$, $t_i \approx_0 s_{\pi(i)}$ and $\nu(f, i) = \nu(g, \pi(i))$, for some permutation π.

Example 1. Take $\mathbb{W} = \{s_0, s_1, \epsilon\}$ as constructor set. Suppose that $\mathbb{F} = \{f\}$ where $ar(f) = 4$. Define ν by $\nu(f, 1) = \nu(f, 2) = 1$ and $\nu(f, 3) = \nu(f, 4) = 0$. We have $s_0(f(a, b; c, d)) \approx_0 s_0(f(b, a; d, c))$. But $s_0(f(a, b; c, d)) \not\approx_0 s_0(f(b, d; a, c))$ because $\nu(f, 1) \neq \nu(f, 3)$.

Let $\prec_{\mathbb{F}}$ be a strict precedence on $\mathbb{F}$ and $\approx$ be an equivalence relation which respects ν on $\mathbb{F}$. Then, we say that the pre-order $\preceq_{\mathbb{F}}$ defined by $\prec_{\mathbb{F}} \cup \approx$ is a precedence which respects the valence ν on $\mathbb{F}$. Wlog, we shall always that $\preceq_{\mathbb{F}}$ is total on $\mathbb{F}$.

3.2 A restricted embedding relation.

Say that a term of $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$ is of valence 1 if each function symbol of $\mathbb{F}$ which occurs in it has a valence function which satisfied $\nu(f, i) = 1$ for all $i \leq ar(f)$. Define the permutative congruence $\approx_1$ as the restriction of $\approx_0$ to term of valence 1.

Definition 4. Let $\mathbb{W}$ be a set of constructors. For any precedence $\preceq_{\mathbb{F}}$ which respects the valence ν on $\mathbb{F}$, the embedding relation $\prec_1$ on terms of $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$ is recursively defined as follows:

1. If $(s \preceq_1 t)$ and if $c \in \mathbb{W}$, then $s \prec_1 c(t)$.

2. If $c \in \mathbb{W}$ and $s \prec_1 f(t_1, \cdots, t_m)$, then $c(s) \prec_1 f(t_1, \cdots, t_m)$.

3. If $(s \preceq_1 t_i)$ and if $\nu(f, i) = 1$ then $s \prec_1 f(\cdots, t_i, \cdots)$.

4. If $(g \prec_{\mathbb{F}} f)$ and if for all $i \leq n$, $s_i \prec_1 f(t_1, \cdots, t_m)$ and $\nu(g, i) = 1$,
then $g(s_1, \cdots, s_n) \prec_1 f(t_1, \cdots, t_m)$.

where $\preceq_1 = \prec_1 \cup \approx_1$, and $f \in \mathbb{F}$.

Examples 2.

1. Two distinct terms of $\mathcal{T}(\mathbb{W})$ are incomparable by $\prec_1$, but if s is a subterm of t, and if both s and t are in $\mathcal{T}(\mathbb{W})$, then $s \preceq_1 t$. Hence, $\{a, c(b)\} \prec_1^{\text{multi}} \{c(a), c(b)\}$ where $c \in \mathbb{W}$.

2. $\texttt{mult}(a, a;) \prec_1 \texttt{cube}(a;)$ if $\texttt{mult} \prec_{\mathbb{F}} \texttt{cube}$.

An important observation is that if $t \preceq_1 f(a)$ then f does not occur in the term t. Notice also that the subterm relation on terms of valence 1 is embedded in $\preceq_1$.

3.3 Light multiset path ordering

Definition 5. Let $\mathbb{W}$ be a set of constructors. Let $\preceq_{\mathbb{F}}$ be a precedence which respects the valence ν on $\mathbb{F}$. The *Light multiset path ordering* (LMPO) $\prec_0$ on terms of $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$ is recursively defined as follows:

1. If $s \preceq_0 t$ and if $c \in \mathbb{W}$, then $s \prec_0 c(t)$.

2. If $s \preceq_k t_i$ and if $\nu(f, i) = k$ then $s \prec_0 f(\cdots, t_i, \cdots)$, for $k = 0, 1$.

3. If $c \in \mathbb{W}$ and $s \prec_0 f(t_1, \cdots, t_m)$, then $c(s) \prec_0 f(t_1, \cdots, t_m)$.

4. If $g \prec_{\mathbb{F}} f$ and for all $i \leq n$, if $\nu(g, i) = 1$ and $s_i \prec_1 f(t_1, \cdots, t_m)$ or if $\nu(g, i) = 0$ and $s_i \prec_0 f(t_1, \cdots, t_m)$, then $g(s_1, \cdots, s_n) \prec_0 f(t_1, \cdots, t_m)$.

5. Suppose $\nu(f, i) = 1$ for some i.
If $g \approx f$ and if $\{s_i \ : \ \nu(g, i) = 1\} \prec_1^{\text{multi}} \{t_i \ : \ \nu(f, i) = 1\}$
and $\{s_i \ : \ \nu(g, i) = 0\} \preceq_0^{\text{multi}} \{t_i \ : \ \nu(f, i) = 0\}$
then $g(s_1, \cdots, s_n) \prec_0 f(t_1, \cdots, t_n)$.

where $\preceq_0 = \prec_0 \cup \approx_0$, and $f, g \in \mathbb{F}$.

Examples 3.

