

Higher-Order Linear Ramified Recurrence

Ugo Dal Lago* Simone Martini* Luca Roversi†

Abstract

Higher-Order Linear Ramified Recurrence (**HOLRR**) is a **PTIME** sound and complete typed lambda calculus. Its terms are those of a linear (affine) λ -calculus – every variable occurs at most once – extended with a limited recursive scheme on a word algebra. Completeness for **PTIME** holds by embedding Leivant’s ramified recurrence on words into **HOLRR**. Soundness is established at all types – and not only for first order terms. Type connectives are limited to tensor and linear implication. Moreover, typing rules are given as a simple deductive system. On one side, **HOLRR** allows to “program” higher-order functions, whose **PTIME** soundness is assured by their types. On the other, **HOLRR** looks like a usual recursive language.

1 Introduction

Historically, the model of **PTIME** is **PTM**, the class of *polytime Turing machines*. One goal of giving machine-independent characterizations of **PTIME** is to overcome the drawback of conceiving feasible algorithms by thinking directly in terms of low-level-machine primitives, like those of **PTM**. The research about this subject has brought to a wide variety of interesting results, that can be classified under two parameters: their originating background, and their expressivity – the ability to naturally express higher-order functions.

Concerning the originating background, proposals range from those which are purely recursion-theoretical to the ones which are purely type-theoretical.

For example, Bellantoni and Cook’s *safe recursion on notation* [BC92] is of the former kind. Its recursive scheme forbids iteration on the result of any iteratively defined function. The constraint is expressed directly inside the syntax of the recursive schemes, by identifying two classes of arguments, namely safe and normal arguments. Another example of first-order function algebra capturing **PTIME** is Leivant’s *ramified recurrence on words* [Lei93, Lei95], which relies on the notion of tier to control the use of arguments in recursive schemes.

On the other side, purely type-theoretical systems are logical, deductive systems, usually expressed on a graph language, that of proof-nets. Main examples of this class are *light linear logic* (**LLL**, [Gir98]), *light affine logic* (**LAL**, [Asp98, AR02]), and *soft linear logic* (**SLL**, [Laf02]). Boxes are certain regions inside a proof-net, and a box may contain other boxes, in a stratified fashion. The computational core is given by cut-elimination, whose complexity is controlled by box stratification: the time necessary to normalize a proof-net is a polynomial in the size of the proof, the exponent of the polynomial depending only on the box-nesting depth. This, together with the fact that usual data types can be coded by fixed-depth proofs, implies polytime soundness.

Many interesting systems should be classified in-between these two styles. In these *hybrid* systems, recursion is embedded into typed calculi, and other mechanisms – usually ramification or linearity – are needed to control the computational growth. Typical examples of hybrid systems are those described in [Hof97, Hof00] and in [BNS00]. In these works, the syntax of Gödel’s system **T** is

*Dipartimento di Scienze dell’Informazione, Università di Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy {dallago,martini}@cs.unibo.it

†Dipartimento di Informatica, Università di Torino, Corso Svizzera, Torino, Italy roversidi@di.unito.it

modified to accomodate safe recursion, but a number of additional constraints, a restricted form of linearity *in primis*, are needed to reach polynomial soundness. Generalizations of ramified recursion to higher-order types are presented in [LM00, LM94, Lei99]. In these systems, however, the lack of a linearity constraint does not allow a polytime bound. Indeed, at higher types they show either a polyspace or a Kalmar elementary bound.

Results. We introduce in this paper the system of Higher-Order Linear Ramified Recurrence (**HOLRR**). It is a hybrid system, where the two components – the recursion-theoretic and the type-theoretic ones – are blent in a smooth way.

The type-theoretical core of the system is a linear affine lambda-calculus: any variable can be used at most once. The types are generated by the usual multiplicative connectives (tensor and arrow).

Starting from a finite family of free algebras $\mathcal{A} = \{\mathbb{A}_1, \dots, \mathbb{A}_n\}$, we take as base types an enumerable family $\{B_{\mathbb{A}}^n\}_n$ of *copies* of \mathbb{A} , for each $\mathbb{A} \in \mathcal{A}$. Recursion is not introduced by way of a constant, but becomes a variable binder term constructor, whose syntax is inspired by the boxes of linear lambda calculi. There is no need of other additional term or type constructs.

Completeness for **PTIME** holds by embedding Leivant’s ramified recurrence on words into **HOLRR**.

As for soundness, we prove a result *à la* **LLL**. Under the given strategy, any term of the language normalizes in a polynomially bounded time. To be precise, we will prove that, for any term M and type derivation π for M , M normalizes in time $O(|M|^h)$, where h depends only on the recursion depth of π . It is clear that the bound is a polynomial only once the recursion depth of terms encoding input data has been fixed – a similar situation occurring in **LLL** or **LAL**.

2 Syntax

A *free algebra* \mathbb{A} is a couple $(\mathcal{C}_{\mathbb{A}}, \mathcal{R}_{\mathbb{A}})$ where $\mathcal{C}_{\mathbb{A}} = \{c_1^{\mathbb{A}}, \dots, c_{k(\mathbb{A})}^{\mathbb{A}}\}$ is a finite set of *constructors* and $\mathcal{R}_{\mathbb{A}} : \mathcal{C}_{\mathbb{A}} \rightarrow \mathbb{N}$ maps every constructor to its *arity*. A free algebra $\mathbb{A} = (\{c_1^{\mathbb{A}}, \dots, c_{k(\mathbb{A})}^{\mathbb{A}}\}, \mathcal{R}_{\mathbb{A}})$ is a *word algebra* if

- $\mathcal{R}(c_i^{\mathbb{A}}) = 0$ for one (and only one) $i \in \{1, \dots, k(\mathbb{A})\}$;
- $\mathcal{R}(c_j^{\mathbb{A}}) = 1$ for every $j \neq i$ in $\{1, \dots, k(\mathbb{A})\}$.

If $\mathbb{A} = (\{c_1^{\mathbb{A}}, \dots, c_{k(\mathbb{A})}^{\mathbb{A}}\}, \mathcal{R}_{\mathbb{A}})$ is a word algebra, we will assume $c_{k(\mathbb{A})}^{\mathbb{A}}$ to be the distinguished element of $\mathcal{C}_{\mathbb{A}}$ whose arity is 0 and $c_1, \dots, c_{k(\mathbb{A})-1}$ will denote the elements of $\mathcal{C}_{\mathbb{A}}$ whose arity is 1. \mathcal{A} will be a fixed, finite family $\{\mathbb{A}_1, \dots, \mathbb{A}_n\}$ of free algebras, where constructor sets $\mathcal{C}_{\mathbb{A}_1}, \dots, \mathcal{C}_{\mathbb{A}_n}$ are assumed to be pairwise disjoint.

The language $\mathcal{M}_{\mathcal{A}}$ of **HOLRR** *terms* is defined by the following productions:

$$\begin{aligned} M ::= & x \mid c \mid (M, M) \mid M M \mid \lambda x. M \mid \\ & \mathbf{let} (x, x) \leftarrow M \mathbf{in} M \mid \\ & \langle\langle M, \dots, M \rangle\rangle [x/M, \dots, x/M] M \end{aligned}$$

where c ranges over the constructors for the algebras in \mathcal{A} . An occurrence of a term N inside another term M has *recursion degree* n if it is nested into n terms in the form $\langle\langle M, \dots, M \rangle\rangle$ inside M . When we write a term as \overline{M} , we are implicitly assuming it to be closed (i.e. contains no free variables).

The language $\mathcal{T}_{\mathcal{A}}$ of **HOLRR** *types* is defined by the following productions:

$$A ::= B_{\mathbb{A}}^n \mid A \otimes A \mid A \multimap A$$

where n ranges over \mathbb{N} and \mathbb{A} ranges over \mathcal{A} . Tensor associates to the left, both in types and terms (that is, pairs). Given $A \in L_{\mathbb{A}}$, define the *lifting* $\#(A) \in \mathcal{T}_{\mathcal{A}}$ of A :

$$\begin{aligned} \#(B_{\mathbb{A}}^n) &= B_{\mathbb{A}}^{n+1} \\ \#(A \otimes B) &= \#(A) \otimes \#(B) \\ \#(A \multimap B) &= \#(A) \multimap \#(B). \end{aligned}$$

