The Geometry of Linear Higher-Order Recursion

UGO DAL LAGO
Università di Bologna

Imposing linearity and ramification constraints allows to weaken higher-order (primitive) recursion in such a way that the class of representable functions equals the class of polynomial-time computable functions, as the works by Leivant, Hofmann, and others show. This article shows that fine-tuning these two constraints leads to different expressive strengths, some of them lying well beyond polynomial time. This is done by introducing a new semantics, called algebraic context semantics. The framework stems from Gonthier's original work (itself a model of Girard's geometry of interaction) and turns out to be a versatile and powerful tool for the quantitative analysis of normalization in the lambda calculus with constants and higher-order recursion.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Lambda calculus and related systems*; F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Machine-independent complexity*

General Terms: Languages, Performance, Theory

Additional Key Words and Phrases: Geometry of interaction, higher-order recursion, implicit computational complexity, lambda calculus, type systems

ACM Reference Format:

Dal Lago, U. 2009. The geometry of linear higher-order recursion. ACM Trans. Comput. Logic 10, 2, Article 8 (February 2009), 38 pages. DOI = 10.1145/1462179.1462180 http://doi.acm.org/10.1145/1462179.1462180

1. INTRODUCTION

Implicit computational complexity aims at giving machine-independent characterizations of complexity classes. In recent years, the field has produced a number of interesting results. Many of them relate complexity classes to function algebras, typed lambda calculi and logics by introducing appropriate restrictions to (higher-order) primitive recursion or second-order linear logic. The resulting subsystems are then shown to correspond to complexity classes by

U. Dal Lago is partially supported by PRIN projects PROTOCOLLO (2002) and FOLLIA (2004).
Author's address: U. Dal Lago, Dipartimento di Scienze dell'Informazione, Mura Anteo Zamboni 7, 40127 Bologna, Italy; email: dallago@cs.unibo.it.

way of a number of different, heterogeneous techniques. Many kinds of constraints have been shown to be useful in this context; this includes ramification [Bellantoni and Cook 1992; Leivant 1993, 1999b], linear types [Hofmann 2000; Bellantoni et al. 2000; Leivant 1999a; Dal Lago et al. 2003], and restricted exponentials [Girard 1998; Lafont 2004]. However, the situation is far from satisfactory. There are still many open problems: For example, it is not yet clear what the consequences are of combining different constraints. Moreover, using such systems as a foundation for resource-aware programming languages relies heavily on their ability to capture interesting algorithms. Despite some recent progress [Hofmann 1999; Bonfante et al. 2004], a lot of work still has to be done.

Undoubtedly, what is still lacking in this field is a powerful and simple mathematical framework for the analysis of quantitative aspects of computation. Indeed, existing systems have been often studied using ad hoc techniques which cannot be easily adapted to other systems. A unifying framework would not just make the task of proving correspondences between systems and complexity classes simpler, but could be possibly used *itself* as a basis for introducing resource consciousness into programming languages. We believe that ideal candidates to pursue these goals are Girard's geometry of interaction [Girard 1989, 1988] and related frameworks, such as context semantics [Gonthier et al. 1992; Mairson 2002]. Using the aforementioned techniques as tools in the study of complexity of normalization has already been done by Baillot and Pedicini [2001] in the context of elementary linear logic, while game models being fully abstract with respect to operational theory of improvement [Sands 1991] have recently been proposed by Ghica [2005]. Ordinal analysis has already been proved useful to the study of ramified systems (e.g., Ostrin and Wainer [2002], Simmons [2005]) but, to the author's knowledge, the underlying framework has not been applied to linear calculi. Similarly, Leivant's intrinsic reasoning framework Leivant [2004, 2002] can help defining and studying restrictions on first-order arithmetic inducing complexity bounds on provably total functions: However, the consequences of linearity conditions cannot be easily captured and studied in the framework.

In this article, we introduce a new semantical framework for higher-order recursion, called algebraic context semantics. It is inspired by context semantics, but designed to be a tool for proving *quantitative* rather than qualitative properties of programs. As we will see, it turns out to be of great help when analyzing quantitative aspects of normalization in presence of linearity and ramification constraints. Informally, algebraic context semantics allows to prove bounds on the *algebraic potential size* of System T terms, where the algebraic potential size of any term M is the maximum size of free algebra terms which appear as subterms of reducts of M. As a preliminary result, the algebraic potential size is shown to be a bound to normalization time, modulo a polynomial overhead. Consequently, bounds obtained through context semantics translate into bounds to normalization time.

Main results of this work are sharp characterizations of the expressive power of various fragments of System T. Almost all of them are novel. Noticeably, these results are obtained in a uniform way and, as a consequence, most of the involved work has been factorized over the subsystems and done just once.

Moreover, we do not simply prove that the class of representable first-order functions equals complexity classes, but instead we give bounds on the time needed to normalize *any* term. This makes our results stronger than similar ones from the literature [Hofmann 2000; Bellantoni et al. 2000; Leivant 1999a]. Our work gives some answers to a fundamental question implicitly raised by Hofmann [2000]: Are linearity conditions sufficient to keep the expressive power of higher-order recursion equal to that of first-order recursion? In particular, a positive answer can be given in case ramification does not hold. The methodology introduced here can be applied to multiplicative and exponential linear logic [Dal Lago 2006], allowing to reprove soundness results for various subsystems of the logic.

The rest of the article is organized as follows: In Section 2 a call-by-value lambda calculus will be described as well as an operational semantics for it; in Section 3 we will define ramification and linearity conditions on the underlying type system, together with subsystems induced by these constraints; in Section 4 we motivate and introduce algebraic context semantics, while in Section 5 we will use it to give bounds on the complexity of normalization. Section 6 is devoted to completeness results.

2. SYNTAX

In this section, we will give some details on our reference system, namely a formulation of Gödel's T in the style of Joachimski and Matthes [2003]. The definitions will be standard. The only unusual aspect of our syntax is the adoption of weak call-by-value reduction. This will help in keeping the language of terms and the underlying type system simpler.

Data will be represented by terms in some free algebras. As will be shown, different free algebras do not necessarily behave in the same way from a complexity viewpoint, as opposed to what happens in computability theory. As a consequence, we cannot restrict ourselves to a canonical free algebra and need to keep all of them in our framework. 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 $\mathcal{R}_\mathbb{A} : \mathcal{C}_\mathbb{A} \to \mathbb{N}$ 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:

(1) $\mathcal{R}_\mathbb{A}(c_i^\mathbb{A}) = 0$ for one (and only one) $i \in \{1, \dots, k(\mathbb{A})\}$;
(2) $\mathcal{R}_\mathbb{A}(c_j^\mathbb{A}) = 1$ 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 $c_1^\mathbb{A}, \dots, c_{k(\mathbb{A})-1}^\mathbb{A}$ will denote the elements of $\mathcal{C}_\mathbb{A}$ whose arity is 1. $\mathbb{U} = (\{c_1^\mathbb{U}, c_2^\mathbb{U}\}, \mathcal{R}_\mathbb{U})$ is the word algebra of unary strings. $\mathbb{B} = (\{c_1^\mathbb{B}, c_2^\mathbb{B}, c_3^\mathbb{B}\}, \mathcal{R}_\mathbb{B})$ is the word algebra of binary strings. $\mathbb{C} = (\{c_1^\mathbb{C}, c_2^\mathbb{C}\}, \mathcal{R}_\mathbb{C})$, where $\mathcal{R}_\mathbb{C}(c_1^\mathbb{C}) = 2$ and $\mathcal{R}_\mathbb{C}(c_2^\mathbb{C}) = 0$ is the free algebra of binary trees. $\mathbb{D} = (\{c_1^\mathbb{D}, c_2^\mathbb{D}, c_3^\mathbb{D}\}, \mathcal{R}_\mathbb{D})$, where $\mathcal{R}_\mathbb{D}(c_1^\mathbb{D}) = \mathcal{R}_\mathbb{D}(c_2^\mathbb{D}) = 2$ and $\mathcal{R}_\mathbb{D}(c_3^\mathbb{D}) = 0$ is the free algebra of binary trees with binary labels. Natural numbers can be encoded by terms in $\mathbb{U}$: $\ulcorner 0 \urcorner = c_2^\mathbb{U}$ and $\ulcorner n+1 \urcorner = c_1^\mathbb{U} \ulcorner n \urcorner$ for all n. In the same vein, elements of $\{0,1\}^*$ are in one-to-one correspondence to terms in $\mathbb{B}$: $\ulcorner \varepsilon \urcorner = c_3^\mathbb{B}$, while for all $s \in \{0,1\}^*$, $\ulcorner 0s \urcorner = c_1^\mathbb{B} \ulcorner s \urcorner$ and $\ulcorner 1s \urcorner = c_2^\mathbb{B} \ulcorner s \urcorner$. When this does not cause ambiguity, $\mathcal{C}_\mathbb{A}$ and $\mathcal{R}_\mathbb{A}$ will be denoted by $\mathcal{C}$ and $\mathcal{R}$, respectively.

$\mathscr{A}$ will be a fixed, finite family $\{\mathbb{A}_1, \ldots, \mathbb{A}_n\}$ of free algebras whose constructor sets $\mathcal{C}_{\mathbb{A}_1}, \ldots, \mathcal{C}_{\mathbb{A}_n}$ are assumed to be pairwise disjoint. We will hereby assume $\mathbb{U}, \mathbb{B}, \mathbb{C}$, and $\mathbb{D}$ to be in $\mathscr{A}$. $\mathscr{K}_{\mathscr{A}}$ is the maximum arity of constructors of free algebras in $\mathscr{A}$, namely, the natural number

$$\max_{\mathbb{A}\in\mathscr{A}} \max_{c\in\mathcal{C}_{\mathbb{A}}} \mathcal{R}_{\mathbb{A}}(c).$$

$\mathscr{E}_{\mathbb{A}}$ is the set of terms for the algebra $\mathbb{A}$, while $\mathscr{E}_{\mathscr{A}}$ is the union of $\mathscr{E}_{\mathbb{A}}$ over all algebras $\mathbb{A}$ in $\mathscr{A}$.

Programs will be written in a fairly standard lambda calculus with constants (corresponding to free algebra constructors) and recursion. The latter will not be a combinator but a term former, as in Joachimski and Matthes [2003]. Moreover, we will use a term former for conditional, keeping it distinct from the one for recursion. This apparent redundancy is actually needed in presence of ramification (see, e.g., Leivant [1993]). The language $\mathscr{M}_{\mathscr{A}}$ of *terms* is defined by the productions

$$M ::= x \mid c \mid MM \mid \lambda x.M \mid M \;\{\!\!\{M, \ldots, M\}\!\!\} \mid M \;\langle\!\langle M, \ldots, M\rangle\!\rangle$$

where c ranges over the constructors for the free algebras in $\mathscr{A}$. Term formers $\cdot \;\{\!\!\{\cdot, \ldots, \cdot\}\!\!\}$ and $\cdot \;\langle\!\langle\cdot, \ldots, \cdot\rangle\!\rangle$ are *conditional* and *recursion* term formers, respectively.

The language $\mathscr{T}_{\mathscr{A}}$ of *types* is defined by the productions

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

where n ranges over $\mathbb{N}$ and $\mathbb{A}$ ranges over $\mathscr{A}$. Indexing base types is needed to define ramification conditions [Leivant 1993]; $\mathbb{A}^n$, in particular, is not a Cartesian product. The notation $A \overset{n}{\multimap} B$ is defined by induction on n as follows: $A \overset{0}{\multimap} B$ is just B, while $A \overset{n+1}{\multimap} B$ is $A \multimap (A \overset{n}{\multimap} B)$. The *level* $V(A) \in \mathbb{N}$ of a type A is defined by induction on the structure of A. We have

$$\begin{aligned} V(\mathbb{A}^n) &= n; \\ V(A \multimap B) &= \max\{V(A), V(B)\}. \end{aligned}$$

When this does not cause ambiguity, we will denote a base type $\mathbb{A}^n$ simply by $\mathbb{A}$.

The rules in Figure 1 define the assignment of types in $\mathscr{T}_{\mathscr{A}}$ to terms in $\mathscr{M}_{\mathscr{A}}$. A type derivation π with 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 mark M as a *typeable* term. A type derivation $\pi : \Gamma \vdash M : A$ is in *standard form* if the typing rule W is used only when necessary, that is, immediately before an instance of $I_{\multimap}$. We will hereby assume to work with type derivations in standard form. This restriction does not affect the class of typeable terms.