1. Put $\mathbb{N} = \{0, s\}$. We have $s(s(\mathtt{d}(x;));) \prec_0 \mathtt{d}(s(x);)$. But $\mathtt{d}(\mathtt{exp}(x;);) \not\prec_0 \mathtt{exp}(s(x);)$ because even if $\mathtt{d} \prec_{\mathbb{F}} \mathtt{exp}$, $\mathtt{exp}(x;) \not\preceq_1 \mathtt{exp}(s(x);)$. This examples illustrates the fact that the exponential function, as defined by the rules below, is not ordered by $\prec_0$. (But $\mathtt{d}$ is ordered.)

$$\begin{aligned} \mathtt{d}(0;) &\to 0 \\ \mathtt{d}(s(x);) &\to s(s(\mathtt{d}(x;))) \\ \mathtt{exp}(0;) &\to s(0) \\ \mathtt{exp}(s(x);) &\to \mathtt{d}(\mathtt{exp}(x;);) \end{aligned}$$

2. Two distinct terms of $\mathcal{T}(\mathbb{W})$ are incomparable.

3. $h(; f(a, s(b);), f(s(a), b;)) \prec_0 f(s(a), s(b);)$ if $h \prec_{\mathbb{F}} f$.
Indeed, e.g. $f(a, s(b);) \prec_0 f(s(a), s(b);)$ since $\{a, s(b)\} \prec_1^{\text{multi}} \{s(a), s(b)\}$.

It turns out that the permutative congruence induced by $\preceq_0$ is exactly $\approx_0$. The full signature of the rewriting system is $\mathbb{S} = \mathbb{W} \cup \mathbb{F}$. The distinction between $\mathbb{W}$ and $\mathbb{F}$ is made through the precedence on $\mathbb{S}$. Indeed, we could say that $\mathbb{S}$ is ordered by the strict precedence $\prec_{\mathbb{S}}$ which would be defined as the smallest relation (i) containing $\prec_{\mathbb{F}}$ and (ii) such that $c \prec_{\mathbb{S}} f$ for all $c \in \mathbb{W}$ and $f \in \mathbb{F}$. In other words, constructors are the least elements for $\prec_{\mathbb{S}}$ and are incomparable with respect to $\prec_{\mathbb{S}}$. Both conditions are crucial.

Actually, $\prec_1 \subset \prec_0$. Moreover, for every term s and t of $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$, if $s \prec_0 t$ holds then we have, by forgetting symbol valences, $s \prec_{mpo} t$, where $\prec_{mpo}$ is the ordering induced by $\prec_{\mathbb{S}}$. As a consequence, if the rules of a program are ordered by a $\prec_0$ then the program is terminating.

3.4 Characterisation of Ptime

Definition 6. A LMPO-program is defined by a tuple $\langle \to, \mathbb{W}, \mathbb{F}, f, \nu, \prec_{\mathbb{F}} \rangle$, where $\langle \to, \mathbb{W}, \mathbb{F}, f \rangle$ is a confluent program, ν is a valence function on $\mathbb{F}$, $\preceq_{\mathbb{F}}$ is a precedence which respects ν, and such that the induced ordering $\prec_0$ satisfies: for every ground substitution σ and for each rule $l \to r$, $r\sigma \prec_0 l\sigma$.

Theorem 2. *Each* LMPO*-program is terminating.*

Examples 4.

1. Consider the LMPO-program $\langle \to, \mathbb{N} = \{0, s\}, \mathbb{F} = \{\mathtt{add}, \mathtt{mult}\}, \prec_{\mathbb{F}}, \nu \rangle$

where

$$\begin{aligned}
\mathtt{add}(0;y) &\to y\\
\mathtt{add}(s(x);s(y)) &\to s(s(\mathtt{add}(x;y)))\\
\mathtt{mult}(0,y;) &\to 0\\
\mathtt{mult}(s(x),y;) &\to \mathtt{add}(y;\mathtt{mult}(x,y;))
\end{aligned}$$

Put $\mathtt{add} \prec_{\mathbb{F}} \mathtt{mult}$. We see that $\mathtt{add}(y;\mathtt{mult}(x,y;)) \prec_0 \mathtt{mult}(s(x),y;)$ because $y \prec_1 \mathtt{mult}(s(x),y;)$ and $\mathtt{mult}(x,y;) \prec_0 \mathtt{mult}(s(x),y;)$. It is easy to see that `add` and `mult` denote the addition and multiplication over numerals written in unary. Now, any polynomial is representable. For example, $x \mapsto x^3$ is obtained by $\mathtt{cube}(x;) \to \mathtt{mult}(\mathtt{mult}(x,x;),x;)$. Put $\mathtt{mult} \prec_{\mathbb{F}} \mathtt{cube}$. We have $\mathtt{mult}(\mathtt{mult}(x,x;),x;) \prec_0 \mathtt{cube}(x;)$ because, as we have already seen, $\mathtt{mult}(x,x;) \prec_1 \mathtt{cube}(x;)$.

2. Given two strings $u = u_1 \cdots u_m$ and $v = v_1, \cdots, v_n$ of $\{0,1\}^*$, a common subsequence of length k is defined by two sequences of indices $i_1 < \cdots < i_k$ and $j_1 < \cdots < j_k$ satisfying $u_{i_q} = v_{j_q}$. Consider the well known problem which consists of computing the longest common subsequence of two words. A recursive algorithm has exponential runtime, but the problem is solved in polynomial time by dynamic programming.