$$\begin{array}{c}
\frac{}{x : A \vdash x : A} A \quad \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B} W \\
\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x. M : A \multimap B} I_{\multimap} \quad \frac{\Gamma \vdash M : A \multimap B \quad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B} E_{\multimap} \\
\frac{\Gamma \vdash M : A \quad \Delta \vdash N : B}{\Gamma, \Delta \vdash (M, N) : A \otimes B} I_{\otimes} \\
\frac{\Gamma \vdash M : A \otimes B \quad \Delta, x : A, y : B \vdash N : C}{\Gamma, \Delta \vdash \mathbf{let}(x, y) \leftarrow M \mathbf{in} N : C} E_{\otimes} \\
\frac{n \in \mathbb{N} \quad c \in \mathcal{C}_{\mathbb{A}}}{\vdash c : \underbrace{B_{\mathbb{A}}^n \multimap \dots \multimap B_{\mathbb{A}}^n \multimap B_{\mathbb{A}}^n}_{\mathcal{R}_{\mathbb{A}}(c) \text{ times}}} I_c \\
\frac{A \equiv B_{\mathbb{A}}^i \quad \Gamma \equiv x_1 : B_1, \dots, x_n : B_n \quad \Gamma \vdash M_{c_i^{\mathbb{A}}} : \underbrace{A \multimap \dots \multimap A}_{\mathcal{R}_{\mathbb{A}}(c_i^{\mathbb{A}}) \text{ times}} \multimap \underbrace{C \multimap \dots \multimap C}_{\mathcal{R}_{\mathbb{A}}(c_i^{\mathbb{A}}) \text{ times}} \multimap C}{\Delta_i \vdash N_i : B_i \quad \Theta \vdash L : A} \\
\hline
\Delta_1, \dots, \Delta_n, \Theta \vdash \langle\langle M_{c_1} \dots M_{c_k} \rangle\rangle[x_1/N_1, \dots, x_n/N_n] L : C \quad E_{\multimap}^{\mathbb{A}}
\end{array}$$

Figure 1: Type assignment rules

The *level* $\mathbb{L}(A) \in \mathbb{N}$ of a type A is defined by induction on A :

$$\begin{aligned}
\mathbb{L}(B_{\mathbb{A}}^n) &= n \\
\mathbb{L}(A \otimes B) &= \max\{\mathbb{L}(A), \mathbb{L}(B)\} \\
\mathbb{L}(A \multimap B) &= \max\{\mathbb{L}(A), \mathbb{L}(B)\}
\end{aligned}$$

The rules in Figure 1 define the assignment of types in $\mathcal{I}_{\mathcal{A}}$ to terms in $\mathcal{M}_{\mathcal{A}}$; $\mathcal{M}_{\mathcal{A}}^{\mathbf{H}}$ is the set of **HOLRR** typable terms. A type derivation π having conclusion $\Gamma \vdash M : A$ will be denoted by $\pi : \Gamma \vdash M : A$. If there is $\pi : \Gamma \vdash M : A$ then we will write $\Gamma \vdash_{\mathbf{H}} M : A$.

For every term t of a free algebra $\mathbb{A} \in \mathcal{A}$ and for every natural number n , there is an **HOLRR** type derivation $\pi(t, n) : \vdash t : B_{\mathbb{A}}^n$.

The *recursion depth* $\mathbb{R}(\pi)$ of a **HOLRR** type derivation $\pi : \Gamma \vdash M : A$ is defined by induction on the structure of π . In particular:

- If π is an instance of rules A or I_c , then $\mathbb{R}(\pi) = 0$.
- If the last rule used in π is $E_{\multimap}^{\mathbb{A}}$, then π has the following shape

$$\frac{\pi_1 \quad \dots \quad \pi_m \quad \Gamma \vdash M : B_{\mathbb{A}}^n}{\Delta \vdash \langle\langle \dots \rangle\rangle[\dots] M : A}$$

and $\mathbb{R}(\pi)$ is $n + \max\{\mathbb{R}(\pi_1), \dots, \mathbb{R}(\pi_m)\}$.

- In all the other cases, π can be written as follows

$$\frac{\pi_1 \quad \dots \quad \pi_m}{\Gamma \vdash M : A}$$

where $m \in \{1, 2\}$. We will define $\mathbb{R}(\pi)$ as $\max\{\mathbb{R}(\pi_1), \dots, \mathbb{R}(\pi_m)\}$.

Proposition 1 *If $\Gamma \vdash_{\mathbf{H}} M : A$ and $\Delta, x : A \vdash_{\mathbf{H}} N : B$, then $\Gamma, \Delta \vdash_{\mathbf{H}} N\{x/M\} : B$.*

Proof. Induction on the structure of the derivation for $\Delta, x : A \vdash_{\mathbf{H}} N : B$. □

Proposition 2 *If $x_1 : A_1, \dots, x_n : A_n \vdash_{\mathbf{H}} M : B$, then $x_1 : \#(A_1), \dots, x_n : \#(A_n) \vdash_{\mathbf{H}} M : \#(B)$*

$$\begin{array}{c}
(\lambda x.M)N \mapsto M\{N/x\} \\
\mathbf{let}(x, y) \leftarrow (M, N) \mathbf{in} L \mapsto L\{M/x, N/y\} \\
\langle\langle M_{c_1}, \dots, M_{c_k} \rangle\rangle[x_1/\overline{N_1}, \dots, x_n/\overline{N_n}] c_i(t_1, \dots, t_{C(c_i)}) \mapsto \\
M_{c_i}\{x_1/\overline{N_1}, \dots, x_n/\overline{N_n}\} t_1 \cdots t_{C(c_i)} \\
\langle\langle M_{c_1}, \dots, M_{c_k} \rangle\rangle[x_1/\overline{N_1}, \dots, x_n/\overline{N_n}] t_1 \\
\dots \\
\langle\langle M_{c_1}, \dots, M_{c_k} \rangle\rangle[x_1/\overline{N_1}, \dots, x_n/\overline{N_n}] t_C
\end{array}$$

Figure 2: Normalization on terms

The reduction rule \mapsto on $M_{\mathbb{A}}$ is given in Figure 2; \rightsquigarrow is the reflexive, transitive and contextual closure of \mapsto , which enjoys the usual Church-Rosser property. Redexes in the form

$$\langle\langle M_{c_1}, \dots, M_{c_k} \rangle\rangle[x_1/\overline{N_1}, \dots, x_n/\overline{N_n}] t$$

are called *recursive redex*; all the other are called *linear redexes*. Strong normalization can be proven by embedding the calculus into System **T**.

The class $\mathcal{M}_{\mathcal{A}}^{\mathbf{H}}$ contains terms that cannot be reduced in polynomial time. To enforce this property, we introduce the notion of a *word-ramified type derivation*, which is a type derivation π such that all instances of rule $E_{\rightarrow}^{\mathbb{A}}$ contained in π have the following two properties:

- $\mathbb{L}(A) > \mathbb{L}(C)$;
- For all $i \in \{1, \dots, n\}$, B_i is in the form $B_{\mathbb{A}}^m$, where \mathbb{A} is a *word algebra*.

In the following section, we will show that these conditions are both crucial to reach polytime soundness. If there is a word-ramified type derivation $\pi : \Gamma \vdash M : A$, then M is said to be *word-ramified* and we will write $\pi : \Gamma \vdash_{\mathbf{WRH}} M : A$. The class of word-ramified typeable **HOLRR** terms will be denoted as $\mathcal{M}_{\mathcal{A}}^{\mathbf{WRH}}$.

3 Comparison with Previous Work

There are a number of hybrid systems with the same goal as **HOLRR** [BNS00, BS01, Hof97, Hof99a, Hof00]. The most similar is certainly **LT**, introduced in [BNS00] and refined in [BS01]. In this section, we will try to underlying major differences between **HOLRR** and **LT**.

First of all, these two systems have been designed from different starting points. **LT** is basically a *restriction* of Gödel system **T**, extending the ideas of safe recursion [BC92] to higher-order. **HOLRR**, on the other hand, has been obtained by *endowing* linear affine lambda calculus with constants and recursion, following Leivant's ramified recurrence.

In the presence of higher-order recursion, linearity is a key ingredient to control the complexity of normalization. **LT** adopts a liberal notion of linearity, where free variables of ground type can appear more than once. In **HOLRR**, any free variable occurs at most once in every typable term and recursion is designed to keep this constraint satisfied.

Ramification and safety are other powerful tools to keep exponential growth out of scope. In **LT**, safety is captured by splitting arrows, products and variables into two families (the complete and the incomplete ones) and by imposing a number of constraints on their interplay. In **HOLRR**, ramification has the same flavour as in the original work on ramified recursion [Lei93, Lei95], without any major change.

HOLRR accomodates generic free algebras in a uniform way, with just one recursion scheme. In **LT**, generic tree algebras are not taken into account, and their introduction would force limiting the nonlinear behaviour of ground variables [Hof00].