The *recursion depth* $R(\pi)$ of a type derivation $\pi : \Gamma \vdash M : A$ is the biggest number of $E^R_{\mathbb{A}}$ instances on any path from the root to a leaf in π. The *highest tier* $I(\pi)$ of a type derivation $\pi : \Gamma \vdash M : A$ is the maximum integer i such that there is an instance

$$\frac{\pi_1 \quad \ldots \quad \pi_n \quad \Delta \vdash L : \mathbb{A}^i}{\Gamma, \Delta \vdash L \;\langle\!\langle M_1, \ldots, M_n\rangle\!\rangle : C}\; E^R_{\mathbb{A}}$$

$$\frac{}{x : A \vdash x : A}\ A \qquad \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}\ W \qquad \frac{\Gamma, x : A, y : A \vdash M : B}{\Gamma, z : A \vdash M\{z/x, z/y\} : B}\ C$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\ I_{\multimap} \qquad \frac{\Gamma \vdash M : A \multimap B \quad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\ E_{\multimap}$$

$$\frac{n \in \mathbb{N} \quad c \in \mathcal{C}_{\mathbb{A}}}{\vdash c : \mathbb{A}^n \overset{\mathcal{R}(c)}{\multimap} \mathbb{A}^n}\ I_{\mathbb{A}} \qquad \frac{\Gamma_i \vdash M_{c_i^{\mathbb{A}}} : \mathbb{A}^m \overset{\mathcal{R}(c_i^{\mathbb{A}})}{\multimap} C \quad \Delta \vdash L : \mathbb{A}^m}{\Gamma_1, \ldots, \Gamma_n, \Delta \vdash L\ \{\!\{M_{c_1^{\mathbb{A}}} \cdots M_{c_{k(\mathbb{A})}^{\mathbb{A}}}\}\!\} : C}\ E_{\mathbb{A}}^C$$

$$\frac{\Gamma_i \vdash M_{c_i^{\mathbb{A}}} : \mathbb{A}^m \overset{\mathcal{R}(c_i^{\mathbb{A}})}{\multimap} C \overset{\mathcal{R}(c_i^{\mathbb{A}})}{\multimap} C \quad \Delta \vdash L : \mathbb{A}^m}{\Gamma_1, \ldots, \Gamma_n, \Delta \vdash L\ \langle\!\langle M_{c_1^{\mathbb{A}}} \cdots M_{c_{k(\mathbb{A})}^{\mathbb{A}}}\rangle\!\rangle : C}\ E_{\mathbb{A}}^R$$

Fig. 1. Type assignment rules.

$$\begin{aligned}
(\lambda x.M)V &\to M\{V/x\} \\
c_i^{\mathbb{A}}\ t_1, \ldots, t_{\mathcal{R}(c_i^{\mathbb{A}})}\{\!\{M_{c_1^{\mathbb{A}}}, \ldots, M_{c_k^{\mathbb{A}}}\}\!\} &\to M_{c_i^{\mathbb{A}}}\ t_1 \cdots t_{\mathcal{R}(c_i^{\mathbb{A}})} \\
c_i^{\mathbb{A}}\ t_1, \ldots, t_{\mathcal{R}(c_i^{\mathbb{A}})}\ \langle\!\langle M_{c_1^{\mathbb{A}}}, \ldots, M_{c_k^{\mathbb{A}}}\rangle\!\rangle &\to M_{c_i^{\mathbb{A}}}\ t_1 \cdots t_{\mathcal{R}(c_i^{\mathbb{A}})} \\
&\qquad (t_1\ \langle\!\langle M_{c_1^{\mathbb{A}}}, \ldots, M_{c_k^{\mathbb{A}}}\rangle\!\rangle) \\
&\qquad \cdots \\
&\qquad (t_{\mathcal{R}(c_i^{\mathbb{A}})}\ \langle\!\langle M_{c_1^{\mathbb{A}}}, \ldots, M_{c_k^{\mathbb{A}}}\rangle\!\rangle)
\end{aligned}$$

Fig. 2. Normalization on terms.

of $E_{\mathbb{A}}^R$ inside π.

Values are defined by the productions

$$\begin{aligned} V &::= x \mid \lambda x.M \mid T; \\ T &::= c \mid TT. \end{aligned}$$

Here c ranges over constructors. Reduction is weak and call-by-value. The reduction rule $\to$ on $\mathscr{M}_{\mathscr{A}}$ is given in Figure 2. We will forbid firing a redex under an abstraction or inside a recursion or a conditional. In other words, we will define $\rightsquigarrow$ from $\to$ by the following rules.

$$\frac{M \to N}{M \rightsquigarrow N} \qquad \frac{M \rightsquigarrow N}{ML \rightsquigarrow NL} \qquad \frac{M \rightsquigarrow N}{LM \rightsquigarrow LN}$$

$$\frac{M \rightsquigarrow N}{M\ \{\!\{L_1, \ldots, L_n\}\!\} \rightsquigarrow N\ \{\!\{L_1, \ldots, L_n\}\!\}} \qquad \frac{M \rightsquigarrow N}{M\ \langle\!\langle L_1, \ldots, L_n\rangle\!\rangle \rightsquigarrow N\ \langle\!\langle L_1, \ldots, L_n\rangle\!\rangle}$$

Redexes in the form $(\lambda x.M)V$ are called *beta redexes*; those like $t\ \{\!\{M_1, \ldots, M_n\}\!\}$ are called *conditional redexes*; those in the form $t\ \langle\!\langle M_1, \ldots, M_n\rangle\!\rangle$ are *recursive redexes*. The *argument* of the beta redex $(\lambda x.M)V$ is V, while that of $t\ \{\!\{M_1, \ldots, M_n\}\!\}$ and $t\ \langle\!\langle M_1, \ldots, M_n\rangle\!\rangle$ is t. As usual, $\rightsquigarrow^*$ and $\rightsquigarrow^+$ denote the reflexive and transitive closure of $\rightsquigarrow$ and the transitive closure of $\rightsquigarrow$, respectively.

PROPOSITION 2.1. *If $\vdash M : \mathbb{A}^n$, then the (unique) normal form of M is a free algebra term t.*

PROOF. In this proof, terms from the grammar $T ::= c \mid TT$ (where c ranges over constructors) are dubbed *algebraic*. We prove the following stronger claim by induction on M: If $\vdash M : A$ and M is a normal form, then it must be a value. We distinguish some cases next.

—A variable cannot be typed in the empty context, so M cannot be a variable.

—If M is a constant or an abstraction, then it is a value by definition.

—If M is an application NL, then there is a type B such that both $\vdash N : B \multimap A$ and $\vdash L : B$. By induction hypothesis both N and L must be values. But N cannot be an abstraction (because otherwise NL would be a redex) nor a variable (because a variable cannot be typed in the empty context). As a consequence, N must be algebraic. Every algebraic term, however, has type $\mathbb{A}^i \overset{n}{\multimap} \mathbb{A}^i$ where $n \geq 0$. Clearly, this implies $n \geq 1$ and $B = \mathbb{A}^i$. This, in turn, implies L to be algebraic (it cannot be a variable nor an abstraction). So, M is itself algebraic.

—If M is $N \{\!\{M_1, \dots, M_n\}\!\}$, then N must be a value such that $\vdash N : \mathbb{A}^i$. As a consequence, it must be a free algebraic term t. But this is a contradiction, since M is assumed a normal form.

—If M is $N \langle\!\langle M_1, \dots, M_n \rangle\!\rangle$, then we can proceed exactly as in the previous case.

This concludes the proof, since the relation $\rightsquigarrow$ enjoys a one-step diamond property (see Dal Lago and Martini [2006]). □

It should be now clear that the usual recursion combinator $\mathbf{R}$ can be retrieved by putting $\mathbf{R} \equiv \lambda x.\lambda y_1.\dots.\lambda y_n.x \langle\!\langle y_1, \dots, y_n \rangle\!\rangle$.

The *size* $|M|$ of a term M is defined as follows by induction on the structure of M.

$$
\begin{aligned}
|x| = |c| &= 1\\
|\lambda x.M| &= |M| + 1\\
|MN| &= |M| + |N|\\
|M \langle\!\langle M_1, \dots, M_n \rangle\!\rangle| = |M \{\!\{M_1, \dots, M_n\}\!\}| &= |M| + |M_1| + \dots + |M_n| + n
\end{aligned}
$$

Notice that, in particular, $|t|$ equals the number of constructors in t for every free algebra term t.

3. SUBSYSTEMS

The system as it has been just defined is equivalent to Gödel System T and, as a consequence, its expressive power equals the one of first-order arithmetic. We are here interested in two different conditions on programs, which can both be expressed as constraints on the underlying type system.

—First of all, we can selectively enforce *linearity* by limiting the applicability of contraction rule C to types in a class $\mathsf{D} \subseteq \mathscr{T}_{\mathscr{A}}$. Accordingly, the constraint

$cod(\Gamma_i) \subseteq \mathsf{D}$ must be satisfied in rule $E^R_{\mathbb{A}}$ (for every $i \in \{1, \dots, n\}$). In this way, we obtain a system $\mathsf{H}(\mathsf{D})$. As an example, $\mathsf{H}(\emptyset)$ is a system where rule C is not allowed on any type and contexts Γ_i are always empty in rule $E^R_{\mathbb{A}}$.

—Secondly, we can introduce a *ramification* condition on the system. This can be done in a straightforward way by adding the premise $m > V(C)$ to rule $E^R_{\mathbb{A}}$. This corresponds to imposing the tier of the recurrence argument to be strictly higher than the tier of the result (analogously to Leivant [1993]). Indeed, m is the integer indexing the type of the recurrence argument, while $V(C)$ is the maximum integer appearing as an index in C, which is the type of the result. For every system $\mathsf{H}(\mathsf{D})$, we obtain in this way a ramified system $\mathsf{RH}(\mathsf{D})$.

The constraint $cod(\Gamma_i) \subseteq \mathsf{D}$ in instances of rule $E^R_{\mathbb{A}}$ is needed to preserve linearity during reduction: If $c_i^{\mathbb{A}} t_1 \dots t_{\mathcal{R}(c_i^{\mathbb{A}})} \langle\langle M_1, \dots, M_{k(\mathbb{A})}\rangle\rangle$ is a recursive redex where M_i has a free variable x of type $A \notin \mathsf{D}$, firing the redex would produce a term with two occurrences of x.

Let us define two classes of types. We have

$$\begin{aligned} \mathsf{W} &= \{\mathbb{A}^n \mid \mathbb{A} \in \mathscr{A} \textit{ is a word algebra}\}; \\ \mathsf{A} &= \{\mathbb{A}^n \mid \mathbb{A} \in \mathscr{A}\}. \end{aligned}$$

In the rest of this article, we will investigate the expressive power of some subsystems $\mathsf{H}(\mathsf{D})$ and $\mathsf{RH}(\mathsf{D})$ where $\mathsf{D} \subseteq \mathsf{A}$. The following table reports the obtained results.

	A	W	∅
H(·)	**FR**	**FR**	**FR**
RH(·)	**FE**	**FP**	**FP**

Here, **FP** (respectively, **FE**) is the class of functions which can be computed in polynomial (respectively, elementary) time. **FR**, on the other hand, is the class of (first-order) primitive recursive functions, which equals the class of functions which can be computed in time bounded by a primitive recursive function. For example, $\mathsf{RH}(\mathsf{A})$ is proved sound and complete with respect to elementary time, while $\mathsf{H}(\emptyset)$ is shown to capture (first-order) primitive recursion.

Forbidding contraction on higher-order types is quite common and has been extensively used as a tool to restrict the class of representable functions inside System T [Hofmann 2000; Bellantoni et al. 2000; Leivant 1999a]. The correspondence between $\mathsf{RH}(\mathsf{W})$ and **FP** is well known from the literature [Hofmann 2000; Bellantoni et al. 2000], although in a slightly different form. To the author's knowledge, all the other characterization results are novel. Similar results can be ascribed to Leivant and Marion [1994] and Leivant [1999b], but they do not take linearity constraints into account.

Notice that, in presence of ramification, going from W to A dramatically increases the expressive power, while going from W to ∅ does not cause any loss of expressivity. The "phase-transition" occurring when switching from $\mathsf{RH}(\mathsf{W})$ to $\mathsf{RH}(\mathsf{A})$ is really surprising, since the only difference between these two systems is the class of types to which linearity applies: In one case we only have word algebras, while in the other case we have all free algebras.

4. ALGEBRAIC CONTEXT SEMANTICS

In this section, we will introduce algebraic context semantics, showing how bounds on the normalization time of any term M can be inferred from its semantics.

The first result we need relates the complexity of normalizing any given term M to the size of free algebra terms appearing as subterms of reducts of M. The *algebraic potential size* $A(M)$ of a typeable term M is the maximum natural number n such that $M \rightsquigarrow^* N$ and there is a redex in N whose argument is a free algebra term t with $|t| = n$. Since the calculus is strongly normalizing, there is always a finite bound to the size of reducts of a term and, as a consequence, the previous definition is well posed. According to the following result, the algebraic potential size of a term M such that $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$ is an overestimate on the time needed to normalize the term (modulo some polynomials that only depends on $R(\pi)$).

PROPOSITION 4.1. *For every $d \in \mathbb{N}$ there are polynomials $p_d, q_d : \mathbb{N}^2 \to \mathbb{N}$ such that whenever $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$ and $M \rightsquigarrow^n N$, then $n \leq p_{R(\pi)}(|M|, A(M))$ and $|N| \leq q_{R(\pi)}(|M|, A(M))$.*

PROOF. Let us first observe that the number of recursive redexes fired during normalization of M is bounded by $s_{R(\pi)}(|M|, A(M))$, where

$$s_d(x, y) = xy^d.$$

Indeed, consider subterms of M in the form $L\langle\langle N_1, \ldots, N_k\rangle\rangle$. Clearly, there are at most $|M|$ such terms. Moreover, each such subterm can result in at most $A(M)^{R(\pi)}$ recursive redexes. Indeed, it can be copied at most $A(M)^{R(\pi)-1}$ times, and each copy can itself result in at most $A(M)$ recursive redexes. Observe that for any subterm $L\langle\langle N_1, \ldots, N_k\rangle\rangle$ of any reduct of a term M, there are at most $|M|$ variable occurrences in $N_1, \ldots, N_k$. Similarly for abstractions. Indeed:

—reducing under the scope of a recursion is prohibited;

—variable occurrences in the scope of a recursion can possibly be substituted for other variables or for free algebra terms.