   The LMPO-program $\langle \to, \mathbb{W} = \{s, 0, s_0, s_1, \epsilon\}, \mathbb{F} = \{\mathtt{max}, \mathtt{lcs}\}, \mathtt{lcs}, \prec_{\mathbb{F}} \rangle$ computes the longest common subsequence of two words from the recursive definition of the problem. For this, words are represented by constructor terms in $\mathcal{T}(\{s_0, s_1, \epsilon\})$ and integers are represented in unary by terms of $\mathcal{T}(\{s, 0\})$.

$$\begin{aligned}
\mathtt{max}(x,y;n,0) &\to n\\
\mathtt{max}(x,y;0,m) &\to m\\
\mathtt{max}(0,s_i(y);s(n),s(m)) &\to \mathtt{max}(0,y;n,m) \qquad i \in \{0,1\}\\
\mathtt{max}(s_i(x),y;s(n),s(m)) &\to \mathtt{max}(x,y;n,m) \qquad i \in \{0,1\}
\end{aligned}$$

$$\begin{aligned}
\mathtt{lcs}(\epsilon,\epsilon,n;) &\to 0\\
\mathtt{lcs}(s_i(x),s_i(y);) &\to s(\mathtt{lcs}(x,y;))\\
\mathtt{lcs}(s_i(x),s_j(y);) &\to \mathtt{max}(x,y;\mathtt{lcs}(x,s_j(y);),\mathtt{lcs}(s_i(x),y;)) \quad i \neq j
\end{aligned}$$

$\mathtt{max}(u, v; n, m)$ returns the maximum n and m providing that $n, m \leq |u| + |v|$. Notice that the trick of using extra parameters in `max` comes from the banning of the using of critical arguments, like $\mathtt{lcs}(x, s_i(y))$, as recurrence parameters. The program is ordered by $\prec_0$ by putting $\mathtt{max} \prec_{\mathbb{F}} \mathtt{lcs}$. In conclusion, this example shows that the class of $LMPO$ programs delineates a broad class of algorithms.

The computational time is measuring from the size of the input arguments. The size $|t|$ of a term t is the number of symbols in t.

$$|t| = \begin{cases} 1 & \text{if } t \text{ is a constant or a variable} \\ \sum_{i=1}^{n} |t_i| + 1 & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Theorem 3. *Each* LMPO*-program is computable in polynomial time, and inversely each polynomial time function is computed by a* LMPO*-program.*

Proof. It is a consequence of Theorem 4 and 5. □

Theorem 4. *Each function ϕ is computable in polynomial time is represented by a* LMPO*-program.*

Proof. We shall sketch how to represent each function of the class B, as it was introduced in [1], by a LMPO-program. Then, the desired conclusion will follow immediately because the class B is exactly the class of polynomial time computable function.

The set of constructors is $\mathbb{W} = \{s_0, s_1, 0\}$. As already said, the positions of normal inputs will be of valence 1, and the positions of safe inputs will be of valence 0. For example, take the safe recursion schemas:

$$f(0, x; a) \to g(x; a)$$
$$f(s_i(y), x; a) \to h_i(y, x; a, f(y, x; a))$$

where $\nu(f, 1) = \nu(f, 2) = 1$ and $\nu(f, 3) = 0$ and $g, h_0, h_1 \prec_{\mathbb{F}} f$. We can check that $h_i(y, x; a, f(y, x; a)) \prec_0 f(s_i(y), x; a)$, because $y, x \prec_1 f(s_i(y), x; a)$ and $f(y, x; a) \prec_0 f(s_i(y), x; a)$, because $\{y, x\} \prec_1^{\text{multi}} \{s_i(y), x\}$, for each $x, y, a \in \mathcal{T}(\mathbb{W})$.

So, we see that for each B function $\phi : \mathbb{N}^n \times \mathbb{N}^m \mapsto \mathbb{N}$, we could write a program of LMPO such that $\forall u_1, \cdots, u_n, v_1, \cdots, v_m \in \mathcal{T}(\mathbb{W})$, we have

$$\tau(\{f(u_1, \cdots, u_n; v_1, \cdots, v_m)\}) = \phi(\tau(u_1), \cdots, \tau(u_n); \tau(v_1), \cdots, \tau(v_m))$$

where $\tau(0) = 0$ and $\tau(s_i(x)) = 2x + i$. □

Theorem 5. *Let* $\langle \rightarrow, \mathbb{W}, \mathbb{F}, f, \nu, \prec_{\mathbb{F}} \rangle$ *be a* LMPO*-program. For all inputs* $t_1, \cdots, t_n \in \mathcal{T}(\mathbb{W})$*, the computation of* $f(t_1, \cdots, t_n) \stackrel{!}{\Rightarrow} v$*, where* $v \in \mathcal{T}(\mathbb{W})$*, is performed in time bounded by* $p(\max_{i=1,n}(|t_i|))$ *where* p *is some polynomial.*

Proof. We shall design in A.3 an algorithm, by tabulation, which evaluates LMPO programs. Jones, in its book [11], used a similar algorithm to characterise Ptime by mean of a class of read-only imperative programs with recursion is allowed.

The time bound is obtained by two arguments. First, we provide a polynomial upper bound on the size of the normal forms involved in a LMPO-program. This is explained in appendix A.1 and demonstrated in appendix B. This upper-bound is not sufficient to prove the time bound. The difficulty of designing the interpretation may be illustrated by (2) of Example 4. Indeed the size of a term of $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$ can be exponential. As a consequence, there is no interpretation which is polynomially bounded and also provides a termination proof. (That is such that $[u] < [v]$ if $u \prec_0 v$.) Therefore, we must analyse terms involved in derivations and show that the number of recursive calls is bounded by a polynomial in the size of the arguments. This is proved in appendix A.2. □

Let us conclude with a technical remark on program evaluation. We could obtain a similar result on any free data structures, like lists, by representing terms, not as trees, but as directed acyclic graphs (dag). Indeed, what we showed is that if we perform rewriting on dag then both the size of the dag and the length of the derivation are bounded by a polynomial in the size of the inputs.