In both cases, the system obtained is polytime complete. However, polytime soundness is formulated and proved in two different ways. In **LT**, any term M with free variables x_1, \dots, x_n is equipped with a polynomial $P_M(y_1, \dots, y_n)$ in such a way that the time to normalize $M\{y_1/N_1, \dots, y_n/N_n\}$ is $O(P_M(|N_1|, \dots, |N_n|))$; this result, however, relies on a number of assumptions: all the terms

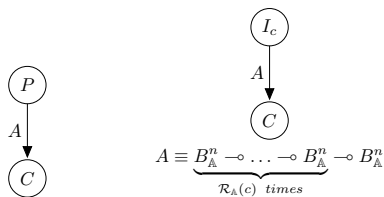


Figure 3: Base case

involved must have linear type, N_1, \dots, N_n must all be closed and cannot contain complete free variables of higher type, all free variables of $M\{y_1/N_1, \dots, y_n/N_n\}$ have to be linear and incomplete. It is not clear which attributes of M influence the degree of P_M ; in other words, it is not possible to *isolate* anything inside M , proving the degree of P_M to depend *only* on that. In **HOLRR**, on the contrary, the time needed to compute the normal form of every word-ramified term M is $O(|M|^h)$, where h only depends on the recursion depth of a type derivation for M ; one can then observe that the recursion depth of any type derivation for any term of any free algebra is null. This enable us to claim that our soundness theorem is deeper and more general (it applies to every word-ramified term) than the one given on **LT**.

4 Polytime Soundness

In this section, we will prove polytime soundness for **HOLRR** through the following result:

Theorem 1 *There is a sound and complete normalization strategy for terms in $\mathcal{M}_{\mathcal{A}}^{\text{WRH}}$ such that the time required to normalize a term M , where $\pi : \Gamma \vdash M : A$, is $O(|M|^h)$ where h depends only on $\mathbb{R}(\pi)$.*

The reduction strategy we are proposing, when applied to M , proceeds by firing those redexes that (seen as subterms of M) have minimum recursion degree among those present in M . A recursive redex

$$\langle\langle M_{c_1}, \dots, M_{c_k} \rangle\rangle [x_1/\overline{N_1}, \dots, x_n/\overline{N_n}] t$$

can be fired only if $\overline{N_1}, \dots, \overline{N_n}$ are in normal form; if a recursive redex is fired, then it is completely unfolded. This strategy is called the *minimum recursion degree strategy*.

We shall prove Theorem 1 studying normalization by way of interaction graphs, which precisely correspond to **HOLLR** type derivations.

Let \mathcal{L} be the set

$$\{W, I_{\rightarrow}, E_{\rightarrow}, I_{\otimes}, E_{\otimes}, P, C\} \cup \bigcup_{A \in \mathcal{A}} \bigcup_{c \in C_A} \{I_c\} \cup \bigcup_{A \in \mathcal{A}} \{E_{\otimes}^A, P^A\}.$$

Notice that elements of \mathcal{L} either are typing rule names or lie in $\{P, C\} \cup \bigcup_{A \in \mathcal{A}} \{P^A\}$

An *interaction graph* is a quadruple (V, E, α, β) such that

- (V, E) is a directed graph;
- $\alpha : V \rightarrow \mathcal{L}$
- $\beta : E \rightarrow \mathcal{T}_{\mathcal{A}}$

A vertex $v \in V$ of an interaction graph $G = (V, E, \alpha, \beta)$ is a *recursion vertex* if $\alpha(v) \in \bigcup_{A \in \mathcal{A}} \{E_{\otimes}^A, P^A\}$.

$\mathcal{G}_{\mathcal{A}}$ is the set of all interaction graphs. We will now introduce a class $\mathcal{G}_{\mathcal{A}}^{\text{H}}$ of interaction graphs corresponding to **HOLRR** type derivations. $\mathcal{G}_{\mathcal{A}}^{\text{H}}$ is defined inductively, mimicking the process of type derivation building. First of all, interaction graphs having the shape of those depicted in figure 3 lies in $\mathcal{G}_{\mathcal{A}}^{\text{H}}$. Moreover, suppose $G_1, \dots, G_{k(\mathbb{A})+n+1} \in \mathcal{G}_{\mathcal{A}}^{\text{H}}$ and every G_i has the shape depicted in figure 4; then all the interaction graphs depicted in figure 5 lie in $\mathcal{G}_{\mathcal{A}}^{\text{H}}$, provided the constraints listed next to each graph are satisfied.

Interaction graphs belonging to $\mathcal{G}_{\mathcal{A}}^{\text{H}}$ enjoy a number of properties:

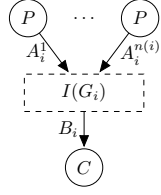


Figure 4: The shape of G_i

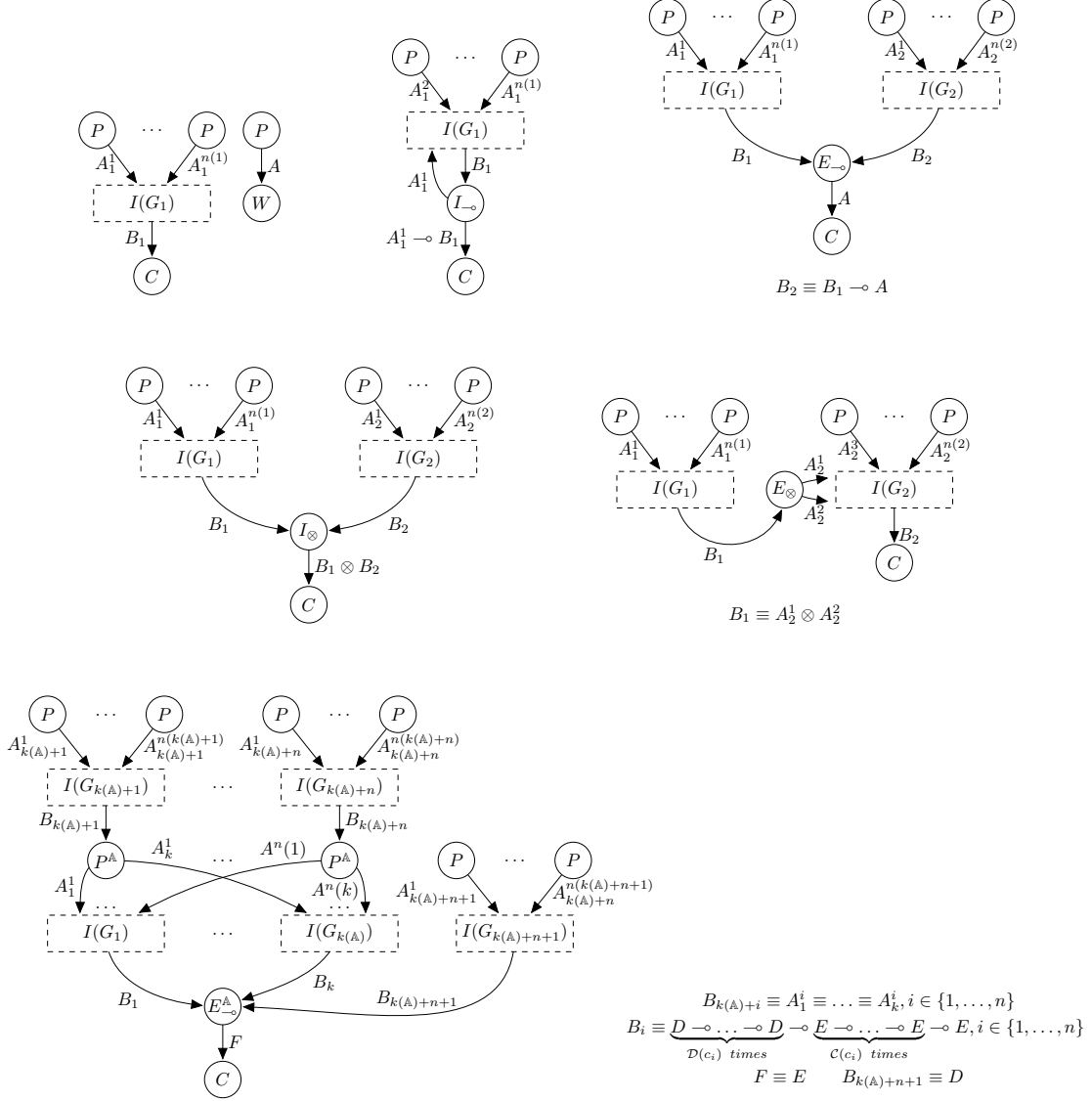


Figure 5: Inductive case

- The indegree $\text{indeg}(v)$ and the outdegree $\text{outdeg}(v)$ of each node v of an interaction graph in $\mathcal{G}_{\mathcal{A}}^{\mathbf{H}}$ are completely determined by the label $\alpha(v)$ as in the following table:

$\alpha(v)$	W	I_{\ominus}	E_{\ominus}	I_{\otimes}	E_{\otimes}	I_C	E_{\ominus}^A	P^A	P	C
$\text{indeg}(v)$	1	1	2	2	1	0	$k(\mathbb{A}) + 1$	1	0	1
$\text{outdeg}(v)$	0	2	1	1	2	1	1	$k(\mathbb{A})$	1	0