Now, notice that firing any beta or conditional redex does not increase the number of variable occurrences in the term. Conversely, firing a recursive redex can make it bigger by at most $\mathscr{K}_{\mathscr{A}}|M|$. Similarly, firing any beta or conditional redex does not increase the number of abstractions in the term, while firing a recursive redex can make it bigger by at most $\mathscr{K}_{\mathscr{A}}|M|$. We can conclude that the number of beta redexes in the form $(\lambda x.M)t$ fired during normalization of M (let us call them *algebraic redexes*) is at most $\mathscr{K}_{\mathscr{A}}|M|(s_{R(\pi)}(|M|, A(M)) + 1)$ and, moreover, they can make the term increase in size by at most $A(M)(\mathscr{K}_{\mathscr{A}}|M|(s_{R(\pi)}(|M|, A(M)) + 1))^2$ altogether. Firing a recursive redex

$$c_i\ t_1, \ldots, t_{\mathcal{R}(c_i)}\langle\langle M_{c_1}, \ldots, M_{c_k}\rangle\rangle$$

can make the size of the underlying term increase by $r_{R(\pi)}(|M|, A(M))$, where

$$r_{R(\pi)}(x, y) = \mathscr{K}_{\mathscr{A}}(y + x + xy).$$

Indeed,

$$
\begin{aligned}
& |M_{c_i}\ t_1 \cdots t_{\mathcal{R}(c_i)} (t_1\ \langle\!\langle M_{c_1}, \ldots, M_{c_k} \rangle\!\rangle) \cdots (t_{\mathcal{R}(c_i)}\ \langle\!\langle M_{c_1}, \ldots, M_{c_k} \rangle\!\rangle)| \\
=\ & |M_{c_i}\ t_1 \cdots t_{\mathcal{R}(c_i)}| + |t_1\ \langle\!\langle M_{c_1}, \ldots, M_{c_k} \rangle\!\rangle)| + \cdots + |t_{\mathcal{R}(c_i)}\ \langle\!\langle M_{c_1}, \ldots, M_{c_k} \rangle\!\rangle| \\
\leq\ & |c_i\ t_1, \ldots, t_{\mathcal{R}(c_i)} \langle\!\langle M_{c_1}, \ldots, M_{c_k} \rangle\!\rangle| + \sum_{i=1}^{\mathcal{R}(c_i)} (|t_i| + |M_{c_1}| + \cdots + |M_{c_k}| + k) \\
\leq\ & |c_i\ t_1, \ldots, t_{\mathcal{R}(c_i)} \langle\!\langle M_{c_1}, \ldots, M_{c_k} \rangle\!\rangle| + \mathscr{K}_{\mathscr{A}} (A(M) + |M| + A(M)|M|)
\end{aligned}
$$

because $|M_{c_1}| + \cdots + |M_{c_k}| + k$ is bounded by $|M| + A(M)|M|$ and $|c_i\ t_1, \ldots, t_{\mathcal{R}(c_i)}|$ is bounded by $A(M)$. We can now observe that firing any redex other than algebraic or recursive ones makes the size of the term strictly smaller. As a consequence, we can argue that

$$
\begin{aligned}
q_d(x, y) &= x + (\mathscr{K}_{\mathscr{A}} (s_d(x, y) + 1)x)^2 y + s_d(x, y) r_d(x, y); \\
p_d(x, y) &= \mathscr{K}_{\mathscr{A}} (s_d(x, y) + 1)x + s_d(x, y) + q_d(x, y).
\end{aligned}
$$

This concludes the proof. □

Observe that in the statement of Proposition 4.1, it is crucial to require M to be typeable in $\mathsf{H}(\mathsf{A})$. Indeed, it is quite easy to build simply-typed (pure) lambda terms which have exponentially big normal forms, although having null algebraic potential size.

In the rest of this section, we will develop a semantics, derived from context semantics [Gonthier et al. 1992] and dubbed *algebraic context semantics*. We will use it to give bounds to the algebraic potential size of terms in subsystems we are interested in, and we will use Proposition 4.1 to derive time bounds.

Consider the term

$$
\mathbf{UnAdd} \equiv \lambda x.\lambda y.x \langle\!\langle \lambda w.\lambda z.c_1^{\mathbb{U}} z, y \rangle\!\rangle.
$$

Clearly, $\mathbf{UnAdd}\ulcorner n \urcorner \ulcorner m \urcorner \rightsquigarrow^* \ulcorner n + m \urcorner$ for every $n, m \in \mathbb{N}$. $\mathbf{UnAdd}\ulcorner 1 \urcorner \ulcorner 1 \urcorner$ will be used as a reference example throughout this section. A type derivation σ for $\mathbf{UnAdd}\ulcorner 1 \urcorner \ulcorner 1 \urcorner$ is

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash c_1^{\mathbb{U}} : \mathbb{U}^0 \multimap \mathbb{U}^0} \quad \dfrac{}{z : \mathbb{U}^0 \vdash z : \mathbb{U}^0}}{z : \mathbb{U}^0 \vdash c_1^{\mathbb{U}} z : \mathbb{U}^0}}{w : \mathbb{U}^1, z : \mathbb{U}^0 \vdash c_1^{\mathbb{U}} z : \mathbb{U}^0}}{w : \mathbb{U}^1 \vdash \lambda z.c_1^{\mathbb{U}} z : \mathbb{U}^0 \multimap \mathbb{U}^0}}{\vdash \lambda w.\lambda z.c_1^{\mathbb{U}} z : \mathbb{U}^1 \multimap \mathbb{U}^0 \multimap \mathbb{U}^0} \quad \dfrac{}{y : \mathbb{U}^0 \vdash y : \mathbb{U}^0} \quad \dfrac{}{x : \mathbb{U}^1 \vdash x : \mathbb{U}^1}}{x : \mathbb{U}^1, y : \mathbb{U}^0 \vdash x \langle\!\langle \lambda w.\lambda z.c_1^{\mathbb{U}} z, y \rangle\!\rangle : \mathbb{U}^0}}{x : \mathbb{U}^1 \vdash \lambda y.x \langle\!\langle \lambda w.\lambda z.c_1^{\mathbb{U}} z, y \rangle\!\rangle : \mathbb{U}^0 \multimap \mathbb{U}^0}}{\mathbf{UnAdd} : \mathbb{U}^1 \multimap \mathbb{U}^0 \multimap \mathbb{U}^0} \quad \eta_1 :\vdash \ulcorner 1 \urcorner : \mathbb{U}^1}{\vdash \mathbf{UnAdd}\ulcorner 1 \urcorner : \mathbb{U}^0 \multimap \mathbb{U}^0} \quad \eta_0 \vdash \ulcorner 1 \urcorner : \mathbb{U}^0}{\vdash \mathbf{UnAdd}\ulcorner 1 \urcorner \ulcorner 1 \urcorner : \mathbb{U}^0}
$$

where η_0 and η_1 are defined in the obvious way.

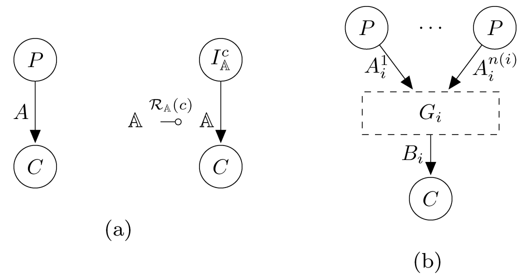


Fig. 3. Base cases.

4.1 Interaction Graphs

We will study the context semantics of interaction graphs, which are graphs corresponding to type derivations. Notice that we will not use interaction graphs as a virtual machine computing normal forms; they are merely a tool facilitating the study of language dynamics. More precisely, we will put every type derivation π in correspondence to an interaction graph G_π. The context semantics of G_π will be a set of trees $T(G_\pi)$ such that every tree T in $T(G_\pi)$ can be associated to a term $t = L(T) \in \mathscr{E}_{\mathscr{A}}$. If $\pi : \Gamma \vdash M : A$, then a lot of information on the normalization of M can be retrieved in $T(G_\pi)$: For every t appearing as an argument of a reduct of M, there is a tree $T \in T(G_\pi)$ such that $t = L(T)$. Proving this property, called completeness, is the aim of Section 4.3. Completeness, together with Proposition 4.1, is exploited in Section 5, where bounds on normalization time for certain classes of terms are inferred.

Let $\mathscr{L}_{\mathscr{A}}$ be the set

$$\{W, X, I_{\multimap}, E_{\multimap}, P, C\} \cup \bigcup_{\mathbb{A} \in \mathscr{A}} \{C^N_{\mathbb{A}}, P^R_{\mathbb{A}}, C^R_{\mathbb{A}}\} \cup \bigcup_{\mathbb{A} \in \mathscr{A}} \bigcup_{c \in \mathcal{C}_{\mathbb{A}}} \{I^c_{\mathbb{A}}\}.$$

An *interaction graph* is a graph-like structure G corresponding to a type derivation, much in the same way proof-nets are graphical representations for proofs in linear logic. It can be defined inductively as follows: An interaction graph is either the graph in Figure 3(a) or one of those in Figure 4, where $G_0, G_1, \ldots, G_{k(\mathbb{A})}$ are themselves interaction graphs as in Figure 3(b). If G is an interaction graph, then V_G denotes the set of vertices of G, E_G denotes the set of directed edges of G, α_G is a labeling function mapping every vertex in V_G to an element of $\mathscr{L}_{\mathscr{A}}$, and β_G maps every edge in E_G to a type in $\mathscr{T}_{\mathscr{A}}$. $\mathscr{G}_{\mathscr{A}}$ is the set of all interaction graphs.

Notice that each of the rules in Figures 3(a) and 4 closely corresponds to a typing rule. Given a type derivation π, we can build an interaction graph G_π corresponding to π. For example, Figure 5 reports an interaction graph G_σ where $\sigma : \vdash \mathbf{UnAdd}\ulcorner 1 \urcorner \ulcorner 1 \urcorner : \mathbb{U}^0$. Let us observe that if $\pi : \Gamma \vdash M : A$ is in standard form, then the size $|G_\pi|$ of G_π is proportional to $|M|$.

Nodes labeled with C (respectively, P) mark the conclusion (respectively, the premises) of the interaction graph. Notice that the rule corresponding to recursion (see Figure 4) allows seeing interaction graphs as nested structures,

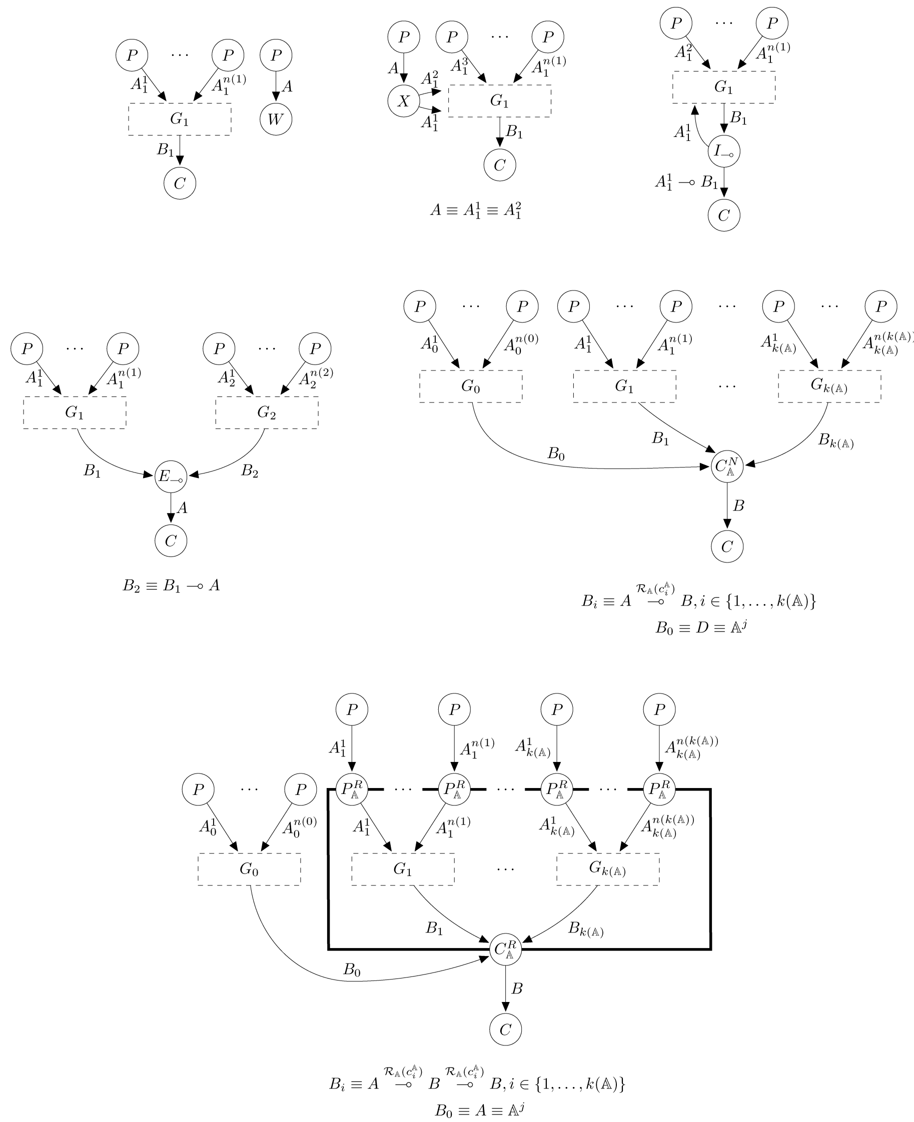


Fig. 4. Inductive cases.

where nodes labeled with $C^R_{\mathbb{A}}$ and $P^R_{\mathbb{A}}$ delimit a *box*, similarly to what happens in linear logic proof-nets. If $e \in E_G$, then the *box premise* of e, denoted $\theta_G(e)$, is the vertex labeled with $C^R_{\mathbb{A}}$ delimiting the box in which e is contained (if such a box exists, otherwise $\theta_G(e)$ is undefined). If $v \in V_G$, the box-premise of v, denoted $\theta_G(v)$, is defined similarly. In our example (see Figure 5), $\theta_G(e_i) = v$ for every $i \leq 7$ and is undefined whenever $i > 7$. If v is a vertex with $\alpha_G(v) = C^R_{\mathbb{A}}$, then the *recursive premise* of v, denoted $\rho_G(v)$, is the edge incident to v and coming from outside the box. In our example, $\rho_G(v)$ is e_{12}.