References

[1] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2:97–110, 1992.

[2] R. Benziger. *Automated complexity analysis of NUPRL extracts.* PhD thesis, Cornell University, 1999.

[3] G. Bonfante, A. Cichon, J.Y Marion, and H. Touzet. Complexity classes and rewrite systems with polynomial interpretation. In *CSL*, number 1584 in Lecture Notes in Computer Science, 1998.

[4] V-H Caseiro. An equational characterization of the poly-time functions on any constructor data strucure. Technical Report 226, University of Oslo, Dept. of informatics, December 1996. `http://www.ifi.uio.no/~ftp/publications`.

[5] R. Constable. Expressing computationnal complexity in constructive type theory. In D. Leivant, editor, *LCC'94*, number 960 in LNCS, pages 131–144, 1995.

[6] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[7] N. Dershowitz. Termination of rewriting. *Journal of symbolic computation*, 1987.

[8] N. Dershowitz and J.P. Jouannaud. *Handbook of Theoretical Computer Science vol.B*, chapter Rewrite systems. Elsevier Science Publishers B. V. (NorthHolland), 1990.

[9] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Comm. ACM*, 22:465–476, 1979.

[10] D. Hofbauer. Termination proofs with multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science*, 105(1):129–140, 1992.

[11] N. Jones. *Computability and complexity, from a programming perspective*. MIT press, 1997.

[12] S. Kamin and J.J. Lévy. Attempts for generalising the recursive path orderings. Technical report, Univerity of Illinois, Urbana, 1980. Unpublished note.

[13] Daniel Leivant. Functions over free algebras definable in the simply typed lambda calculus. *Theoretical Computer Science*, 121:309–321, 1993.

[14] Daniel Leivant. Predicative recurrence and computational complexity I: Word recurrence and poly-time. In Peter Clote and Jeffery Remmel, editors, *Feasible Mathematrics II*. Birkhäuser, 1994.

[15] Jean-Yves Marion. An hierarchy of terminating algorithms with semantic interpretation termination proofs. Rapport de recherche 98-R-273, LORIA, 1998. `http://www.loria.fr/~marionjy`.

[16] Harold Simmons. The realm of primitive recursion. *Archive for Mathematical Logic*, 27:177–, 1988.

A Computing LMPO in PTIME

Define $\mathbb{F}_1, \cdots, \mathbb{F}_k$ as the partition of $\mathbb{F}$ determined by $\preceq_{\mathbb{F}}$ such that if $g \in \mathbb{F}_q$ and $f \in \mathbb{F}_{q+1}$ then $g \prec_{\mathbb{F}} f$, and if f and g are in $\mathbb{F}_q$, then $f \approx g$. We say that if f is in $\mathbb{F}_q$ then f is of rank q. We write $\#S$ to denote the cardinal of the set S.

A.1 A polynomial interpretation

During the reduction of say $f(t_1, \cdots, t_n)$ where all t_i's are in $\mathcal{T}(\mathbb{W})$, the subterms in normal forms which occur in the derivation are like intermediate results. We shall establish that the size of those normal forms which are in $\mathcal{T}(\mathbb{W})$, is bounded by a polynomial in the size of the argument of f. For this, we give a number-theoretic interpretation [ ] of terms and of multisets of terms in $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$.

Fix a constant d and define the sequence of polynomials by

$$F_0(X) = X^d \tag{1}$$

$$F_{k+1}(X) = F_k^d(X) \tag{2}$$

where $d > 3$ is some constant. ($F_k^\alpha(X)$ means α iterations of F_k, i.e. $F_k(\cdots(F_k(X))\cdots)$.)

The interpretation [ ] is defined recursively on $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$ as follows:

- $[a] = d$ for every 0-ary constant a of $\mathbb{W}$.
- $[c(t)] = [t] + d$ for every constructor c of $\mathbb{W}$.
- $[f_k(t_1, \cdots, t_n)] = F_k([t_i \ : \ \nu(f_k, i) = 1]) + \max_{\nu(f_k,i)=0}([t_i])$ for every $f \in \mathbb{F}$ of rank k.
- $[M] = \sum_{t \in \mathsf{lub}(M)} [t]$ for every multiset M, where $\mathsf{lub}(M)$ is the set of maximal terms in M with respect to $\preceq_0$. Formally, $\mathsf{lub}(M)$ is the smallest set including in M satisfying $\forall s \in M$, there is $t \in \mathsf{lub}(M)$ such that $s \preceq_0 t$. In particular, put $[\emptyset] = 3$.

To illustrate this last point, take $f_k(a, a;)$ and $f_k(s(a), b;)$ where a and b are incomparable for $\preceq_0$. Then, we have $f_k(a, a;) \prec_0 f_k(s(a), b;)$ because $\{a, a\} \prec_1 \{s(a), a\}$. Now $\mathsf{lub}(\{a, a\}) = a$ and $\mathsf{lub}(\{s(a), b\}) = \{s(a), b\}$ So $[f_k(a, a;)] = F_k([a]) \leq [f_k(s(a), b;)] = F_k([s(a)] + [b])$.

Lemma 6. *If* $[s] \leq [t]$ *then* $[f(\cdots, s, \cdots)] \leq [f(\cdots, t, \cdots)]$.

Lemma 7. *For each* $u \in \mathcal{T}(\mathbb{W})$, *we have* $|u| \leq [u]$.