- Edges of a graph in $\mathcal{G}_{\mathcal{A}}^{\mathbf{H}}$ that are incident to certain vertexes must be labelled according to some rules. In particular, we can suppose, for each vertex v , to sort all incoming and outgoing edges in two lists $in_1^v, \dots, in_{indeg(v)}^v$ and $out_1^v, \dots, out_{outdeg(v)}^v$ in such a way that the following constraints are satisfied (for every v):

$\alpha(v)$	Constraint
I_{\rightarrow}	$\beta(out_1^v) = A \wedge \beta(in_1^v) = B \wedge \beta(out_2^v) = A \multimap B$
E_{\rightarrow}	$\beta(out_1^v) = B \wedge \beta(in_1^v) = A \wedge \beta(in_2^v) = A \multimap B$
I_{\otimes}	$\beta(in_1^v) = A \wedge \beta(in_2^v) = B \wedge \beta(out_1^v) = A \otimes B$
E_{\otimes}	$\beta(out_1^v) = A \wedge \beta(out_2^v) = B \wedge \beta(in_1^v) = A \otimes B$
I_c	$\beta(out_1^v) = \underbrace{B_{\mathbb{A}}^n \otimes \dots \otimes B_{\mathbb{A}}^n}_{\mathcal{R}_{\mathbb{A}}(c) \text{ times}} \multimap B_{\mathbb{A}}^n$
$E_{\rightarrow}^{\mathbb{A}}$	$\beta(out_1^v) = A \wedge \beta(in_{k+1}^v) = B \wedge \bigwedge_{i=1}^k \beta(in_i^v) = \underbrace{B \multimap \dots \multimap B}_{\mathcal{R}_{\mathbb{A}}(c_i^+) \text{ times}} \multimap \underbrace{A \multimap \dots \multimap A}_{\mathcal{R}_{\mathbb{A}}(c_i^+) \text{ times}} \multimap A$
$P^{\mathbb{A}}$	$\beta(in_1^v) = A \wedge \bigwedge_{i=1}^k \beta(out_i^v) = A$

- For every $G \in \mathcal{G}_{\mathcal{A}}^{\mathbf{H}}$, there is exactly one vertex v (called *conclusion vertex*) such that $\alpha(v) = C$; the only edge (w, v) is called the *conclusion edge* (*premise vertices* and *premise edges* are defined in the same way).

To every **HOLRR** type derivation $\pi : \Gamma \vdash M : A$ corresponds an interaction graph $\mathcal{G}(\pi) \in G_{\mathbb{A}}^{\mathbf{H}}$. Moreover, every instance of rules $I_{\rightarrow}, E_{\rightarrow}, I_{\otimes}, E_{\otimes}, I_c, E_{\rightarrow}^{\mathbb{A}}$ in π corresponds to a node in $\mathcal{G}(\pi)$ with the same label.

As an example, consider the **HOLRR** term

$$M \equiv \lambda w. \langle \langle \lambda x. \lambda z. c_1^{\mathbb{B}} z, \lambda x. \lambda z. c_2^{\mathbb{B}} z, y \rangle \rangle [y / c_1^{\mathbb{B}} c_3^{\mathbb{B}}] c_2^{\mathbb{B}} c_3^{\mathbb{B}}$$

A word-ramified type derivation $\pi : \vdash M : B_{\mathbb{B}}^1 \multimap B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0$ is the following

$$\frac{\rho_1 \quad \rho_2 \quad \overline{y : B_{\mathbb{B}}^0 \vdash y : B_{\mathbb{B}}^0} \quad \overline{\pi(c_1^{\mathbb{B}} c_3^{\mathbb{B}}, 0)} \quad \overline{\pi(c_2^{\mathbb{B}} c_3^{\mathbb{B}}, 1)}}{\vdash \langle \langle \lambda x. \lambda z. c_1 z, \lambda x. \lambda z. c_2 z, y \rangle \rangle [y / c_1^{\mathbb{B}} c_3^{\mathbb{B}}] c_2^{\mathbb{B}} c_3^{\mathbb{B}} : B_{\mathbb{B}}^0}$$

where ρ_i (with $i \in \{1, 2\}$) is the following:

$$\frac{\frac{\frac{\frac{\frac{\overline{z : B_{\mathbb{B}}^0 \vdash z : B_{\mathbb{B}}^0} \quad \overline{\vdash c_1 : B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0}}{z : B_{\mathbb{B}}^0 \vdash c_1 z : B_{\mathbb{B}}^0}}{\vdash \lambda z. c_1^{\mathbb{B}} z : B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0}}{x : B_{\mathbb{B}}^1 \vdash \lambda z. c_1^{\mathbb{B}} z : B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0}}{\vdash \lambda x. \lambda z. c_1^{\mathbb{B}} z : B_{\mathbb{B}}^1 \multimap B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0}}{y : B_{\mathbb{B}}^0 \vdash \lambda x. \lambda z. c_1^{\mathbb{B}} z : B_{\mathbb{B}}^1 \multimap B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0}}$$

The interaction graph $\mathcal{G}(\pi)$ is represented in figure 4. Subgraphs $G(c_1^{\mathbb{B}} c_3^{\mathbb{B}}, 0)$ and $G(c_2^{\mathbb{B}} c_3^{\mathbb{B}}, 1)$ correspond, respectively, to $\mathcal{G}(\pi(c_1^{\mathbb{B}} c_3^{\mathbb{B}}, 0))$ and $\mathcal{G}(\pi(c_2^{\mathbb{B}} c_3^{\mathbb{B}}, 1))$ (where conclusion vertices and edges omitted).

Lemma 1 *There are two constants $n, m \in \mathbb{Q}$ such that, for every $\pi : \Gamma \vdash M : A$ in standard form, we have $n|M| \leq (|V| + |E|) \leq m|M|$, where $\mathcal{G}(\pi) = (V, E, \alpha, \beta)$.*

The sets of positive and negative contexts are defined by induction as follows:

- $[\cdot]$ is a positive context;
- If $C[\cdot]$ is a positive context and $A \in L_{\mathbb{A}}$, then $C[\cdot] \otimes A$, $A \otimes C[\cdot]$ and $A \multimap C[\cdot]$ are positive contexts, while $C[\cdot] \multimap A$ is a negative context;
- If $C[\cdot]$ is a negative context and $A \in L_{\mathbb{A}}$, then $C[\cdot] \otimes A$, $A \otimes C[\cdot]$ and $A \multimap C[\cdot]$ are negative contexts, while $C[\cdot] \multimap A$ is a positive context;

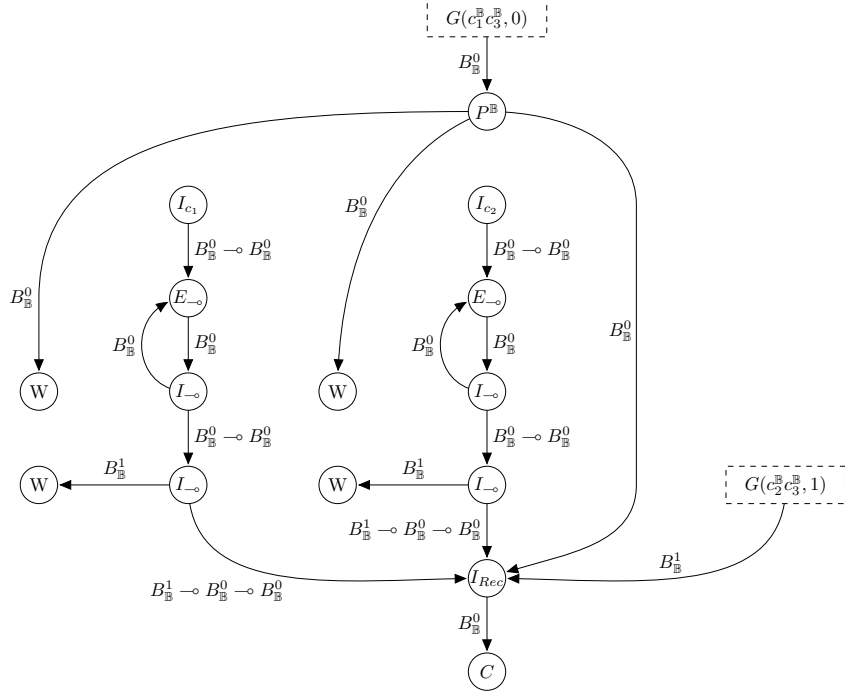


Figure 6: The interaction graph $\mathcal{G}(\pi)$.