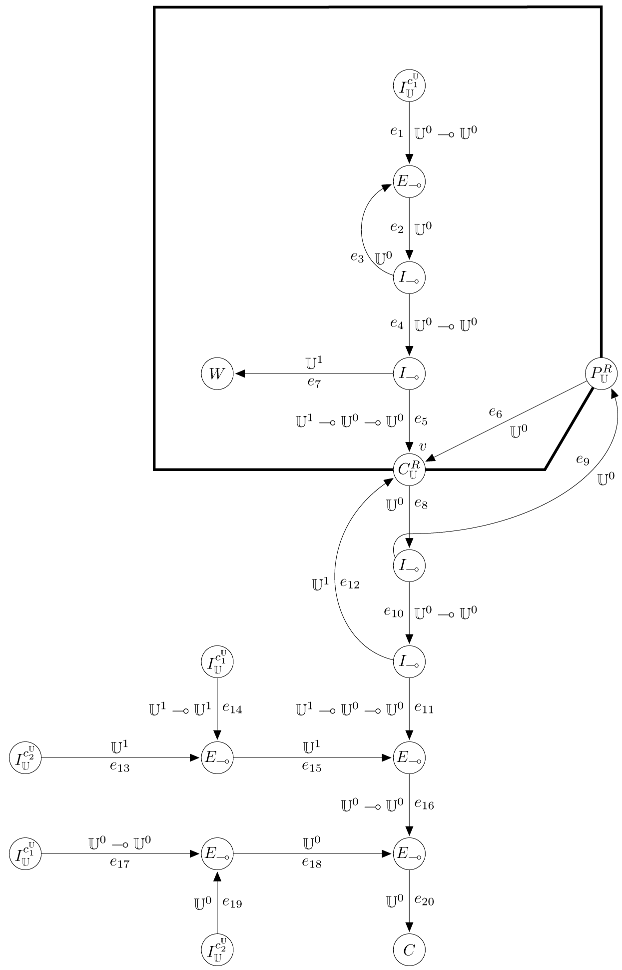


Fig. 5. The interaction graph corresponding to a type derivation for $\mathbf{UnAdd}\ulcorner 1\urcorner\ulcorner 1\urcorner$.

4.2 Algebraic Context Semantics as a Set of Trees

Defining algebraic context semantics requires a number of auxiliary concepts. In particular, we need to be able to discriminate between different occurrences of base types inside any type. Given a type A, a *focalization* B of A is an expression obtained from A by underlining an occurrence of a base type $\mathbb{A}$ inside A. If this is the case, we say that B *focuses on* $\mathbb{A}$. If the underlined occurrence of $\mathbb{A}$ in B occurs in positive (negative, respectively) position, B is said to be a *positive* (*negative*, respectively) focalization. For instance, the type $\mathbb{A} \multimap \mathbb{B} \multimap \mathbb{C}$ has the positive focalization $\mathbb{A} \multimap \mathbb{B} \multimap \underline{\mathbb{C}}$ and the negative focalizations $\underline{\mathbb{A}} \multimap \mathbb{B} \multimap \mathbb{C}$ and $\mathbb{A} \multimap \underline{\mathbb{B}} \multimap \mathbb{C}$. When B is a focalization of A, C is any type, and $1 \leq k \leq n$, the expression $B \overset{n,k}{\multimap} C$ denotes

$$\underbrace{A \multimap \ldots \multimap A}_{k-1 \text{ times}} \multimap B \multimap \underbrace{A \multimap \ldots \multimap A}_{n-k \text{ times}} \multimap C$$

which is a focalization of $A \overset{n}{\multimap} C$. $\mathscr{C}_{\mathscr{A}}$ denotes the class of all focalizations of types in $\mathscr{T}_{\mathscr{A}}$.

We need a similar notion of focalization for free algebra terms: Given a free algebra term t, a *focalization* s of t is an expression obtained from t by underlining one occurrence of a subterm of t. As previously, we say that s *focuses on* the underlined occurrence. As an example, the three focalizations of $c_2^{\mathbb{B}}c_1^{\mathbb{B}}c_3^{\mathbb{B}}$ are $\underline{c_2^{\mathbb{B}}c_1^{\mathbb{B}}c_3^{\mathbb{B}}}$, $c_2^{\mathbb{B}}\underline{c_1^{\mathbb{B}}c_3^{\mathbb{B}}}$, and $c_2^{\mathbb{B}}c_1^{\mathbb{B}}\underline{c_3^{\mathbb{B}}}$. Notice that any free algebra term t has exactly $|t|$ focalizations. When s is a focalization of t focusing on r, r is a term in the form $ct_1, \dots, t_n$, and $1 \leq k \leq n$, the expression $s \downarrow k$ denotes the (unique) focalization of t focusing on (the obviously defined occurrence of) t_k. For example $\underline{c_2^{\mathbb{B}}c_1^{\mathbb{B}}c_3^{\mathbb{B}}} \downarrow 1 = c_2^{\mathbb{B}}\underline{c_1^{\mathbb{B}}c_3^{\mathbb{B}}}$. If s is a focalization of t, $[s]$ stands for t.

We are now in a position to define configurations, which are tuples labeling nodes of trees in the context semantics. A *stack* for an interaction graph G is a (possibly empty) sequence of pairs (s, v), where s is a focalization of some free algebra term and v is a vertex of G labeled with $C_{\mathbb{A}}^{R}$. $C(G)$ denotes the set of stacks for G. A *configuration* for G is a quadruple (t, e, U, A) where:

—t is a free algebra term;

—e is an edge in E_G;

—U is a stack in $C(G)$; and

—A is a focalization of the type $\beta_G(e)$ labeling the edge e.

In the following, $S(G)$ denotes the set of configurations for G.

As we already mentioned, the context semantics of an interaction graph G is given by a set $T(G)$ of rooted, ordered trees, called *configuration trees*. Branches of configuration trees correspond to paths inside G, namely, finite sequences of consecutive edges of G. The path corresponding to a branch in G can be retrieved by considering the second component of tuples labeling vertices in the branch. The third and fourth components serve as contexts and are necessary to build the tree in a correct way. Indeed, this way of building trees by traversing paths is reminiscent of token machines in the context of game semantics and geometry of interaction [Danos et al. 1996; Danos and Regnier 1999]. Using this terminology, we can informally describe a configuration (t, e, U, A) as the current state of a token traversing an edge of an interaction graph. In particular, we have the following.

—The first component t is a value carried by the token; it can be modified when the token crosses a node labeled with $I_{\mathbb{A}}^{c}$.

—The token is traversing the second component e.

—If A is a positive focalization, then the token is moving in the same direction as the orientation of the edge e. If A is a negative focalization, then the token is moving in the opposite direction.

—The stack U keeps track of the boxes in which the token is currently located. Moreover, U gives us precise information about which *copy* of boxes the token is located into. For example, if $U = (s, v)(r, u)$, then the token is currently located at an edge at box-depth 2, inside the copy s of the box v, which in turn is inside the copy r of the box u.

Several tokens can possibly merge into a single one in correspondence to vertices of interaction graphs which are labeled with $I^c_{\mathbb{A}}$ (where the arity of c is strictly greater than 1). Therefore, the history of a token can be naturally described by a tree structure such as a configuration tree. With this intuition, the configuration $C = (t, e, U, A)$ labeling the root of any configuration tree T will be called the *current configuration* of T; likewise, t, e, and U will be the *current value*, the *current edge*, and the *current stack* of T, respectively. $L(T)$ is simply the current value of T.

Given an interaction graph G, the set $T(G)$ is defined as the smallest set of configuration trees for G satisfying the closure condition detailed next. In the following, we adopt the following conventions.

—An expression in the form

$$C \leftarrow C_1, \ldots, C_n$$

should be understood as the following closure condition on $T(G)$: If $T(G)$ contains configuration trees $T_1, \ldots, T_n$ with current configurations $C_1, \ldots, C_n$, then $T(G)$ also contains a tree T whose root is labeled with C and which has $T_1, \ldots, T_n$ as its immediate subtrees.

—We will often say a box v is *activated with* (t, U) *by* $T \in T(G)$ (or simply that the box is *activated with* (t, U) *in* $T(G)$), meaning that T has current configuration $(t, u, U, \underline{\mathbb{A}})$, where u is the recursive premise of v and $\mathbb{A} = \beta_G(u)$. Intuitively, a token activates a box v when it reaches the recursive premise of v; at that point, other tokens can be created inside the box, can enter or exit the box, and can move from one copy of the box to another.

—Whenever A is a type, the metavariable A^+ (respectively, A^-) will range over positive (respectively, negative) focalizations for A.

All closure conditions we are going to define will be in one of the following two forms:

—In the form $(t, e, U, A) \leftarrow (t, g, V, B)$. In other words, the root of the newly defined tree will have just one immediate descendant and the current value of T will be the same as the current value of its immediate descendant.

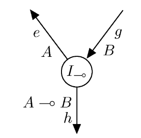
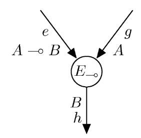
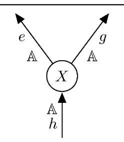
—In the form $(ct_1 \ldots t_n, e, U, A) \leftarrow (t_1, g_1, V_1, B_1), \ldots (t_n, g_n, V_n, B_n)$. In this case, the root of the newly defined tree T will have n immediate descendants and the current value of T will be built by applying a constructor to the current values of its immediate descendants.

As a consequence, if the current value of a configuration tree T in $T(G)$ is t and s is a focalization of t focusing on r, then there is at least one subtree S of T such that the current value of S is r. We will denote the smallest of such subtrees by $B(T, s)$. Observe the root of $B(T, s)$ is the last node we find when traveling from the root of T toward its leaves and being guided by s.

Formally, $T(G)$ is defined as the smallest set satisfying three families of closure conditions.

—Vertices of G with labels $I_{\multimap}$,$E_{\multimap}$, X induce closure conditions on $T(G)$. These conditions are detailed in Table I.

Table I. Closure Conditions

	$(t,e,U,A^+) \leftarrow (t,h,U,A^+ \multimap B)$; $(t,h,U,A^- \multimap B) \leftarrow (t,e,U,A^-)$; $(t,h,U,A \multimap B^+) \leftarrow (t,g,U,B^+)$; $(t,g,U,B^-) \leftarrow (t,h,U,A \multimap B^-)$.
	$(t,e,U,A^+ \multimap B) \leftarrow (t,g,U,A^+)$; $(t,g,U,A^-) \leftarrow (t,e,U,A^- \multimap B)$; $(t,e,U,A \multimap B^-) \leftarrow (t,h,U,B^-)$; $(t,h,U,B^+) \leftarrow (t,e,U,A \multimap B^+)$.
	$(t,e,U,\underline{\mathbb{A}}) \leftarrow (t,h,U,\underline{\mathbb{A}})$; $(t,g,U,\underline{\mathbb{A}}) \leftarrow (t,h,U,\underline{\mathbb{A}})$.

—Vertices of G with labels $I^c_{\mathbb{A}}$ and $C^N_{\mathbb{A}}$ induce slightly more complicated closure conditions on $T(G)$, as reported in Table II.

—Every vertex with labels $C^R_{\mathbb{A}}$ and $P^R_{\mathbb{A}}$ forces $T(G)$ to satisfy more complex closure conditions, as reported in Table III.