Theorem 8. *Let* s *and* t *be two terms such that* $|s| < d$. *For every substitution* σ, *if* $s\sigma \prec_0 t\sigma$ *then* $[s\sigma] \leq [t\sigma]$.

Proof. See the proof B.3 in appendix. □

Theorem 9. *Let* $\langle \rightarrow, \mathbb{W}, \mathbb{F}, f, \nu, \prec_{\mathbb{F}} \rangle$ *be a program of* LMPO. *Assume that* $f(t_1, \cdots, t_n) \stackrel{!}{\Rightarrow} u$, *where* $t_1, \cdots, t_n, u \in \mathcal{T}(\mathbb{W})$. *Then* $|u| \leq p(\max_{i=1,n} |t_i|)$ *for some polynomial which depends on the program.*

Proof. Choose d in the interpretation $[\ ]$ such that for each rule $l \rightarrow r$, we have $d > |r|$. Since $[\ ]$ is increasing by Lemma 6, and by applying Theorem 8, we get $[u] \leq [f(t_1, \cdots, t_n)]$. The definition of $[\ ]$ yields a polynomial p such that $[f(t_1, \cdots, t_n)] \leq p(\max_{i=1,n} [t_i])$. Lastly, by Lemma 7, we obtain $|u| \leq p(\max_{i=1,n} |t_i|)$. □

A.2 Properties of recursive calls

We shall now show that recursive calls are always made on subterms of arguments of the main call. This important property lies on the fact that constructors are treated as the least symbol of signature and that they are incomparable.

Lemma 10. *Let* N *be a constant and let* $\langle \rightarrow, \mathbb{F}, \mathbb{W}, f, \nu, \prec_{\mathbb{F}} \rangle$ *be a* LMPO-*program. Assume that* $f(t_1, \cdots, t_n) \stackrel{*}{\rightarrow} u$ *where each* t_i *is in* $\mathcal{T}(\mathbb{W})$ *and* $|t_i| \leq N$. *Each function symbol* g *of* $\mathbb{F}$ *such that* $g(s_1, \cdots, s_n)$ *is a sub-term of* u, *satisfies*

1. *The rank of* g *is less or equal to the rank of* f,
2. *if* f *and* g *are of the same rank and if* s_i *is a term of* $\mathcal{T}(\mathbb{W})$, *then* s_i *is a subterm of some argument* t_j,
3. *the cardinal of* $\{g(s_1, \cdots, s_n) \prec_0 f(t_1, \cdots, t_n) : g \approx f \text{ and } \forall i, s_i \in \mathcal{T}(\mathbb{W})\}$ *is bounded by* $\#\mathbb{F}_p \cdot n^n \cdot N^n$, *where* f *is of rank* p.

Proof. (1) We have $u \prec_0 f(t_1, \cdots, t_n)$. If the former inequality is justified by say $v \prec_0 t_i$ then v is necessarily in $\mathcal{T}(\mathbb{W})$. So the function symbol f dominates any other function symbol which occurs in u.

Next (2), the definition of $\prec_0$ when $f \approx g$ yields that there is a mapping $\rho : \{1, \cdots, n\} \mapsto \{1, \cdots, n\}$ such that $s_i \preceq_0 t_{\rho(i)}$ or $s_i \preceq_1 t_{\rho(i)}$. Because in both

cases, the inequality is obtained by applying the first rule of the definition of the ordering, we conclude that $s_i \trianglelefteq t_{\rho(i)}$.

(3) Given a mapping ρ as above, there are N^n different n-uplets of arguments which are subterms of $(t_{\rho(1)}, \cdots, t_{\rho(n)})$ and $|t_i| \leq N$. Since there are n^n mapping on $\{1, \cdots, n\}$ and since there are $\#\mathbb{F}_p$ choices for the head function symbols, we conclude that the cardinal these sets is bounded by $\#\mathbb{F}_p \times n^n \times N^n$. □

A.3 Evaluation procedure

Proof of Theorem 5. We describe an evaluation procedure of LMPO programs. which consists of a procedure `eval` and a global store G called the minimal function graph, abbreviated *MFG*.

The notion of *MFG* is taken from [11]. A *MFG* is a set of triples $(f, t_1 :: \cdots :: t_n, k)$ where f is a function symbol in $\mathbb{F}$ of arity n, $t_1 :: \cdots :: t_n$ is the list of the arguments of f, and v is in $\mathcal{T}(\mathbb{W}) \cup \{\bot\}$. The intended meaning is that if $v \in \mathcal{T}(\mathbb{W})$, then $f(t_1, \cdots, t_n)$ has been evaluated to v. Otherwise, if $v = \bot$ then the value of f on $t_1 :: \cdots :: t_n$ is not yet known.

The inputs of `eval` are a term t of $\mathcal{T}(\mathbb{W} \cup \mathbb{F}, \mathcal{V})$ and a substitution σ. `eval`(t, σ) tries to return the value of $t\sigma$ which is obtained by simply looking at the value in G. If it fails then it returns $\bot$ and updates G. G is updated by adding new triples $(g, u_1 :: \cdots :: u_k, \bot)$ which are necessary to carry out the computation of $t\sigma$. The procedure `eval` is detailed in Figure 1.