$\alpha(v)$	$V_{C[\cdot],e}^G$
I_{\to}	if $e = in_1^u \wedge \beta(out_2^u) = A$ then $\{u\} \cup V_{A \to C[\cdot],out_2^u}^G$
E_{\to}	if $e = out_2^u \wedge C[\cdot] = D[\cdot] \multimap A$ then $\{u\} \cup V_{D[\cdot],out_2^u}^G$
I_{\otimes}	if $e = out_2^u \wedge C[\cdot] = A \multimap D[\cdot]$ then $\{u\} \cup V_{D[\cdot],in_1^u}^G$
E_{\otimes}	$\{u\} \cup V_{A \to C[\cdot],in_2^u}^G$, where $\beta(in_1^u) = A$
E_{\otimes}	if $C[\cdot] = D[\cdot] \otimes A$ then $\{u\} \cup V_{D[\cdot],in_1^u}^G$
E_{\otimes}	if $C[\cdot] = A \otimes D[\cdot]$ then $\{u\} \cup V_{D[\cdot],in_2^u}^G$
E_{\otimes}	if $e = out_1^u \wedge \beta(out_2^u) = A$ then $\{u\} \cup V_{C[\cdot] \otimes A, in_1^u}^G$
E_{\otimes}	if $e = out_2^u \wedge \beta(out_1^u) = A$ then $\{u\} \cup V_{A \otimes C[\cdot], in_1^u}^G$
E_{\to}^A, P^A, C	$\{u\}$
I_c	$\{u\} \cup \bigcup_{i=0}^{\mathcal{R}(c)} V_{C_i[\cdot]}^G$ where $C_i[\cdot] \equiv \underbrace{(B_{\mathbb{A}}^m \multimap \dots \multimap (B_{\mathbb{A}}^m) \multimap [\cdot] \multimap (B_{\mathbb{A}}^m) \multimap \dots \multimap (B_{\mathbb{A}}^m))}_{i \text{ times}} \multimap \underbrace{[\cdot] \multimap (B_{\mathbb{A}}^m) \multimap \dots \multimap (B_{\mathbb{A}}^m)}_{\mathcal{R}_{\mathbb{A}}(c) - i \text{ times}}$

Table 1: Rules to compute $V_{C[\cdot],e}^G$ when $C[\cdot]$ is positive

If $C[\cdot]$ is a context, then $C[B_{\mathbb{A}}^n] \in L_{\mathbb{A}}$ is the type expression obtained by replacing $[\cdot]$ by $B_{\mathbb{A}}^n$ in $C[\cdot]$. Let $G = (V, E, \alpha, \beta) \in G_{\mathbb{A}}^{\mathbf{H}}$, $e = (u, w) \in E$ and $C[\cdot]$ such that $\beta(e) = C[B_{\mathbb{A}}^m]$. Then, the set $V_{C[\cdot],e}^G \subseteq V$ is defined according to tables 1 and 2. Studying the cardinality of the sets $V_{C[\cdot],e}^G$ is very useful in order to understand the dynamics of **HOLRR** minimum recursive degree strategy. Consider for example an occurrence of the recursive redex $N \equiv \langle\langle M_{c_1}, \dots, M_{c_k} \rangle\rangle[x_1/s_1, \dots, x_m/s_m] t$ inside term M , where $\pi_M : \Gamma \vdash M : A$ is word-ramified, and suppose we want to bound the time needed to completely unfold the given redex. In $\mathcal{G}(\pi_M)$, there must be nodes v, v_1, \dots, v_m , corresponding to the occurrence of N under consideration, with $\alpha(v) = E_{\to}^A$ and $\alpha(v_i) = P^A$ for $i \in$

$\alpha(w)$	$V_{C[\cdot],e}^G$
I_{\neg}	$\{w\} \cup V_{A \neg C[\cdot], out_2^w}^G$, where $\beta(out_1^w) = A$
E_{\neg}	if $e = in_1^w \wedge \beta(out_1^w) = A$ then $\{w\} \cup V_{C[\cdot] \neg A, in_2^w}^G$ if $e = in_2^w \wedge C[\cdot] = D[\cdot] \neg A$ then $\{w\} \cup V_{D[\cdot], in_1^w}^G$ if $e = in_2^w \wedge C[\cdot] = A \neg D[\cdot]$ then $\{w\} \cup V_{D[\cdot], out_1^w}^G$
I_{\otimes}	if $e = in_1^w \wedge \beta(in_2^w) = A$ then $\{w\} \cup V_{C[\cdot] \otimes A, out_1^w}^G$ if $e = in_2^w \wedge \beta(in_1^w) = A$ then $\{w\} \cup V_{A \otimes C[\cdot], out_1^w}^G$
E_{\otimes}	if $C[\cdot] = D[\cdot] \otimes A$ then $\{w\} \cup V_{D[\cdot], out_1^w}^G$ if $C[\cdot] = A \otimes D[\cdot]$ then $\{w\} \cup V_{D[\cdot], out_2^w}^G$
E_{\neg}^A, P^A, P, W	$\{w\}$

Table 2: Rules to compute $V_{C[\cdot],e}^G$ when $C[\cdot]$ is negative

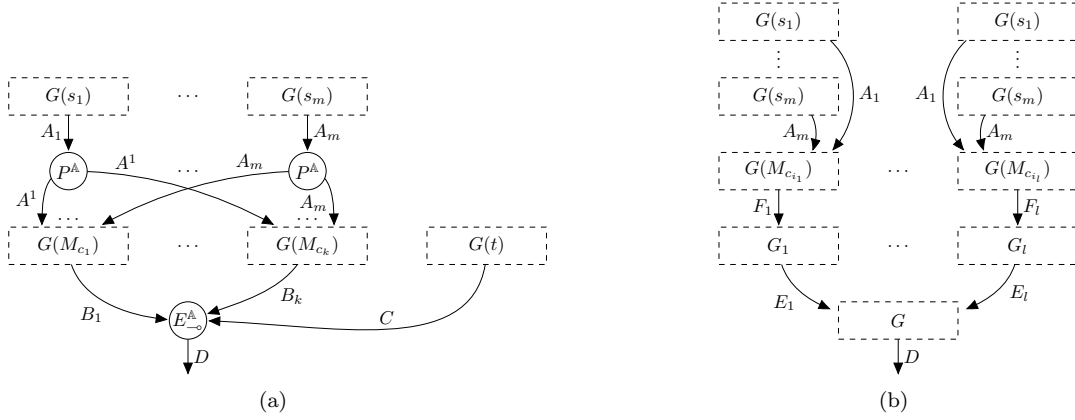


Figure 7: The graph rewriting produced by the complete unfolding of a recursive redex.

$\{1, \dots, m\}$. The size of t, s_1, \dots, s_m can be easily seen to be bounded by $|V_{[\cdot], in_{k(\mathbb{A})+1}^v}^{\mathcal{G}(\pi_M)}|, |V_{[\cdot], in_1^v}^{\mathcal{G}(\pi_M)}|, \dots, |V_{[\cdot], in_1^v}^{\mathcal{G}(\pi_M)}|$.

The unfolding of the redex under consideration produces a term L such that $\pi_L : \Gamma \vdash L : A$. $\mathcal{G}(\pi_L)$ differs from $\mathcal{G}(\pi_M)$ in that a subgraph as the one in figure 7(a) is replaced by the one in figure 7(b), where

- l is bounded by $|t|$;
- the number of nodes in each of G, G_1, \dots, G_l is bounded by $|t|$;

Notice that, due to the fact that π_M is word-ramified, $A_i \equiv B_{\mathbb{A}(i)}^{m(i)}$, where all the $\mathbb{A}(i)$ are word algebras; moreover, the following chain of equalities and inequalities holds:

$$\mathbb{L}(D) = \mathbb{L}(E_1) = \dots = \mathbb{L}(E_l) \leq \mathbb{L}(F_1) = \dots = \mathbb{L}(F_l) = \mathbb{L}(C).$$

Proposition 3 *Let $\pi_M : \Gamma \vdash M : A$. Suppose we apply the minimum recursion depth strategy to M obtaining N , where $\pi_N : \Gamma \vdash N : A$. Then, for every possible edge e of $\mathcal{G}(\pi_N)$ and every possible context $C[\cdot]$,*

$$|V_{C[\cdot],e}^{\mathcal{G}(\pi_N)}| \leq |\mathcal{G}(\pi_M)|^{(\mathbb{R}(\pi_M)+1)^{\mathbb{R}(\pi_M)+1}}.$$

Proof. Let $C[B_{\mathbb{A}}^m] = \beta(e)$. If $m > \mathbb{R}(\pi_M)$ then

$$|V_{C[\cdot],e}^{\mathcal{G}(\pi_N)}| \leq |\mathcal{G}(\pi_M)| \leq |\mathcal{G}(\pi_M)|^{(\mathbb{R}(\pi_M)+1)^{\mathbb{R}(\pi_M)+1}}$$

If $m \leq \mathbb{R}(\pi_M)$, we can show by induction on $\mathbb{R}(\pi_M) - m \geq 0$ that

$$|V_{C[\cdot],e}^{\mathcal{G}(\pi_N)}| \leq (|\mathcal{G}(\pi_M)|)^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}(\mathbb{R}(\pi_M)-m+1)}.$$