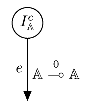
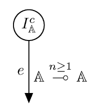
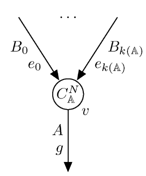
In Figure 6, we report two trees in $T(G_\sigma)$, where $\sigma : \mathbf{UnAdd}\ulcorner 1 \urcorner \ulcorner 1 \urcorner : \mathbb{U}^0$.

Some observations about the closure conditions in Tables I, II and III are now in order.

—The only way of proving a one-node tree to be in $T(G)$ consists in applying the closure condition induced by a vertex w labeled with $I^c_{\mathbb{A}}$, where $\mathcal{R}_{\mathbb{A}}(c) = 0$. Notice that if $\theta_G(w)$ is defined (i.e., w is inside a box), we must check the existence of another (potentially big) tree T. And similarly when we want to "enter" a box by traversing a vertex w labeled with $P^R_{\mathbb{A}}$.

—Closure conditions induced by vertices labeled with $C^R_{\mathbb{A}}$ are quite complicated. Consider one such vertex w. First of all, a preliminary condition to be checked is the existence of a tree T whose current edge is the recursive premise of w. The existence of T certifies that exactly $|L(T)|$ copies of the box under consideration will be produced during reduction, each of them corresponding to a tuple (s, w) where s is a focalization of $L(T)$. The vertex w induces five distinct closure rules. The first and second rules correspond to paths that enter or exit the box from its conclusion: A pair is either popped from the underlying stack (when exiting the box) or pushed into it (when entering the box). The third and fourth rules correspond to paths that come from the interior of the box under consideration and stay inside the same box: We go from one copy of the box to another one, and accordingly the leftmost element of the underlying stack is changed. The last rule is definitely the trickiest. First of all, remember that $L(T)$ represents an argument to the recursion corresponding to w. If we look at the reduction rule for recursive redexes, we immediately realize that subterms of this argument should be passed

Table II. Closure Conditions

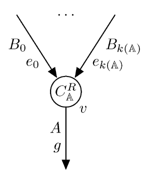
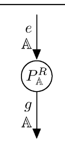
<table>
<tr>
<td></td>
<td>If e is not inside any box, then we can create a new token:
$$(c, e, \varepsilon, \underline{\mathbb{A}}) \leftarrow .$$
If e is immediately contained in a box v and v is activated with (t, U) in $T(G)$ then for any focalization s of t, we can create a new token:
$$(c, e, (s, v)U, \underline{\mathbb{A}}) \leftarrow .$$</td>
</tr>
<tr>
<td></td>
<td>$$\begin{aligned}(ct_1 \dots t_n, e, U, \mathbb{A} \overset{n}{\multimap} \underline{\mathbb{A}}) \;\leftarrow\;& (t_1, e, U, \underline{\mathbb{A}} \overset{1,n}{\multimap} \mathbb{A}),\\ & (t_2, e, U, \underline{\mathbb{A}} \overset{2,n}{\multimap} \mathbb{A}),\\ & \vdots\\ & (t_n, e, U, \underline{\mathbb{A}} \overset{n,n}{\multimap} \mathbb{A}).\end{aligned}$$</td>
</tr>
<tr>
<td>
$B_i \equiv \mathbb{A} \overset{n_i}{\multimap} A,\ i \geq 1$
$n_i = \mathcal{R}_{\mathbb{A}}(c_i^{\mathbb{A}}),\ i \geq 1$
$B_0 \equiv \mathbb{A}$</td>
<td>If the current configuration of $T \in T(G)$ is $(t, U, e_0, \underline{\mathbb{A}})$, then:
—If the head constructor of t is $c_i^{\mathbb{A}}$, then you can traverse v:
$$(r, g, U, A^+) \leftarrow (r, e_i, U, \mathbb{A} \overset{n_i}{\multimap} A^+);$$
$$(r, e_i, U, \mathbb{A} \overset{n_i}{\multimap} A^-) \leftarrow (r, g, U, A^-).$$
—For any focalization s of t with focus $p = c_i^{\mathbb{A}} t_1 \dots t_{n_i}$ and for any $1 \leq j \leq n_i$, let T_j^s be $B(T, s \downarrow k)$ and let C_j^s be the current configuration of T_j^s. Then:
$$(t_i, e_j, U, \underline{\mathbb{A}} \overset{n_i,j}{\multimap} A) \leftarrow C_j^s.$$</td>
</tr>
</table>

to the bodies of the recursion itself. Now, suppose we want to build a new tree in the context semantics by extending T itself. In other words, suppose we want to proceed with the paths corresponding to T. Intuitively, those paths should proceed inside the box. However, we cannot extend T itself, but subtrees of it. This is the reason why we extend T_j^s and not T itself in the last rule.

—Closure conditions induced by vertices labeled with $C_{\mathbb{A}}^N$ can be seen as slight simplifications on those induced by vertices labeled with $C_{\mathbb{A}}^R$. Here there are no boxes, we do not modify the underlying stack, and accordingly, there are no rules like the third and fourth rules induced by $C_{\mathbb{A}}^R$.

$T(G)$ has been defined as the smallest set satisfying certain closure conditions. This implies it will only contain *finite* trees. Moreover, it can be endowed with an induction principle which does not coincide with the trivial one. For example, the first of the two trees reported in Figure 6 is smaller (as an element of $T(G)$) than the second one, even if it is not a subtree of it. Saying it another way, proving properties about trees $T \in T(G)$ we can induce on the structure of the *proof* that T is an element of $T(G)$, rather than inducing on the structure of T as a tree. This induction principle turns out to be very powerful and will be extensively used in the following.

Table III. Closure Conditions

 $B_i \equiv \mathbb{A} \overset{n_i}{\multimap} A \overset{n_i}{\multimap} A,\ i \geq 1$ $n_i = \mathcal{R}_{\mathbb{A}}(c_i^{\mathbb{A}}),\ i \geq 1$ $B_0 \equiv \mathbb{A}$	Suppose that v is activated with (t, U) in $T(G)$. Then —If the head constructor of t is $c_i^{\mathbb{A}}$, then you can enter or exit the box v from then main door: $(r, e_i, (\underline{t}, v)U, \mathbb{A} \overset{n_i}{\multimap} (A \overset{n_i}{\multimap} A^-)) \leftarrow (r, g, U, A^-)$; $(r, g, U, A^+) \leftarrow (r, e_i, (\underline{t}, v)U, \mathbb{A} \overset{n_i}{\multimap} (A \overset{n_i}{\multimap} A^+))$. —For any focalization s of t with focus $p = c_i^{\mathbb{A}} t_1 \ldots t_{n_i}$ and for any $1 \leq k \leq n_i$, you can switch between the copy s and the copy $s \downarrow k$: $(r, e_j, (s \downarrow k, v)U, \mathbb{A} \overset{n_j}{\multimap} (A \overset{n_j}{\multimap} A^-)$ $\leftarrow (r, e_i, (s, v)U, \mathbb{A} \overset{n_i}{\multimap} (A^- \overset{n_i,k}{\multimap} A))$; $(r, e_i, (s, v)U, \mathbb{A} \overset{n_i}{\multimap} (A^+ \overset{n_i,k}{\multimap} A))$ $\leftarrow (r, e_j, (s \downarrow k, v)U, \mathbb{A} \overset{n_j}{\multimap} (A \overset{n_j}{\multimap} A^+)$. where the head constructor of t_k is $c_j^{\mathbb{A}}$ (and, as a consequence, the edge is e_j comes into play). —Suppose that $T \in T(G)$ activates v with (t, U). For any focalization s of t with focus $p = c_i t_1 \ldots t_{n_i}$ and for any $1 \leq j \leq n_i$, let T_j^s be $B(T, s \downarrow k)$ and let C_j^s be the current configuration of T_j^s. Then: $(t_j, e_i, (s, v)U, \underline{\mathbb{A}} \overset{n_i,j}{\multimap} (A \overset{n_i}{\multimap} A)) \leftarrow C_j^s$.
	If $\theta_G(g) = v$ and v is activated with (t, U) in $T(G)$, then for any focalization s of t, $(r, g, (s, v)U, \underline{\mathbb{A}}) \leftarrow (r, e, U, \underline{\mathbb{A}})$.

If $T \in T(G)$, we will denote by $U(T)$ the set containing all the elements of $C(G)$ which appear as second components of labels in T. The elements $U(T)$ are the *legal stacks* for T. Stacks in $U(T)$ have a very constrained structure. In particular, all vertices found (as third components of tuples) in a legal stack are labeled with $C^R_{\mathbb{A}}$ and are precisely the vertices of this type which lie at boundaries of boxes in which the current edge is contained. Moreover, if a focalization of t is found (as the first component of a tuple) in a legal stack, then there must be a certain tree $S \in T(G)$ such that $L(S) = t$. More precisely, we have the following lemma.

LEMMA 4.2 (LEGAL STACK STRUCTURE). *For every configuration tree $T \in T_G$ with current stack $(s_1, v_1) \ldots (s_k, v_k)$:*

(1) *For every $1 \leq i \leq k$ the box v_i is activated with $(t_i, (s_{i+1}, v_{i+1}) \ldots (s_k, v_k))$, where $t_i = [s_i]$;*
(2) *for every $1 \leq i < k$, the box v_i is immediately contained in v_{i+1};*
(3) *the box v_k is not contained in any other box.*

Moreover, $k = 0$ iff the current edge e of T is not inside any box. If $k \geq 1$, then e is contained in v_k.

$$
\begin{array}{c}
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{12}, \varepsilon, [\cdot]) \\
| \\
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{11}, \varepsilon, [\cdot] \multimap \mathbb{U}^0 \multimap \mathbb{U}^0) \\
| \\
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{15}, \varepsilon, [\cdot]) \\
| \\
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{14}, \varepsilon, \mathbb{U}^1 \multimap [\cdot]) \\
| \\
(c_2^{\mathbb{U}}, e_{14}, \varepsilon, [\cdot] \multimap \mathbb{U}^1) \\
| \\
(c_2^{\mathbb{U}}, e_{13}, \varepsilon, [\cdot])
\end{array}
$$

(a)

$$
\begin{array}{cc}
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_8, \varepsilon, [\cdot]) & \vdots \\
| & | \\
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_6, ([\cdot], c_1^{\mathbb{U}}c_2^{\mathbb{U}}, v), [\cdot]) & (c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{16}, \varepsilon, [\cdot] \multimap \mathbb{U}^0) \\
| & | \\
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_9, \varepsilon, [\cdot]) & (c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{18}, \varepsilon, [\cdot]) \\
| & | \\
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{10}, \varepsilon, [\cdot] \multimap \mathbb{U}^0) & (c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{17}, \varepsilon, \mathbb{U}^0 \multimap [\cdot]) \\
| & | \\
(c_1^{\mathbb{U}}c_2^{\mathbb{U}}, e_{11}, \varepsilon, \mathbb{U}^1 \multimap [\cdot] \multimap \mathbb{U}^0) & (c_2^{\mathbb{U}}, e_{17}, \varepsilon, [\cdot] \multimap \mathbb{U}^0) \\
| & | \\
\vdots & (c_2^{\mathbb{U}}, e_{19}, \varepsilon, [\cdot])
\end{array}
$$

(b)

Fig. 6. Examples of trees.

PROOF. By a straightforward induction on the proof that $T \in \mathcal{T}(G)$. □

4.3 Completeness

This section is devoted to proving the completeness of algebraic context semantics as a way to get the algebraic potential size of a term.

THEOREM 4.3 (COMPLETENESS). *If $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$, $M \rightsquigarrow^* N$ and t is the argument of a redex in N, then there is $T \in \mathcal{T}(G_\pi)$ such that $L(T) = t$.*

Two lemmas will suffice for proving Theorem 4.3. On one side, arguments of redexes inside a term M can be retrieved in the context semantics of M.

LEMMA 4.4 (ADEQUACY). *If $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$ and M contains a redex with argument t, then there is $T \in T(G_\pi)$ such that $L(T) = t$.*

PROOF. First of all, we can observe that there must be a subderivation ξ of π such that $\xi : \Delta \vdash t : \mathbb{A}$. Moreover, the path from the root of π to the root of ξ does not cross any instance of rule $E^R_{\mathbb{A}}$. We can prove that there is $e \in E_{G_\xi}$ such that $(t, e, \varepsilon, \underline{\mathbb{A}})$ is the current configuration of a tree in $T(G_\xi)$ by induction on the structure of ξ (with some effort if $\mathbb{A}$ is not a word algebra). The thesis follows once we observe that G_ξ is a subgraph of G_π, e always lies at the outermost level (i.e., outside any box), and g is undefined whenever g is part of the subgraph of G_π corresponding to G_ξ. □

This, however, does not suffice. Context semantics must also reflect arguments that will eventually appear during normalization.

LEMMA 4.5 (BACKWARD PRESERVATION). *If $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$ and $M \rightsquigarrow N$, there is $\xi : \Gamma \vdash N : A$ such that for every $T \in T(G_\xi)$, there is $S \in T(G_\pi)$ with $L(S) = L(T)$.*

PROOF. First of all we prove the following lemma: If $\pi : \Gamma, x : A \vdash M : B$ and $\xi : \Delta \vdash V : A$, then the interaction graph G_σ, where $\sigma : \Gamma, \Delta \vdash M\{V/x\} : B$ can be obtained by plugging G_ξ into the premise of G_π corresponding to x and applying one or more rewriting steps as those in Figure 7(a). This lemma can be proved by an induction on the structure of π.