We are ready to see how the evaluation procedure for LMPO programs works. Initially, G just contains $(f, t_1 :: \cdots :: t_n, \bot)$. It chooses a triple $(h, s_1 :: \cdots :: s_m, \bot)$ in G, and searches for a rewrite rule $l \to r$ which matches $h(s_1, \cdots, s_m)$ with respect to a substitution σ. Then, it runs `eval`(r, σ). Suppose that `eval`(r, σ) returns $v \in \mathcal{T}(\mathbb{W})$.
Then, it replaces $(h, s_1 :: \cdots :: s_m, \bot)$ by $(h, s_1 :: \cdots :: s_m, v)$. This process is repeated until $(f, t_1 :: \cdots :: t_n, v)$ appears in G. The result of the computation of $f(t_1, \cdots, t_n)$ is then v.

We now analyse the time complexity of the procedure described. For this, say that the maximal rank of function symbols is k. We partition first G into $G_0, \cdots, G_k$ where G_p contains all triples in G whose function symbols are of rank p, i.e. in $\mathbb{F}_p$. We have $G_p = \{(f_p, u_1 :: \cdots :: u_n, v) \in G \ : \ f_p \in \mathbb{F}_p\}$. Next, we split G_p into two disjoint sets $G_p^\vee$ and $G_p^\wedge$. $G_p^\vee$ contains all the maximal call $f_p(u_1, \cdots, u_n)$ with respect to $\prec_0$. More precisely, $G_p^\vee$ and $G_p^\wedge$ are the smallest sets satisfying $(f'_p, u'_1 :: \cdots :: u'_n, v') \in G_p^\wedge$ iff there is $(f_p, u_1 :: \cdots :: u_n, v) \in G_p^\vee$ such that $f'(u'_1, \cdots, u'_n) \prec_0 f_p(u_1, \cdots, u_n)$.

(1) and (2) of Lemma 10 yields that a computation of a triple in $G_p^\vee$ was

run by a function symbol of rank $> p$. So, the cardinality of $G_p^\vee$ is bounded by $\sum_{p<i\leq k} \#G_i$.

Theorem 9 claims that the size of normals forms involved in the computation is bounded by $p(\max|t_i|)$ for some polynomial p. Henceforth, the cardinality of $G_p^\wedge$ is bounded by $G_p^\vee \cdot \#\mathbb{F} \times n^n \times p(\max|t_i|)^n$ by Lemma 10(3). Then $\#G_p = \#G_p^\vee + \#G_p^\wedge = (\sum_{p<i\leq k} \#G_i) \times (1 + \#\mathbb{F} \times n^n \times p(\max|t_i|)^n)$. It follows that the cardinal of G is bounded by a polynomial in $p(\max|t_i|)$ whose degree depends on the maximal rank k and on the maximal arity of the signature. We conclude that the evaluation algorithm is running in time bounded by a polynomial in $\max(|t_i|)$. □

B Polynomial bound on the interpretation

B.1 Properties of F_k

We study some properties on (F_k) that we shall use as lemmas, later on.

Proposition 11. *Assume that $d \geq 3$, for all $X \geq d$, and all k,*

1. $F_k(X) = X^{d^{d^k}}$
2. $F_{k+1}^\alpha(X) = F_k^{\alpha \cdot d}(X)$
3. $F_k(X) \geq X$,
4. $F_k^\alpha(X) \leq F_{k+1}(X)$, $\alpha \leq d$
5. $X \leq Y$ *implies* $F_k(X) \leq F_k(Y)$,
6. $F_k(X) \geq d$

Proof. (1) and (2) are proved by showing by induction on (k, α) that we have $F_k^\alpha(X) = X^{d^{\alpha \cdot d^k}}$. □

Proposition 12. *For all k, all $d \geq 3$ and all $X \geq d$, we have*

1. *For all $\alpha, \beta > 0$, $F_k^\alpha(X) + F_k^\beta(X) \leq F_k^{\alpha+\beta}(X)$*
2. *For all $\alpha < d$, $F_k^\alpha(X+1) + F_{k+1}(X) \leq F_{k+1}(X+1)$*
3. *For all $\alpha < d$, $F_k^\alpha(X) + d \leq F_{k+1}(X)$.*

Proof. (1) By induction on k. Case $k = 0$, assume that $\alpha \geq \beta$. We have $X^{d^\alpha} + X^{d^\beta} \leq 2.X^{d^\alpha} \leq X^{d^{(\alpha+1)}}$ because $X \geq d$, and so $\leq X^{d^{(\alpha+\beta)}}$.
Case $k > 0$,

$$\begin{aligned} F^\alpha_{k+1}(X) + F^\beta_{k+1}(X) &= F^{d\alpha}_k(X) + F^{d\beta}_k(X) && \text{by (2)} \\ &\leq F^{d\cdot(\alpha+\beta)}_k(X) && \text{by ind. hyp.} \\ &= F^{\alpha+\beta}_{k+1}(X) && \text{by (2)} \end{aligned}$$

(2) Assume that $\gamma < \beta$, we have $(X+1)^\gamma + X^\beta \leq (X+1)^\beta$, because $(X+1)^\beta = (X+1)(X+1)^{\beta-1} \geq X^\beta + (X+1)^{\beta-1}$. We conclude by setting $\gamma = d^{d^k\alpha} < d^{d^{k+1}} = \beta$.

(3) We have $d \leq 2^d \leq F_k(X)$. Thus, $F^\alpha_k(X) + d \leq F^{\alpha+1}_k(X)$ by (1) of Proposition 12. By (3) of Proposition 11, we conclude that $F^\alpha_k(X) + d \leq F_{k+1}(X)$ □

B.2 Bounds on arguments of valence 1

Lemma 13. *Let s and $t = f(t_1, \cdots, t_n)$ be two terms of $\mathcal{T}(\mathbb{W} \cup \mathbb{F}, \mathcal{V})$ such that $\mathrm{FV}(s) \subseteq \mathrm{FV}(t)$. Let σ be a substitution which assigns each variable of $\mathrm{FV}(t)$ to a ground term of $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$.*

Suppose that $s\sigma \prec_1 t\sigma$. Assume also that for all $i \in \{1, \cdots, n\}$, and all terms u, $u\sigma \preceq_1 t_i\sigma$ implies $[u\sigma] \leq [t_i\sigma]$. Then, $[s\sigma] \leq F^{|s|}_k([t_i\sigma \ : \ \nu(f,i) = 1])$ where f is a function symbol of rank $k+1$.