The case $m = \mathbb{R}(\pi_M)$ is trivial:

$$|V_{C[\cdot],e}^{\mathcal{G}(\pi_N)}| \leq |\mathcal{G}(\pi_M)| = |\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{0 \cdot 1}} |\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}(\mathbb{R}(\pi_M)-m+1)}.$$

If $m < \mathbb{R}(\pi_M)$, then the worst case is the one in which M is made up of $\mathbb{R}(\pi_M)$ nested **Rec** terms. In this case,

$$\begin{aligned} |V_{C[\cdot],e}^{\mathcal{G}(\pi_N)}| &\leq |\mathcal{G}(\pi_M)| \left(|\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m-1}(\mathbb{R}(\pi_M)-m)} \right)^{\mathbb{R}(\pi_M)} \\ &= |\mathcal{G}(\pi_M)| \left(|\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m-1}(\mathbb{R}(\pi_M)-m)\mathbb{R}(\pi_M)} \right) \\ &= |\mathcal{G}(\pi_M)| \left(|\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}(\mathbb{R}(\pi_M)-m)} \right) \\ &= |\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}(\mathbb{R}(\pi_M)-m)+1} \\ &\leq |\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}(\mathbb{R}(\pi_M)-m)+\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}} \\ &= |\mathcal{G}(\pi_M)|^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}(\mathbb{R}(\pi_M)-m+1)}. \end{aligned}$$

Now, for every $m \in \{0, \mathbb{R}(\pi_M)\}$,

$$\begin{aligned} |V_{C[\cdot],e}^{\mathcal{G}(\pi_N)}| &\leq (|\mathcal{G}(\pi_M)|)^{\mathbb{R}(\pi_M)^{\mathbb{R}(\pi_M)-m}(\mathbb{R}(\pi_M)-m+1)} \\ &\leq (|\mathcal{G}(\pi_M)|)^{(\mathbb{R}(\pi_M)+1)\mathbb{R}(\pi_M)} \\ &= (|\mathcal{G}(\pi_M)|)^{(\mathbb{R}(\pi_M)+1)^{\mathbb{R}(\pi_M)+1}} \end{aligned}$$

that is the thesis. \square

Proposition 4 (Recursive normalization steps) *Let $M \in \mathcal{M}_{sd}^{\mathbf{WRH}}$. Suppose we apply the minimum recursion depth strategy to M obtaining N . Then, the number of recursive redexes fired during normalization is at most*

$$(|\mathcal{G}(\pi_M)|)^{(\mathbb{R}(\pi_M)+1)^{\mathbb{R}(\pi_M)+2}+1}.$$

Proof. First, we will prove that a recursive redex occurrence inside M with recursion degree m can be copied at most

$$(|\mathcal{G}(\pi_M)|)^{(m(\mathbb{R}(\pi_M)+1))^{(\mathbb{R}(\pi_M)+1)}}$$

times. We can proceed by induction on m . If $m = 0$, then the redex occurrence cannot be copied. If $m \geq 1$, then in the worst case, the redex occurrence is copied

$$\begin{aligned} &(|\mathcal{G}(\pi_M)|)^{(m-1)(\mathbb{R}(\pi_M)+1)^{(\mathbb{R}(\pi_M)+1)}} (|\mathcal{G}(\pi_M)|)^{(\mathbb{R}(\pi_M)+1)^{(\mathbb{R}(\pi_M)+1)}} \\ &= (|\mathcal{G}(\pi_M)|)^{(m-1)(\mathbb{R}(\pi_M)+1)^{(\mathbb{R}(\pi_M)+1)} + (\mathbb{R}(\pi_M)+1)^{(\mathbb{R}(\pi_M)+1)}} \\ &= (|\mathcal{G}(\pi_M)|)^{m(\mathbb{R}(\pi_M)+1)^{(\mathbb{R}(\pi_M)+1)}}. \end{aligned}$$

Now, notice that $m \leq \mathbb{R}(\pi_M)$, from which the desired bound can be easily proved. \square

Proof of Theorem 1. From proposition 4 we know that the number of recursive redexes fired during normalization is at most

$$(|\mathcal{G}(\pi_M)|)^{(\mathbb{R}(\pi_M)+1)^{\mathbb{R}(\pi_M)+2}+1},$$

which is $O(|M|^{f(\mathbb{R}(\pi_M))})$. From proposition 3 we can infer that the time needed to completely unfold a recursive redex is itself $O(|M|^{g(\mathbb{R}(\pi_M))})$. The thesis easily follows. \square

5 Polytime Completeness

This property holds by embedding ramified recurrence on words [Lei93] into **HOLRR**.

Ramified recurrence (or *predicative recursion*) on words is a function algebra generating all the polynomial functions in the form $f : \mathbb{A}^n \rightarrow \mathbb{A}$, being \mathbb{A} a word algebra. Here we propose a formulation of *predicative sorting*, based on predicative recurrence:

1. The function $f_{c_{k(\mathbb{A})}^{\mathbb{A}}} : \mathbb{A}^0 \rightarrow \mathbb{A}$ that returns $c_{k(\mathbb{A})}^{\mathbb{A}}$ can be predicatively sorted by $\varepsilon \rightarrow n$ for every $n \in \mathbb{N}$;
2. For every $i \in \{1, \dots, k(\mathbb{A}) - 1\}$, the function $f_{c_i^{\mathbb{A}}} : \mathbb{A} \rightarrow \mathbb{A}$ defined by $f_{c_i^{\mathbb{A}}}(t) = c_i^{\mathbb{A}} t$ can be predicatively sorted by $(n) \rightarrow n$ for every $n \in \mathbb{N}$;
3. For every $n \in \mathbb{N}$ and $1 \leq i \leq n$, the projection $\pi_i^n : \mathbb{A}^n \rightarrow \mathbb{A}$ can be predicatively sorted by $(m_1, \dots, m_n) \rightarrow m$ for every $m, m_1, \dots, m_n \in \mathbb{N}$, with $m_i = m$;
4. If $f : \mathbb{A}^n \rightarrow \mathbb{A}$ can be predicatively sorted by $(m_1, \dots, m_n) \rightarrow m$ and $g_1, \dots, g_p : \mathbb{A}^p \rightarrow \mathbb{A}$ are such that g_i can be predicatively sorted by $(r_1, \dots, r_p) \rightarrow m_i$, then the function $h : \mathbb{A}^p \rightarrow \mathbb{A}$ defined by the equation

$$h(t_1, \dots, t_p) = f(g_1(t_1, \dots, t_p), \dots, g_n(t_1, \dots, t_p))$$

can be predicatively sorted by

$$(r_1, \dots, r_p) \rightarrow m;$$

5. Suppose $g : \mathbb{A}^n \rightarrow \mathbb{A}$ can be predicatively sorted by $(m_1, \dots, m_n) \rightarrow m$ and that, for every $i \in \{1, \dots, k(\mathbb{A}) - 1\}$ there is a function $f_i : \mathbb{A}^{n+2} \rightarrow \mathbb{A}$ that can be predicatively sorted by $(l, m_1, \dots, m_n, m) \rightarrow m$; Then a function $h : \mathbb{A}^{1+n} \rightarrow \mathbb{A}$ can be easily defined recursively:

$$\begin{aligned} h(c_{k(\mathbb{A})}^{\mathbb{A}}, t_1, \dots, t_n) &= g(t_1, \dots, t_n) \\ h(c_i^{\mathbb{A}} t, t_1, \dots, t_n) &= f_i(t, t_1, \dots, t_n, h(t, t_1, \dots, t_n)). \end{aligned}$$

If $l > m$, then h can be predicatively sorted by $(l, m_1, \dots, m_n) \rightarrow m$.

If $f : \mathbb{A}^n \rightarrow \mathbb{A}$ can be predicatively sorted, then it is *defineable by predicative recurrence*.