Now, suppose $\pi : \Gamma \vdash M : A$ and $M \rightsquigarrow N$ by firing a beta redex. Then, a type derivation $\xi : \Gamma \vdash N : A$ can be obtained from π applying one rewriting step as that in Figure 7(b) and one or more rewriting steps as those in Figure 7(a). We can verify that, for every rewriting step in Figure 7, if H is obtained from G applying the rewriting step and $T \in T(H)$, then there is $S \in T(G)$ such that $L(S) = L(T)$.

We now prove the same for conditional and recursive redexes. To keep the proof simple, we assume to deal with conditionals and recursion on the algebra $\mathbb{U}$. Suppose $\pi : \Gamma \vdash M : A$ and $M \rightsquigarrow N$ by firing a recursive redex $c^{\mathbb{U}}_1 t \langle\!\langle M_1, M_2 \rangle\!\rangle$. Then there is a type derivation $\xi : \Gamma \vdash N : A$ such that G_ξ can be obtained from G_π by rewriting as in Figure 8(a). We can define a partial function

$$\varphi : E_{G_\xi} \times C(G_\xi) \times \mathscr{C}_{\mathscr{A}} \rightharpoonup E_{G_\pi} \times C(G_\pi) \times \mathscr{C}_{\mathscr{A}}$$

in such a way that if a tree in $T(G_\xi)$ has current configuration (t, e, U, B), then there is a tree in $T(G_\pi)$ with current configuration $(t, \varphi(e, U, B))$. In defining φ, we will take advantage of Lemma 4.2. For example, we can assume $U = \varepsilon$ whenever (r, e_2, U, B) appear as a label of any $T \in T(G_\xi)$. Indeed, we cannot fire any recursive redex "inside a box" because the reduction relation $\rightsquigarrow$ forbids it.

—The function φ acts as the identity on triples (e, U, B), where e lies outside the portion of G_ξ affected by rewriting.

—Observe there are two copies of $G(t)$ in G_ξ; if e is an edge of one of these two copies, then $\varphi(e, U, B)$ will be (g, U, B), where g is the edge corresponding to e in $G(c^{\mathbb{U}}_1 t)$.

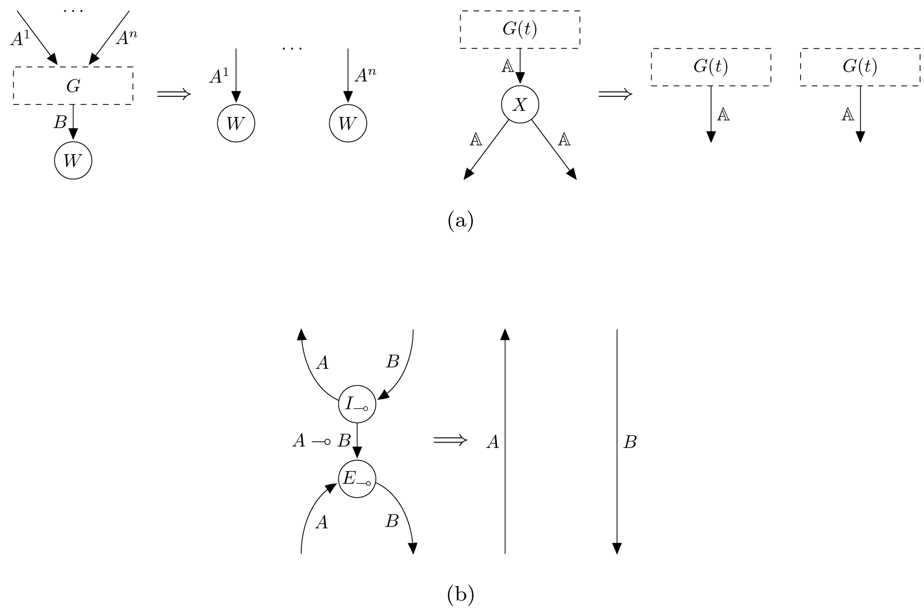


Fig. 7. Graph transformation produced by firing a beta redex.

—Observe there are two copies of $G(M_1)$ in G_ξ, the leftmost one inside a box w, and the rightmost outside it; if e is an edge of the rightmost of these two copies, then $\varphi(e, U, B)$ will be $(g, (\underline{c_1^{\mathbb{U}}t}, v)U, B)$, where g is the edge corresponding to e in $G(M_1)$; if e is an edge of the leftmost of these two copies, then $\varphi(e, (s, w)U, B)$ will be $(g, (c_1^{\mathbb{U}}s, v)U, B)$.

—In G_ξ there is just one copy of $G(M_2)$; if e is an edge of this copy of $G(M_2)$, then $\varphi(e, (s, w)U, L)$ will be $(g, (c_1^{\mathbb{U}}s, v)U, L)$.

—The following equations hold.

$$\begin{aligned}
\varphi(e_1^i, \varepsilon, B) &= \left(g_1^i, \varepsilon, B\right) \\
\varphi(e_2, \varepsilon, B) &= \left(g_2, (\underline{c_1^{\mathbb{U}}t}, v), \mathbb{U} \multimap B \multimap A\right) \\
\varphi(e_3, \varepsilon, A \multimap B) &= (g_3, \varepsilon, B) \\
\varphi(e_3, \varepsilon, B \multimap A) &= \left(g_2, \left(\underline{c_1^{\mathbb{U}}t}, v\right), \mathbb{U} \multimap B \multimap A\right)
\end{aligned}$$

We can prove that if $T \in T(G_\xi)$ has current configuration (r, e, ε, B), then there is a tree in $T(G_\pi)$ with current configuration $(r, \varphi(e, \varepsilon, B))$ by induction on T. Let us just analyze some of the most interesting cases.

—Suppose there is a tree $T \in T(G_\xi)$ with current configuration $(r, e_4^i, \varepsilon, \underline{\mathbb{A}})$. By applying the closure rule induced by vertices labeled with X, we can extend T to a tree with current configuration $(r, e_1^i, \varepsilon, \underline{\mathbb{A}})$. By the induction hypothesis applied to T, there is a tree in $T(G_\pi)$ with current configuration $(r, \varphi(e_4^i, \varepsilon, \underline{\mathbb{A}})) = (r, g_1^i, \varepsilon, \underline{\mathbb{A}})$. But observe that $\varphi(e_1^i, \varepsilon, \underline{\mathbb{A}}) = (g_1^i, \varepsilon, \underline{\mathbb{A}})$.

—Suppose there is a tree $T \in T(G_\xi)$ with current configuration $(r, e_4^i, \varepsilon, \underline{\mathbb{A}})$. By applying the closure rule induced by vertices labeled with X, we can extend

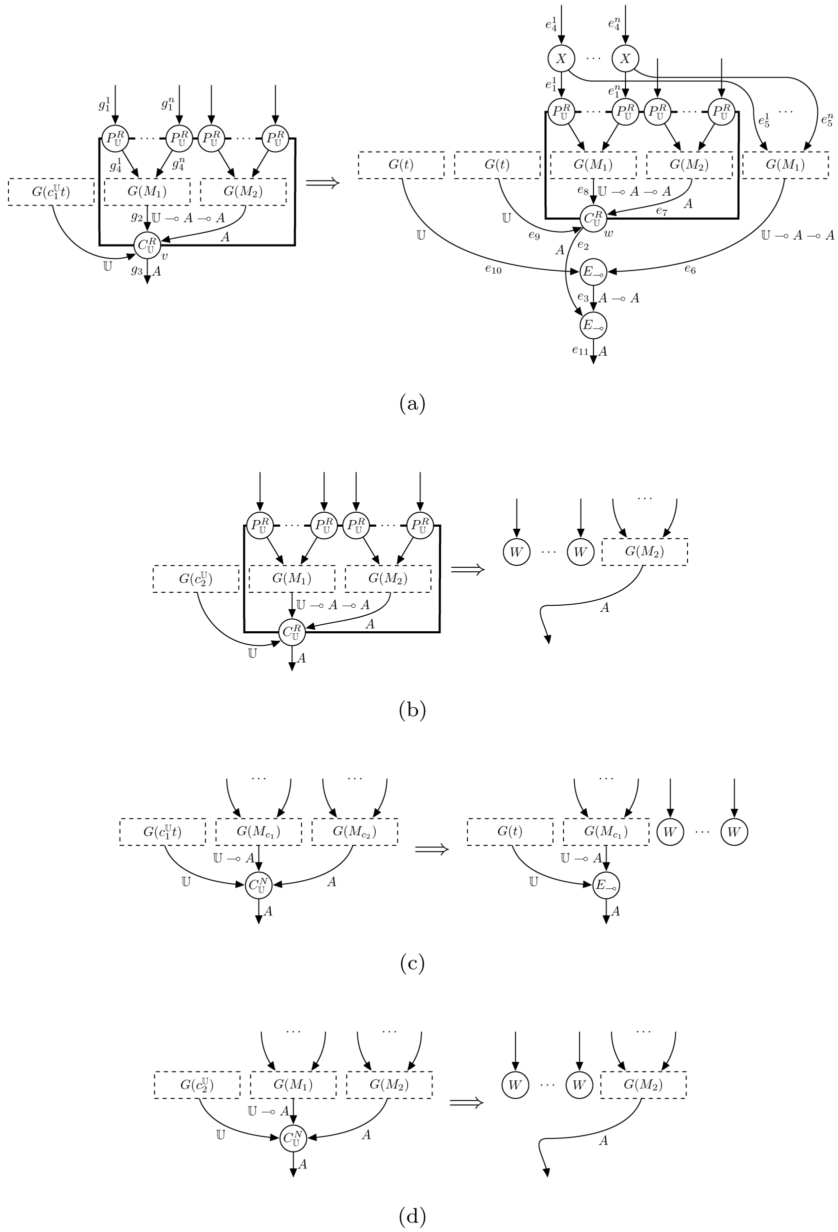


Fig. 8. The graph transformations induced by firing a recursive or conditional redex.

T to a tree in $T(G_\xi)$ with current configuration $(r, e_5^i, \varepsilon, \underline{\mathbb{A}})$. By the induction hypothesis applied to T, there is a tree $S \in T(G_\pi)$ with current configuration $(r, \varphi(e_4^i, \varepsilon, \underline{\mathbb{A}})) = (r, g_1^i, \varepsilon, \underline{\mathbb{A}})$. By applying the closure rule induced by vertices labeled with $P_{\mathbb{U}}^R$, we can extend S to a tree in $T(G_\pi)$ with current configuration $(r, g_4^i, (\underline{c_1^{\mathbb{U}} t}, v), \underline{\mathbb{A}})$ But observe that $\varphi(e_5^i, \varepsilon, \underline{\mathbb{A}}) = (g_4^i, (\underline{c_1^{\mathbb{U}} t}, v), \underline{\mathbb{A}})$, because e_5^i is part of the rightmost copy of $G(M_1)$.

—Suppose there is a tree $T \in T(G_\xi)$ whose root is labeled with $(r, e_{11}, \varepsilon, B)$, where B is a negative focalization of A. By applying the closure rule induced by vertices labeled with $E_{\multimap}$, we can extend T to a tree with current configuration $(r, e_3, \varepsilon, A \multimap B)$ and, by applying again the same closure rule, we can obtain a tree with current configuration $(r, e_6, \varepsilon, \mathbb{U} \multimap A \multimap B)$. By the induction hypothesis applied to T, there is a tree $S \in T(G_\pi)$ with current configuration $(r, \varphi(e_{11}, \varepsilon, B)) = (r, g_3, \varepsilon, B)$. Observe that $\varphi(e_3, \varepsilon, A \multimap B) = (g_3, \varepsilon, B)$. By applying the closure rule induced by vertices labeled with $C^R_{\mathbb{U}}$, we can extend S to a tree in $T(G_\pi)$ with current configuration $(r, g_2, (\underline{c^{\mathbb{U}}_1} t, v), \mathbb{U} \multimap A \multimap B)$. But observe that $\varphi(e_6, \varepsilon, \mathbb{U} \multimap A \multimap B) = (g_2, (\underline{c^{\mathbb{U}}_1} t, v), \mathbb{U} \multimap A \multimap B)$, because e_6 is part of the rightmost copy of $G(M_1)$.

—Suppose there is a tree $T \in T(G_\xi)$ whose root is labeled with $(r, e_6, \varepsilon, \mathbb{U} \multimap B \multimap A)$ and B is a negative focalization of A. By applying the closure rule induced by vertices labeled with $E_{\multimap}$, we can extend T to a tree with current configuration $(r, e_3, \varepsilon, B \multimap A)$ and, by applying another closure rule induced by the same vertex, we can obtain a tree with current configuration (r, e_2, ε, B). By the induction hypothesis applied to T, there is a tree $S \in T(G_\pi)$ with current configuration $(r, \varphi(e_6, \varepsilon, \mathbb{U} \multimap B \multimap A)) = (r, g_2, (\underline{c^{\mathbb{U}}_1} t, v), \mathbb{U} \multimap B \multimap A)$. Observe that $\varphi(e_3, \varepsilon, B \multimap A) = (g_2, (\underline{c^{\mathbb{U}}_1} t, v), \mathbb{U} \multimap B \multimap A)$ and $\varphi(e_2, \varepsilon, B) = (g_2, (\underline{c^{\mathbb{U}}_1} t, v), \mathbb{U} \multimap B \multimap A)$.