Proof. We write A for a short hand for $[t_i\sigma \ : \ \nu(f,i) = 1]$. Notice that if $u\sigma \preceq_1 t_i\sigma$ then there is a term t_q of $\mathsf{lub}\{t_i \ : \ \nu(f,i) = 1\}$ such that $u\sigma \preceq_1 t_i\sigma \preceq_1 t_q\sigma$. The Lemma assumption yields that $[u\sigma] \leq [t_q\sigma] \leq A$.
The proof is by inductions on $|s|$.

Assume $|s| = 1$. Suppose that s is a constant of $\mathbb{W}$. We have $[s\sigma] = d \leq F_k(A)$, by (6) of Proposition 11.

Suppose that s is a variable. By assumption, $s \in \mathrm{FV}(t_i)$ for some i satisfying $\nu(f,i) = 1$. By the preliminary observation, $[s\sigma] \leq A$. So, $s\sigma \leq F_k(A)$ by (3) of Proposition 11.

Assume $|s| > 1$. Let s be $f_j(s_1, \cdots, s_m)$ where f_j is a function symbol of $\mathbb{F}$ of rank j. There are two cases to consider.

Suppose that $s\sigma \preceq_1 t_i\sigma$ and that $\nu(f,i) = 1$. The preliminary observation claims that $[s\sigma] \leq A$. By (3) of Proposition 11, we have $[s\sigma] \leq F^{|s|}_k(A)$.

Suppose that and $j \leq k$ and that for all $i \in \{1, \cdots, m\}$, $s_i\sigma \prec_1 t\sigma$. We

have

$$
\begin{aligned}
[s\sigma] &\leq F_j(\sum_{i=1,m}[s_i\sigma]) \\
&\leq F_j(\sum_{i=1,m}F_k^{|s_i|}(A)) \quad \text{by induction hypothesis and by (5) of Prop. 11} \\
&\leq F_k(F_k^{\sum_{i=1,m}|s_i|}(A)) \quad \text{by (1) of Prop. 12} \\
&\leq F_k^{\sum_{i=1,m}|s_i|+1}(A)
\end{aligned}
$$

Lastly, the case when $s = c(s')$ where c is a constructor of $\mathbb{W}$, is proved by (3) of Proposition 12 □

Corollary 14. *Let $s_1, \cdots, s_m$ be incomparable terms of $\mathcal{T}(\mathbb{W} \cup \mathbb{F}, \mathcal{V})$ with respect to $\prec_1$ and such that $\sum_{i=1,m}|s_i| < d$. Let t be terms of $\mathcal{T}(\mathbb{W} \cup \mathbb{F}, \mathcal{V})$, such that $\mathrm{FV}(s_i) \subseteq \mathrm{FV}(t)$. If for each ground substitution σ, and for all i, we have $s_i\sigma \prec_1 t\sigma$, then $\sum_{i=1,m}[s_i] < [t]$.*

Sketch of proof. Assume that $t = f_{k+1}(t_1, \cdots, t_n)$ where f_{k+1} is a function symbol of $\mathbb{F}$ of rank $k+1$. It follows from Lemma 13 that $[s_i\sigma] \leq F_k^{|s_i|}(A)$. Now, we have $\sum_{i=1,m}F_k^{|s_i|}(A) \leq F_k^{\sum_{i=1,m}|s_i|}(A)$ by (1) of Proposition 12. Since $\sum_{i=1,m}|s_i| < d$, by (4) of Proposition 11, we have $F_k^{\sum_{i=1,m}|s_i|}(A) < F_{k+1}(A)$. So we obtain the conclusion $\sum_{i=1,m}[s_i] < F_{k+1}(A)$. □

B.3 An upper bound on the interpretation

Lemma 15. *Let s and $t = f(t_1, \cdots, t_n)$ be two terms of $\mathcal{T}(\mathbb{W} \cup \mathbb{F}, \mathcal{V})$ such that $\mathrm{FV}(s) \subseteq \mathrm{FV}(t)$. Let σ be a substitution which assigns each variable of $\mathrm{FV}(t)$ to a ground term of $\mathcal{T}(\mathbb{W} \cup \mathbb{F})$. Suppose that $s\sigma \prec_0 t\sigma$.*

Assume also, that for all $i \in \{1, \cdots, n\}$, and all terms u, $u\sigma \preceq_0 t_i\sigma$ implies $[u\sigma] \leq [t_i\sigma]$. Then,

$$[s\sigma] \leq F_k^{|s|}(A) + F_{k+1}(A-1) + \max_{\nu(f,i)=0}([t_i\sigma])$$

where $A = [\{t_i\sigma \ : \ \nu(f,i) = 1\}]$ and f is a function symbol of rank $k+1$.

Proof. By induction on $|s|$. We skip base cases which are immediate.

Assume $|s| > 1$. Let s be $f_j(s_1, \cdots, s_m)$ where f_j is a function symbol of $\mathbb{F}$ of rank j. There are three cases to consider.