Theorem 2 (Leivant) *Let \mathbb{A} be a word algebra. Then $f : \mathbb{A}^n \rightarrow \mathbb{A}$ is defineable by predicative recurrence if and only if f is polytime.*

Remark 1 *If f can be predicatively sorted by $(m_1, \dots, m_n) \rightarrow m$ and $m_i < m$, then f does not depend on its i -th argument (see, for example, [Hof99b]). We will suppose that, in rule 3, $m_1, \dots, m_n \geq m$. This will ensure that if f can be predicatively sorted by $(m_1, \dots, m_n) \rightarrow m$, then $m_1, \dots, m_n \geq m$, making our completeness proof simpler. It is easy to realize that this hypothesis does not restrict the space of functions defineable by predicative recurrence.*

Lemma 2 *There are two **HOLRR** terms **Coerc** and **Duplicate**, encoding identity and duplication, such that, for every $n \in \mathbb{N}$ and for every $m, l < n$,*

$$\begin{aligned} \mathbf{Coerc} &: B_{\mathbb{A}}^n \multimap B_{\mathbb{A}}^m \\ \mathbf{Duplicate} &: B_{\mathbb{A}}^n \multimap B_{\mathbb{A}}^m \otimes B_{\mathbb{A}}^l. \end{aligned}$$

Proof. **Coerc** is $\mathbf{Rec}(M_{c_1} \cdots M_{c_k} N)$ where $M_{c_i} \equiv \lambda z. \mathbf{let} (x, y) \leftarrow z \mathbf{in} c_i y$ and $N \equiv e$. **Duplicate** is $\mathbf{Rec}(M_{c_1} \cdots M_{c_k} N)$, where (with a slight abuse of notation), M_{c_i} is $\lambda w. \mathbf{let} (x, y, z) \leftarrow w \mathbf{in} (c_i y, c_i z)$ and N is (e, e) . \square

Lemma 3 *If*

$$M : B_{\mathbb{A}}^{i_1} \otimes \cdots \otimes B_{\mathbb{A}}^{i_n} \otimes B_{\mathbb{A}}^i \otimes B_{\mathbb{A}}^i \multimap B_{\mathbb{A}}^i$$

is typable, then there exists a term

$$N : B_{\mathbb{A}}^{i_1} \otimes \cdots \otimes B_{\mathbb{A}}^{i_n} \otimes B_{\mathbb{A}}^i \multimap B_{\mathbb{A}}^i$$

where N encode the same function as M with the last two arguments contracted.

Proof. From $M : B_{\mathbb{A}}^{i_1} \otimes \dots \otimes B_{\mathbb{A}}^{i_n} \otimes B_{\mathbb{A}}^i \otimes B_{\mathbb{A}}^i \multimap B_{\mathbb{A}}^i$ we can easily obtain, using weakening, a term $L : B_{\mathbb{A}}^{i_1+1} \otimes B_{\mathbb{A}}^{i_1} \otimes \dots \otimes B_{\mathbb{A}}^{i_n} \otimes B_{\mathbb{A}}^i \otimes B_{\mathbb{A}}^i \multimap B_{\mathbb{A}}^i$ which behave exactly as M , discarding the first argument. Let now $P \equiv \mathbf{Rec}(L \cdots L R)$, where $R : B_{\mathbb{A}}^{i_1} \otimes \dots \otimes B_{\mathbb{A}}^{i_n} \otimes B_{\mathbb{A}}^i \multimap B_{\mathbb{A}}^i$ is the projection to the $n+1$ -th argument. Finally, let N be the term

$$\lambda y. \mathbf{let} (x_1, \dots, x_{n+1}) \leftarrow y \mathbf{in} P(c_1 e, x_1, \dots, x_{n+1})$$

Clearly, $N : B_{\mathbb{A}}^{i_1} \otimes \dots \otimes B_{\mathbb{A}}^{i_n} \otimes B_{\mathbb{A}}^i \multimap B_{\mathbb{A}}^i$ is the desired term. \square

Theorem 3 *If $f : \mathbb{A}^n \rightarrow \mathbb{A}$ can be predicatively sorted by $(m_1, \dots, m_n) \rightarrow m$, then there is a term M_f encoding f that can be assigned type*

$$B_{\mathbb{A}}^{l_1} \otimes \dots \otimes B_{\mathbb{A}}^{l_n} \multimap B_{\mathbb{A}}^l$$

where $l, l_1, \dots, l_n \in \mathbb{N}$ and, for every $i \in \{1, \dots, n\}$, either $m_i = m$ and $l_i = l$, or $m_i > m$ and $l_i > l$.

Proof. We induce on the number of rules used to conclude that f can be predicatively sorted; we distinguish a number of cases, depending on the last rule used:

- If the last rule used is rule 1, then $n = 0$ and $M_f = e$, which can be given type $B_{\mathbb{A}}^m$;
- If the last rule used is rule 2, then $f = f_{c_i}$ $n = 1$, $m_1 = m$ and $M_f = c_i$, which can be given type $B_{\mathbb{A}}^m \multimap B_{\mathbb{A}}^m$;
- If the last rule used is rule 3, then $f = \pi_i^n$, and M_f is

$$\lambda y. \mathbf{let} (x_1, \dots, x_{n+1}) \leftarrow y \mathbf{in} x_i,$$

which can be given type $B_{\mathbb{A}}^{m_1} \otimes \dots \otimes B_{\mathbb{A}}^{m_n} \multimap B_{\mathbb{A}}^m$, which satisfies the requirements on m_1, \dots, m_n, m due to remark 1;

- If the last rule used is rule 4, we can use the inductive hypothesis to infer the existence of the following terms:

$$\begin{aligned} M_f & : B_{\mathbb{A}}^{q_1} \otimes \dots \otimes B_{\mathbb{A}}^{q_n} \multimap B_{\mathbb{A}}^q \\ M_{g_1} & : B_{\mathbb{A}}^{s_1^1} \otimes \dots \otimes B_{\mathbb{A}}^{s_p^1} \multimap B_{\mathbb{A}}^{s_1^1} \\ & \vdots \\ M_{g_n} & : B_{\mathbb{A}}^{s_1^n} \otimes \dots \otimes B_{\mathbb{A}}^{s_p^n} \multimap B_{\mathbb{A}}^{s_1^n}. \end{aligned}$$

Let now u, u^1, \dots, u^n be positive integers such that, for every $i \in \{1, \dots, n\}$ $q_i + u = s^i + u^i$. By proposition 2,

$$\begin{aligned} M_f & : B_{\mathbb{A}}^{q_1+u} \otimes \dots \otimes B_{\mathbb{A}}^{q_n+u} \multimap B_{\mathbb{A}}^{q+u} \\ M_{g_1} & : B_{\mathbb{A}}^{s_1^1+u^1} \otimes \dots \otimes B_{\mathbb{A}}^{s_p^1+u^1} \multimap B_{\mathbb{A}}^{q_1+u} \\ & \vdots \\ M_{g_n} & : B_{\mathbb{A}}^{s_1^n+u^n} \otimes \dots \otimes B_{\mathbb{A}}^{s_p^n+u^n} \multimap B_{\mathbb{A}}^{q_n+u}. \end{aligned}$$

All these terms can be composed, obtaining a term

$$N : \bigotimes_{i=1}^n (B_{\mathbb{A}}^{s_1^i+u^i} \otimes \dots \otimes B_{\mathbb{A}}^{s_p^i+u^i}) \multimap B_{\mathbb{A}}^{q+u}$$

from which, using iteratively lemmas 2 and 3 to contract corresponding arguments, we can obtain a term

$$M_h : B_{\mathbb{A}}^{v_1} \otimes \dots \otimes B_{\mathbb{A}}^{v_p} \multimap B_{\mathbb{A}}^{q+u}$$

encoding h and satisfying the requirements on v_1, \dots, v_p ;

- If the last rule used is rule 5, we can use the inductive hypothesis to infer the existence of the following terms:

$$\begin{aligned}
M_g & : B_{\mathbb{A}}^{q_1} \otimes \dots \otimes B_{\mathbb{A}}^{q_n} \multimap B_{\mathbb{A}}^q \\
M_{f_1} & : B_{\mathbb{A}}^{s_1^1} \otimes \dots \otimes B_{\mathbb{A}}^{s_{n+2}^1} \multimap B_{\mathbb{A}}^{s^1} \\
& \vdots \\
M_{f_k} & : B_{\mathbb{A}}^{s_1^k} \otimes \dots \otimes B_{\mathbb{A}}^{s_{n+2}^k} \multimap B_{\mathbb{A}}^{s^k}.
\end{aligned}$$

Let now $u = \max\{q, s^1, \dots, s^k\}$. By proposition 2:

$$\begin{aligned}
M_g & : B_{\mathbb{A}}^{q_1+u-q} \otimes \dots \otimes B_{\mathbb{A}}^{q_n+u-q} \multimap B_{\mathbb{A}}^u \\
M_{f_1} & : B_{\mathbb{A}}^{s_1^1+u-s^1} \otimes \dots \otimes B_{\mathbb{A}}^{s_{n+2}^1+u-s^1} \multimap B_{\mathbb{A}}^u \\
& \vdots \\
M_{f_k} & : B_{\mathbb{A}}^{s_1^k+u-s^k} \otimes \dots \otimes B_{\mathbb{A}}^{s_{n+2}^k+u-s^k} \multimap B_{\mathbb{A}}^u.
\end{aligned}$$

Using lemma 2, we can easily build (for suitable positive integers v_1, \dots, v_{n+2}) the terms

$$\begin{aligned}
N_g & : B_{\mathbb{A}}^{v_2} \otimes \dots \otimes B_{\mathbb{A}}^{v_{n+1}} \multimap B_{\mathbb{A}}^u \\
N_{f_1} & : B_{\mathbb{A}}^{v_1} \otimes \dots \otimes B_{\mathbb{A}}^{v_{n+2}} \multimap B_{\mathbb{A}}^u \\
& \vdots \\
N_{f_k} & : B_{\mathbb{A}}^{v_1} \otimes \dots \otimes B_{\mathbb{A}}^{v_{n+2}} \multimap B_{\mathbb{A}}^u.
\end{aligned}$$

that are functionally equivalent to $M_g, M_{f_1}, \dots, M_{f_k}$ (respectively). The term $M_h \equiv \mathbf{Rec}(N_{f_1} \cdots N_{f_k} N_g)$ encodes h and can be given the type $B_{\mathbb{A}}^{v_1} \otimes \dots \otimes B_{\mathbb{A}}^{v_{n+1}} \multimap B_{\mathbb{A}}^u$ satisfying the requirements on v_1, \dots, v_{n+1} .