This shows that the thesis holds for recursive redexes in the form $c^{\mathbb{U}}_1 t \langle\langle M_1, M_2 \rangle\rangle$. Very similar arguments hold for redexes in the form $c^{\mathbb{U}}_2 \langle\langle M_1, M_2 \rangle\rangle$, $c^{\mathbb{U}}_1 t \{\!\{ M_1, M_2 \}\!\}$, $c^{\mathbb{U}}_2 \{\!\{ M_1, M_2 \}\!\}$ (see Figure 8(b), Figure 8(c) and Figure 8(d), respectively). □

Summing up, any possible algebraic term appearing in any possible reduct of a typeable term M can be found in the context semantics of the interaction graph for a type derivation for M. This proves Theorem 4.3.

5. ON THE COMPLEXITY OF NORMALIZATION

In this section, we will give some bounds on the time needed to normalize terms in subsystems H(A), RH(A), and RH(W). Our strategy consists in studying how constraints like linearity and ramification induce bounds on $|L(T)|$, where T is any tree built up from the context semantics. These bounds, by Theorem 4.3 and Proposition 4.1, translate into bounds on normalization time (modulo appropriate polynomials). Noticeably, many properties of the context semantics which are very useful in studying $|L(T)|$ are true for all of the aforesaid subsystems and can be proved just once. These are precisely the properties that will be proved in the first part of this section.

First of all we observe that, by definition, every subtree of $T \in T(G)$ is itself a tree in $T(G)$. Moreover, a uniqueness property can be proved.

PROPOSITION 5.1 (UNIQUENESS). *For every interaction graph G, for every $e \in E_G$, $U \in C(G)$, and $L \in \mathscr{C}_{\mathscr{A}}$, there is at most one tree $T \in T(G)$ such that (t, e, U, L) is the current configuration of T.*

PROOF. We can show the following: If $T, S \in T(G)$ have current configurations (s, e, U, A) and (t, e, U, A) (respectively), then $T = S$ (and, as a consequence, $s = t$). We can prove this by an induction on the structure of the proof that $T \in T(G)$. First of all, observe that A and e uniquely determine the last closure rule used to prove that $T \in T(G)$. In particular, if $e = (v, w)$ and A is positive, then it is one induced by v, otherwise it is one induced by w. At this point, however, it can easily be seen that the number of children of the roots of T and S must be the same. If $S \neq T$, then there must be a child of T and a corresponding child of S which are different, but with corresponding labels. This, however, would contradict the inductive hypothesis. □

The previous result implies the following: Every triple $(e, U, L) \in E_G \times C(G) \times \mathscr{C}_{\mathscr{A}}$ can appear at most once in any branch of any $T \in T(G)$. As a consequence, any $T \in T(G)$ (and more importantly, any t such that $t = L(T)$ for some $T \in T(G)$) cannot be too big compared to $|C(G)|$ and $|G|$. But, in turn, the structure of relevant elements of $C(G)$ is very contrived.

Indeed, Lemma 4.2 implies the length of any stack in $U(T)$, where $T \in T(G_\pi)$ cannot be bigger than the recursion depth $R(\pi)$ of π: The length of U is equal to the "box-depth" of e whenever (t, e, U, A) appears as a label in T.

Along a path, the fourth component of the underlying tuple can change, but there is something which stays invariant.

LEMMA 5.2. *For every $T \in T(G)$ there is a type $\mathbb{A}^i$ such that for every (t, e, U, A) appearing as a label of a vertex of T, A is a focalization of $\mathbb{A}^i$. We will say T is* guided *by $\mathbb{A}^i$.*

PROOF. By a straightforward induction on the proof that $T \in T(G)$. □

The previous lemmas shed some light on the combinatorial properties of tuples $(t, e, U, A) \in S(G_\pi)$ labeling vertices of trees in $T(G_\pi)$. This is enough to prove $|L(T)|$ to be exponentially related to the cardinality of $U(T)$.

PROPOSITION 5.3. *Suppose $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$ and $T \in T(G_\pi)$. Then $|L(T)| \leq \mathscr{K}_{\mathscr{A}}^{|G_\pi||U(T)|}$.*

PROOF. First of all, we observe that whenever (t, e, U, A) labels a vertex v of T and (s, f, V, B) labels one child of v, then either $s = t$ or $t = ct_1 \ldots t_k$, $t_i = s$ and $A = \mathbb{A} \overset{k}{\multimap} \underline{\mathbb{A}}$. The thesis follows from Lemma 5.2 and Proposition 5.1. □

Proposition 5.3 will lead to primitive recursive bounds for $\mathsf{H(A)}$ and elementary bounds for $\mathsf{RH(A)}$. However, we cannot expect to prove any polynomial bound from it. In the case of $\mathsf{RH(W)}$, a stronger version of Proposition 5.3 can be proved by exploiting ramification.

PROPOSITION 5.4. *Suppose $\pi : \Gamma \vdash_{\mathsf{RH(W)}} M : A$ and $T \in T(G_\pi)$. Then $|L(T)| \leq |G_\pi||U(T)|$.*

PROOF. First of all, we prove the following lemma: For every $T \in T(G_\pi)$, if T is guided by $\mathbb{A}^i$ and $\mathbb{A}$ is a word algebra, there are at most one tree $S \in T(G_\pi)$ and one integer $j \in \mathbb{N}$ such that S has T as its jth child. To prove the lemma, suppose $S, R \in T(G_\pi)$ such that T is the jth child of S and the kth child of

R. T uniquely determines the closure condition used to prove that both S and R are in $T(G_\pi)$, which must be the same because those induced by typing rule X are forbidden. But by inspecting all the closure rules, we can conclude that $S = R$ and $j = k$. Then, we can proceed exactly as in Proposition 5.3. □

Notice how the elementary bound of Proposition 5.3 has become a polynomial bound in Proposition 5.4. Quite surprisingly, this phase transition happens as soon as the class of types on which we allow contraction is restricted from A to W.

5.1 H(A) and Primitive Recursion

Given an interaction graph G, we now need to define subclasses $T(\mathcal{U})$ of $T(G)$ for any subset $\mathcal{U}$ of $C(G)$. In principle, we would like $U(T)$ to be a subset of $\mathcal{U}$ whenever $T \in T(\mathcal{U})$. However, this is too strong a constraint, since we should allow $U(T)$ to contain extensions of stacks in $\mathcal{U}$, the extensions being obtained themselves in this constrained way. The following definition captures the previous intuition. Let G be an interaction graph and $\mathcal{U} \subseteq C(G)$. A tree $T \in T(G)$ is said to be *generated* by $\mathcal{U}$ iff for every $U \in U(T)$:

—either $U \in \mathcal{U}$,

—or $U = (s_1, v_1)\dots(s_k, v_k)V$, where V is itself in $\mathcal{U}$ and has maximal length (between all the elements of $\mathcal{U}$). Moreover, for every $i \in \{1, \dots, k\}$, v_i is activated with $([s_i], (s_{i+1}, v_{i+1})\dots(s_k, v_k)V)$ by a tree which is itself generated by $\mathcal{U}$.

The set of all trees generated by $\mathcal{U}$ will be denoted by $T(\mathcal{U})$. This definition is well posed because of the induction principles on $T(G)$. Indeed, we require some trees $T_1, \dots, T_n$ to be in $T(\mathcal{U})$ when defining conditions on T being an element of $T(\mathcal{U})$ itself; however, $T_1, \dots, T_n$ are "smaller" than T. Notice that T is not monotone as an operator on subsets of $C(G)$. For example, $T(\{\varepsilon\}) = T(G)$, while $T(\{\varepsilon, C\}) \subset T(G)$ whenever $C \notin U(T)$ for any $T \in T(G)$. This is due to the requirement of V having maximal length in the preceding definition.

Lemma 5.5. *For every $d \in \mathbb{N}$ there is a primitive recursive function $p_d : \mathbb{N}^2 \to \mathbb{N}$ such that if $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$, $\mathcal{U} \subseteq C(G_\pi)$, the maximal length of elements of $\mathcal{U}$ is n, and $T \in T(\mathcal{U})$, then $|L(T)| \leq p_{R(\pi)-n}(|G_\pi|, |\mathcal{U}|)$.*

Proof. We can put

$$\begin{aligned} p_0(x, y) &= \mathscr{K}_{\mathscr{A}}^{xy} \\ \forall i \geq 1. h_i(x, y, 0) &= \mathscr{K}_{\mathscr{A}}^{xy} \\ \forall i \geq 1. h_i(x, y, z+1) &= h_i(x, y, z) + p_{i-1}(x, y + h_i(x, y, z)) \\ \forall i \geq 1. p_i(x, y) &= h_i(x, y, xy) \end{aligned}$$

where every p_i and h_i are primitive recursive. Moreover, all these functions are monotone in each of their arguments. We will now prove the thesis by induction on $R(\pi) - n$. If $R(\pi) = n$, then there are elements in $\mathcal{U}$ having length equals to $R(\pi)$. This, by Lemma 4.2 and Proposition 5.3, implies that if $T \in T(G)$ is generated by $\mathcal{U}$, then $|L(T)|$ is bounded by $p_0(|G|, |\mathcal{U}|)$ since none of the elements of $\mathcal{U}$ having maximal length can be extended into an element of $U(T)$ and, as a consequence, $U(T) \subseteq \mathcal{U}$. Now, let us suppose $R(\pi) - n \geq 1$. Let us define $\mathcal{W} \subseteq C(G)$ as follows.

$$\mathcal{W} = \{(s, v)U \mid U \in \mathcal{U} \text{ has maximal length} \\ \text{and } v \text{ is activated with } ([s], U) \text{ by a tree in } T(\mathcal{U})\}.$$

Clearly, $T(\mathcal{U} \cup \mathcal{W}) = T(\mathcal{U})$. Now, consider the sequence $(v_1, U_1), \ldots, (v_k, U_k)$ of all the pairs $(v_i, U_i) \in V_G \times \mathcal{U}$ such that $(s, v_i)U_i \in \mathcal{W}$ for some s. Obviously, $k \leq |G||\mathcal{U}|$. If $k = 0$, then the thesis is trivial, since

$$\begin{aligned} |L(T)| &\leq \mathscr{K}_{\mathscr{A}}^{|G||\mathcal{U}|} \\ &= h_{R(\pi)-n}(|G|, |\mathcal{U}|, 0) \\ &\leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, |G||\mathcal{U}|) \\ &= p_{R(\pi)-n}(|G|, |\mathcal{U}|). \end{aligned}$$

From now on, suppose $k \geq 1$. Let $\mathcal{W}_1, \ldots, \mathcal{W}_k \subseteq \mathcal{W}$ be defined as follows: $\mathcal{W}_i = \{(s, v_j)U_j \in \mathcal{W} \mid j \leq i\}$. By definition, $\mathcal{W}_k = \mathcal{W}$. We can assume, without losing generality, that:

(1) v_1 is activated with (t_1, U_1) by a tree T_1 only containing elements from $\mathcal{U}$ as part of its labels.
(2) For every $i \in \{2, \ldots, k\}$, v_i is activated with (t_i, U_i) by a tree T_i generated by $\mathcal{U} \cup \mathcal{W}_{i-1}$.

We can now prove that

$$\sum_{j=1}^{i+1} |t_j| \leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, i)$$

by induction on i. The tree T_1 only contains elements of $\mathcal{U}$ as part of its labels and, by Proposition 5.3,

$$|t_1| \leq \mathscr{K}_{\mathscr{A}}^{|G||U(T_1)|} \leq \mathscr{K}_{\mathscr{A}}^{|G||\mathcal{U}|} = h_{R(\pi)-n}(|G|, |\mathcal{U}|, 0).$$

If $i \geq 1$, by inductive hypothesis (on i) we get

$$\sum_{j=1}^{i} |t_j| \leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, i-1).$$

This yields $|\mathcal{W}_i| \leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, i-1)$, because for every every term t_j there at most $|t_j|$ couples (s, v_j) such that s is a focalization of t_j. By induction hypothesis (both on i and $R(\pi) - n$), we get

$$\begin{aligned} \sum_{j=1}^{i+1} |t_j| &= \sum_{j=1}^{i} |t_j| + |t_{i+1}| \\ &\leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, i-1) + p_{R(\pi)-n-1}(|G|, |\mathcal{U}| + h_{R(\pi)-n}(|G|, |\mathcal{U}|, i-1)) \\ &= h_{R(\pi)-n}(|G|, |\mathcal{U}|, i), \end{aligned}$$

because T_{i+1} is generated by $\mathcal{U} \cup \mathcal{W}_i$. So, $|\mathcal{W}| = |\mathcal{W}_k| \leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, k-1)$. Now, suppose $T \in T(\mathcal{U}) = T(\mathcal{U} \cup \mathcal{W})$. Then by inductive hypothesis (on $R(\pi) - n$), we have

$$
\begin{aligned}
|L(T)| &\leq p_{R(\pi)-n-1}(|G|, |\mathcal{U} \cup \mathcal{W}|) \\
&= p_{R(\pi)-n-1}(|G|, |\mathcal{U}| + |\mathcal{W}|) \\
&= p_{R(\pi)-n-1}(|G|, |\mathcal{U}| + |\mathcal{W}_k|) \\
&\leq p_{R(\pi)-n-1}(|G|, |\mathcal{U}| + h_{R(\pi)-n}(|G|, |\mathcal{U}|, k-1)) \\
&\leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, k) \\
&\leq h_{R(\pi)-n}(|G|, |\mathcal{U}|, |G||\mathcal{U}|) \\
&= p_{R(\pi)-n}(|G|, |\mathcal{U}|).
\end{aligned}
$$

This concludes the proof. □

As a corollary, we get the next theorem.