Suppose that $s \prec_0 t_i$ and $\nu(f,i) = 0$. Lemma assumptions yield $[s] \leq [t_i]$. Thus, $[s] \leq \max_{\nu(f,i)=0}([t_i])$.

Suppose that $j \leq k$. Then, for all $i \in \{1, \cdots, m\}$, if $\nu(f,i) = 1$, we have $s_i \preceq_1 t$. Lemma 13 yields $[s_i] \leq F_k^{|s_i|}(A)$, so we have

$$\begin{aligned} F_j([s_i\sigma \ : \ \nu(f_j,i) = 1]) &\leq F_j(\sum_{\nu(f_j,i)=1} F_k^{|s_i|}(A)) && \text{repl. } s_i \text{ by } F_k^{|s_i|}(A) \\ &\leq F_j(F_k^{\sum_{\nu(f_j,i)=1}|s_i|}(A)) && \text{by (12.1)} \\ &\leq F_k^{1+\sum_{\nu(f_j,i)=1}|s_i|}(A) && \text{by (11.4)} \end{aligned}$$

Otherwise, that is if $\nu(f,i) = 0$, we have $s_i \preceq_0 t$. It follows by induction hypothesis that

$$[s_i\sigma] \max_{\nu(f_j,i)=0}([s_i\sigma]) \leq F_k^{\sum_{\nu(f,i)=1}|s_i|}(A) + F_{k+1}(A-1) + \max_{\nu(f,i)=0}([t_i\sigma])$$

By definition, $[s\sigma] = F_j([s_i\sigma : \nu(f_j,i) = 1]) + \max_{\nu(f_j,i)=0}([s_i\sigma])$. By replacing with the above inequalities, we get the following bound on

$$[s\sigma] \leq F_k^{1+\sum_{i=1,m}|s_i|}(A) + F_{k+1}(A-1) + \max_{\nu(f,i)=0}([t_i])$$

The case when $s = c(s')$ is similar to the previous case.

Suppose that $j = k+1$.
Corollary 14 implies that $[s_i : \nu(f_{k+1},i) = 1] < [t_i : \nu(f_{k+1},i) = 1] = A$.
Since $\{s_i \ : \ \nu(f_{k+1},i) = 0\} \preceq_0^{\text{multi}} \{t_i \ : \ \nu(f_{k+1},i) = 0\}$, the Lemma assumptions yields $\max_{\nu(f_{k+1},i)=0}([s_i\sigma]) \leq \max_{\nu(f_{k+1},i)=0}([t_i\sigma])$. From both inequalities, we conclude that

$$\begin{aligned} [s\sigma] &= F_{k+1}([s_i\sigma : \nu(f_{k+1},i) = 1]) + \max_{\nu(f_{k+1},i)=0}([s_i\sigma]) && \text{by def of } [s\sigma] \\ &\leq F_{k+1}(A-1) + \max_{\nu(f_{k+1},i)=0}([t_i\sigma]) \end{aligned}$$

□

Proof of Theorem 8. The proof goes by induction on $|t|$. Assume that $t = c(t')$, $c \in \mathbb{W}$. We have $s\sigma \preceq_0 t'\sigma$, and so $[s\sigma] \leq [t'\sigma]$ by induction hypothesis.

Assume that $t = f_{k+1}(t_1, \cdots, t_n)$. Lemma 15 yields that $[s\sigma] \leq F_k^{|s|}(A) + F_{k+1}(A-1) + \max_{\nu(f,i)=0}([t_i\sigma])$. By Proposition 12, $F_k^{|s|}(A) + F_{k+1}(A-1) < F_{k+1}(A)$. Therefore $[s\sigma] < F_{k+1}(A) + \max_{\nu(f,i)=0}([t_i]) = [f_{k+1}(t_1, \cdots, t_n)\sigma]$.
□

Algorithm 1 `eval`(t, σ) evaluates $t\sigma$ by looking in the *MFG* G.

Require: Global store : *MFG* G
  Input : $t \in \mathcal{T}(\mathbb{W} \cup \mathbb{F}, \mathcal{V})$
  Input : a substitution σ
Ensure: return value in $\mathcal{T}(\mathbb{W}) \cup \{\bot\}$
  {We write (f, σ, k) for $(f, \sigma(x_1) :: \cdots :: \sigma(x_n), k)$.}
  if $t = f(x_1, \cdots, x_n)$ & $(f, \sigma, k) \in M$ **then**
    if $k \in \mathcal{T}(\mathbb{W})$ **then**
      return k
    else
      let σ' and $l \to r$ such that $f(\sigma) = l\sigma'$
      $v =$ `eval`(r, σ')
      $G = G \setminus \{(f, \sigma, \bot)\} \cup \{(f, \sigma, v)\}$
    end if
  end if
  if $t = c(h_1, \cdots, h_n)$ & $c \in \mathbb{W}$ **then**
    for $i = 1$ to n **do**
      $H_i :=$ `eval`(h_i, σ)
      if $H_i = \bot$ **then**
        return $\bot$
      end if
    end for
    return $c(H_1, \cdots, H_n)$
  end if
  if $t = f(h_1, \cdots, h_n)$ & $f \in \mathbb{W}$ **then**
    for $i = 1$ to n **do**
      $H_i :=$ `eval`(h_i, σ)
      if $H_i = \bot$ **then**
        return $\bot$
      end if
    end for
    if $(f, H_1 :: \cdots :: H_n, v) \in G$ **then**
      return v
    else
      $G := G \cup (f, H_1 :: \cdots :: H_n, \bot)$
      return $\bot$
    end if
  end if