6 Extensions

6.1 Embedding Safe Recursion

Safe recursion is a function algebra generating all the polynomial functions in the form $f : \mathbb{A}^n \rightarrow \mathbb{A}$ where $\mathbb{A} = (\{e, c_1, \dots, c_k\}, \mathcal{R})$ is a word algebra. We can define the notion of *safe sorting* by way of a set of rules that assign a pair (m, l) (where $m + l = n$) to every well-sorted function $f : \mathbb{A}^n \rightarrow \mathbb{A}$; the first m arguments of f are called *normal*, while the remaining l are *safe*. **HOLRR** cannot directly capture safe recursion; however, we can easily define a sound extension **HOLRR**⁺ of **HOLRR**, embedding safe recursion by the introduction of three new combinators **Destroy**, **Head** and **Branch**, typable as follows

$$\begin{aligned}
& \frac{n \in \mathbb{N}}{\vdash \mathbf{Destroy} : B_{\mathbb{A}}^n \multimap B_{\mathbb{A}}^n} I_{\mathbf{Destroy}} \\
& \frac{n \in \mathbb{N}}{\vdash \mathbf{Head} : B_{\mathbb{A}}^n \multimap B_{\mathbb{A}}^n} I_{\mathbf{Head}} \\
& \frac{n \in \mathbb{N}}{\vdash \mathbf{Branch} : B_{\mathbb{A}}^n \otimes B_{\mathbb{A}}^n \otimes B_{\mathbb{A}}^n \multimap B_{\mathbb{A}}^n} I_{\mathbf{Branch}}
\end{aligned}$$

and subject to the following normalization rules:

$$\begin{aligned}
\mathbf{Destroy} \ e &\mapsto e \\
\mathbf{Destroy} \ ct &\mapsto t \\
\mathbf{Head} \ e &\mapsto e \\
\mathbf{Head} \ ct &\mapsto c \\
\mathbf{Branch} \ e \ t_1 \ t_2 &\mapsto t_1 \\
\mathbf{Branch} \ ct \ t_1 \ t_2 &\mapsto t_2
\end{aligned}$$

Theorem 4 *If $f : \mathbb{A}^n \rightarrow \mathbb{A}$ can be safely sorted by (m, l) (where $m + l = n$), then there is a term M encoding f that can be assigned type*

$$B_{\mathbb{A}}^{p_1} \otimes \cdots \otimes B_{\mathbb{A}}^{p_m} \otimes \overbrace{B_{\mathbb{A}}^p \otimes \cdots \otimes B_{\mathbb{A}}^p}^{l \text{ times}} \multimap B_{\mathbb{A}}^p$$

where $p, p_1, \dots, p_m \in \mathbb{N}$ and $p < p_1, \dots, p_m$.

6.2 Relaxing Word-Ramification Conditions

If we relax some constraints in our definition of word ramification conditions, we immediately get outside **PTIME**. For example, if we allowed $\mathbb{L}(A)$ to be equal to $\mathbb{L}(C)$ in rule $E_{\multimap}^{\mathbb{A}}$, we would be able to build a term M with $\vdash M : B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0$ such that

$$M \ c_{i_1}^{\mathbb{B}} \dots c_{i_n}^{\mathbb{B}} \ c_3^{\mathbb{B}} \rightsquigarrow^* c_{i_1}^{\mathbb{B}} c_{i_1}^{\mathbb{B}} c_{i_2}^{\mathbb{B}} c_{i_2}^{\mathbb{B}} \dots c_{i_n}^{\mathbb{B}} c_{i_n}^{\mathbb{B}} c_3^{\mathbb{B}}.$$

Iterating M , we easily obtain an exponential behaviour. Assume now that, in rule $E_{\multimap}^{\mathbb{A}}$, B_1, \dots, B_n can be arbitrary types. The term $\lambda x. \lambda y. \lambda z. x(yz)$ encoding function composition can be given type $(B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0) \multimap (B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0) \multimap (B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0)$; using the obvious variation of lemma ??, we obtain a term N with type $(B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0) \multimap (B_{\mathbb{B}}^0 \multimap B_{\mathbb{B}}^0)$ encoding self duplication. Again, an exponential behaviour can be obtained by iterating N . Another similar example can be built starting from term $c_1^{\mathbb{C}}$, having type $B_{\mathbb{C}}^0 \multimap B_{\mathbb{C}}^0 \multimap B_{\mathbb{C}}^0$.

7 Conclusions

We have proposed a higher-order system allowing for time-bounded computations at all types. Our aim has been to obtain a higher-order system embedding in a natural way Leivant's and Bellantoni-Cook's systems. In view of Murawsky-Ong result [MO00], this cannot be achieved in systems like **LLL** or **LAL**.

One last remark is due on the **PTIME**-soundness result. For a (closed) term $M : A$, we obtained a bound $O((|M| + |A|)^{f(\mathbb{R}(M))})$, for some suitable function f . Now, the exponent *does depend* on M . However, as for **LLL**, when we see the term $M : B \multimap A$ as a program, the time needed to compute M on an argument $N : B$, is $O((|MN| + |A|)^{f(\mathbb{R}(MN))})$. Since $|M|$, $|A|$, and $\mathbb{R}(M)$ are all constants, the bound is $O(|N|^{f(\mathbb{R}(N))})$. The time needed to compute the program M is thus a polynomial on all inputs with a constant recursion depth, which is the case when N is a closed normal form of a base type. More work should be done to characterize exactly those higher-order types whose normal forms have all the same recursion depth.

References

- [AR02] Andrea Asperti and Luca Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic*, 3(1):137–175, 2002.

- [Asp98] Andrea Asperti. Light affine logic. In *Proceedings of the 13th IEEE Symposium on Logic in Computer Science (LICS)*, pages 300–308, 1998.
- [BC92] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.
- [BNS00] Stephen Bellantoni, Karl Heinz Niggl, and Helmut Schwichtenberg. Higher type recursion, ramification and polynomial time. *Annals of Pure and Applied Logic*, 104:17–30, 2000.
- [BS01] Stephen Bellantoni and Helmut Schwichtenberg. Feasible computation with higher types. Marktobendorf Summer School Proceedings, 2001.
- [Gir98] Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.
- [Hof97] Martin Hofmann. A mixed modal/linear lambda calculus with applications to bellantoni-cook safe recursion. In *Proceedings of the 11th International Workshop on Computer Science Logic*, pages 275–294, 1997.
- [Hof99a] Martin Hofmann. Linear types and non-size-increasing polynomial time computation. In *Logic in Computer Science*, pages 464–473, 1999.
- [Hof99b] Martin Hofmann. *Type systems for polynomial-time computation*. Habilitationsschrift, Darmstadt University of Technology, 1999.
- [Hof00] Martin Hofmann. Safe recursion with higher types and BCK-algebra. *Annals of Pure and Applied Logic*, 104:113–166, 2000.
- [Laf02] Yves Lafont. Soft linear logic and polynomial time. To appear, 2002.
- [Lei93] Daniel Leivant. Stratified functional programs and computational complexity. In *Proceedings of 20th ACM Symposium on Principles of Programming Languages*, pages 325–333, 1993.
- [Lei95] Daniel Leivant. Ramified recurrence and computational complexity I: word recurrence and poly-time. In *Feasible Mathematics II*, pages 320–343. Birkhäuser, 1995.
- [Lei99] Daniel Leivant. Ramified recurrence and computational complexity III: Higher type recurrence and elementary complexity. *Annals of Pure and Applied Logic*, 96:209–229, 1999.
- [LM94] Daniel Leivant and Jean-Yves Marion. Ramified recurrence and computational complexity II: Substitution and poly-space. In *Proceedings of 8th International Workshop on Computer Science Logic (CSL)*, pages 486–500, 1994.
- [LM00] Daniel Leivant and Jean-Yves Marion. Predicative recurrence and computational complexity IV: Predicative functionals and poly-space. *Information and Computation*, 2000. to appear.
- [MO00] Andrzej Murawski and Luke Ong. Can safe recursion be interpreted in light logic. In *Workshop on Implicit Computational Complexity*, 2000.