THEOREM 5.6. *For every $d \in \mathbb{N}$, there is a primitive recursive function p_d : $\mathbb{N} \to \mathbb{N}$ such that for every type derivation $\pi : \Gamma \vdash_{\mathsf{H(A)}} M : A$, if $T \in T(G_\pi)$ then $|L(T)| \leq p_{R(\pi)}(|M|)$.*

PROOF. Trivial, since every tree $T \in T(G_\pi)$ is generated by $\{\varepsilon\}$. □

Theorem 5.6 implies, by Proposition 4.1, that the time needed to normalize a term M with a type derivation π in H(A) is bounded by a primitive recursive function (just depending on the recursion depth of π) applied to the size of M. This, in particular, implies that every function $f : \mathbb{N} \to \mathbb{N}$ which can be represented in H(A) must be primitive recursive, because all terms corresponding to calls to f can be typed with bounded-recursion-depth type derivations. This is a *leitmotif*: Elementary bounds for RH(A) and polynomial bounds for RH(W) will have the same flavor. This way of formulating soundness results is one of the strongest ones. Indeed, since bounds are given on the normalization time of *any* term in the subsystem and the subsystem itself is complete for a complexity class, we cannot hope to prove, say, that *every* term in H(A) can be normalized with a *fixed*, primitive recursive, bound on its size.

5.2 RH(A) and Elementary Time

Consider the interpretation of branches of trees in $T(G)$ as paths in G: Any such path can enter and exit boxes by traversing vertices labeled with $C^R_{\mathbb{A}}$ or $P^R_{\mathbb{A}}$. The stack U in the underlying context can change as a result of the traversal. Indeed, U changes only when entering and exiting boxes (other vertices of G leave U unchanged, as can be easily verified). As a consequence, by Proposition 5.3, entering and exiting boxes is essential to obtain a hyperexponential complexity: If paths induced by a tree $T \in T(G)$ do not enter or exit boxes, $U(T)$ will be a singleton and $L(T)$ will be bounded by a fixed exponential on $|G|$. In general, paths induced by trees can indeed enter or exit boxes. If ramification holds, on the other hand, a path induced by a tree guided by $\mathbb{A}^i$ entering into a box whose main premise is labeled by $\mathbb{F}^j$ (where $j \leq i$), will stay inside the box and the third component of the underlying context will only increase in size. More formally, we have the next lemma.

LEMMA 5.7. *Suppose π to be a type derivation satisfying the ramification condition, $S \in T(G_\pi)$ to be guided by $\mathbb{A}^i$, (t, e, U, A) to label a vertex v of S and $U = (s, w)V$, where $\beta_G(\rho_G(w)) = \mathbb{F}^j$ and $j \leq i$. Then all the ancestors of v in S are labeled with quadruples (u, f, W, B), where $W = ZU$.*

PROOF. By a straightforward induction on the structure of S. In particular, the only vertices in G_π whose closure conditions affect the third component of $C(G)$ are those labeled with $P^R_{\mathbb{G}}$ and $C^R_{\mathbb{G}}$, where $\mathbb{G}$ is any free algebra. The rule induced by a vertex $P^R_{\mathbb{G}}$, however, makes the underlying stack bigger (from U, it becomes $(s, v)U$). As a consequence, the statement of the lemma is verified. Now, consider rules induced by $C^R_{\mathbb{G}}$ vertices.

— If one of the first four rules can be applied, then the current stack of the tree we are extending must be ZU, where $Z \neq \varepsilon$, since otherwise the ramification conditions would not be satisfied. As a consequence, the thesis holds.
— The fifth rule is a bit delicate: T^s_j satisfies the lemma, being that it is a subtree of a tree T to which we can apply the inductive hypothesis. The rule appends to T^s_j a configuration whose third component is $(s, v)U$, where U is the current stack of T. The thesis clearly holds.

This concludes the proof. □

This in turn allows to prove a theorem bounding the algebraic potential size of terms in system $\mathsf{RH(A)}$.

THEOREM 5.8. *For every $d, e \in \mathbb{N}$, there are elementary functions $p^d_e : \mathbb{N} \to \mathbb{N}$ such that for every type derivation $\pi : \Gamma \vdash_{\mathsf{RH(A)}} M : A$, if $T \in T(G_\pi)$ then $|L(T)| \leq p^{I(\pi)}_{R(\pi)}(|M|)$.*

PROOF. Consider the elementary functions

$$\begin{aligned}\forall n, m \in \mathbb{N}. p^n_m &: \mathbb{N} \to \mathbb{N};\\ p^0_m(x) &= \mathscr{K}^{x^2}_{\mathscr{A}};\\ p^{n+1}_m(x) &= \mathscr{K}^{x(x \cdot p^n_m(x))^m}_{\mathscr{A}}.\end{aligned}$$

First of all, notice that for every x, m, n, $p^{n+1}_m(x) \geq p^n_m(x)$. We will prove that if $T \in T(G_\pi)$ is guided by $\mathbb{A}^i$, then $|L(T)| \leq p^j_{R(\pi)}(|G_\pi|)$, where $j = \max\{I(\pi) - i, 0\}$. We go by induction on j.

If $j = 0$, then $I(\pi) \leq i$. This implies that $|U(T)| \leq |G_\pi|$, by Lemmas 4.2 and 5.7. Indeed, by Lemma 5.7, stacks can only get bigger along paths induced by T and any vertex in G uniquely determines the length of stacks (Lemma 4.2). As a consequence, $|L(T)| \leq \mathscr{K}^{|G_\pi|^2}_{\mathscr{A}} = p^0_{R(\pi)}(|G_\pi|)$.

Now, suppose the thesis holds for j and suppose T to be guided by $\mathbb{A}^i$, where $I(\pi) - i = j + 1$. By Lemma 5.7 and the induction hypothesis,

$$|U(T)| \leq (|G_\pi| p^j_{R(\pi)}(|G_\pi|))^{R(\pi)}.$$

Indeed, elements of $U(T)$ are stacks in the form $(s_1, v_1) \cdots (s_k, v_k)$ where $k \leq R(\pi)$, and for every $l \in \{1, \ldots, k\}$:

—either $\beta_G(\rho_G(v_l)) = \mathbb{F}^h$, where $h \leq i$ and T uniquely determines s_l due to Lemma 5.7;

—or $\beta_G(\rho_G(v_l)) = \mathbb{F}^h$ where $h > i$, $(s_{l+1}, v_{l+1}) \cdots (s_k, v_k)$ and v_l uniquely determine $u = [s_l]$ and $|u| \leq p^j_{R(\pi)}(|G_\pi|)$ by the inductive hypothesis.

As a consequence,

$$|L(T)| \leq \mathscr{K}_{\mathscr{A}}^{|G_\pi|(|G_\pi|p^j_{R(\pi)}(|G_\pi|))^{R(\pi)}} \leq p^{j+1}_{R(\pi)}(|G_\pi|).$$

The thesis follows by observing that $j \leq I(\pi)$. □

This implies that every function which can be represented inside RH(A) is elementary-time computable.

5.3 RH(W) and Polynomial Time

Notice that the exponential bound of Proposition 5.3 has become a polynomial bound in Proposition 5.4. Since Proposition 5.3 has been the essential ingredient in proving the elementary bounds of Section 5.2, polynomial bounds are to be expected for RH(W). Indeed, we have the following.

THEOREM 5.9. *For every $d, e \in \mathbb{N}$, there are polynomials $p^d_e : \mathbb{N} \to \mathbb{N}$ such that for every type derivation $\pi : \Gamma \vdash_{\mathsf{RH(W)}} M : A$, if $T \in T(G_\pi)$ then $|L(T)| \leq p^{I(\pi)}_{R(\pi)}(|M|)$.*

PROOF. We can proceed very similarly to the proof of Theorem 5.8. Consider the following polynomials.

$$\begin{aligned} \forall n, m \in \mathbb{N}. p^n_m &: \mathbb{N} \to \mathbb{N} \\ p^0_m(x) &= x^2 \\ p^{n+1}_m(x) &= x(x \cdot p^n_m(x))^m \end{aligned}$$

For every x, m, n, $p^{n+1}_m(x) \geq p^n_m(x)$. We will prove that if $T \in T(G_\pi)$ is generated by $\mathbb{A}^i$, then $|L(T)| \leq p^j_{R(\pi)}(|G_\pi|)$, where $j = \max\{I(\pi) - i, 0\}$. We go by induction on j.

If $j = 0$, then $I(\pi) \leq i$. This implies that $|U(T)| \leq |G_\pi|$ by Lemmas 4.2 and 5.7, similarly to Theorem 5.8. As a consequence of Proposition 5.4, $|t| \leq |G_\pi|^2 = p^0_{R(\pi)}(|G_\pi|)$.

Now, suppose the thesis holds for j and suppose T to be guided by $\mathbb{A}^i$, where $R(\pi) - i = j + 1$. By Lemma 5.7 and the induction hypothesis,

$$|U(T)| \leq (|G_\pi|p^j_{R(\pi)}(|G_\pi|))^{R(\pi)},$$

similarly to Theorem 5.8. As a consequence of Proposition 5.4,

$$|L(T)| \leq |G_\pi|(|G_\pi|p^j_{R(\pi)}(|G_\pi|))^{R(\pi)} \leq p^{j+1}_{R(\pi)}(|G_\pi|)).$$

The thesis follows by observing that $j \leq I(\pi)$. □

6. EMBEDDING COMPLEXITY CLASSES

In this section, we will provide embeddings of **FR** into H(∅), **FE** into RH(A), and **FP** into RH(∅). This will complete the picture sketched in Section 3. First of all, we can prove that a weaker notion of contraction can be retrieved in $\mathsf{RH}(\mathsf{D})$, even if $\mathsf{D} = \emptyset$.

LEMMA 6.1. *For every term M, there is a term $[M]^w_{x,y}$ such that for every $t \in \mathscr{E}_{\mathbb{U}}$, $([M]^w_{x,y})\{t/w\} \rightsquigarrow^* M\{t/x, t/y\}$. For every $n \in \mathbb{N}$, if $x : \mathbb{U}^n, y : \mathbb{U}^n \vdash_{\mathsf{H}(\emptyset)} M : A$ then $w : \mathbb{U}^n \vdash_{\mathsf{H}(\emptyset)} [M]^w_{x,y} : A$ and if $x : \mathbb{U}^n, y : \mathbb{U}^n \vdash_{\mathsf{RH}(\emptyset)} M : A$ then $w : \mathbb{U}^{n+2} \vdash_{\mathsf{RH}(\emptyset)} [M]^w_{x,y} : A$.*

PROOF. Given a term $t \in \mathscr{E}_{\mathbb{U}}$, the term $\overline{t} \in \mathscr{E}_{\mathbb{C}}$ is defined as follows, by induction on t.

$$\begin{aligned} \overline{c_2^{\mathbb{U}}} &= c_2^{\mathbb{C}}; \\ \overline{c_1^{\mathbb{U}} t} &= c_1^{\mathbb{C}} \overline{t} c_2^{\mathbb{C}} \end{aligned}$$

We can define two closed terms **Extract**, **Duplicate** $\in \mathscr{M}_{\mathscr{A}}$ such that, for every $t \in \mathscr{E}_{\mathbb{U}}$

$$\begin{aligned} \textbf{Extract}\ \overline{t} &\rightsquigarrow^* t; \\ \textbf{Duplicate}\ t &\rightsquigarrow^* c_2^{\mathbb{C}} \overline{t}\ \overline{t}. \end{aligned}$$

The terms we are looking for are

$$\begin{aligned} \textbf{Extract} &\equiv \lambda x.x\langle\!\langle \lambda y.\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}} z, c_2^{\mathbb{U}} \rangle\!\rangle; \\ \textbf{Duplicate} &\equiv \lambda x.x\langle\!\langle M_1, M_2 \rangle\!\rangle; \end{aligned}$$

where

$$\begin{aligned} M_1 &\equiv \lambda y.\lambda w.w\{\!\{\lambda z.\lambda q.c_1^{\mathbb{C}}(c_1^{\mathbb{C}} z c_2^{\mathbb{C}})(c_1^{\mathbb{C}} q c_2^{\mathbb{C}}), c_2^{\mathbb{C}}\}\!\}; \\ M_2 &\equiv c_1^{\mathbb{C}} c_2^{\mathbb{C}} c_2^{\mathbb{C}}. \end{aligned}$$

Indeed:

$$\begin{aligned} c_2^{\mathbb{C}}\langle\!\langle \lambda y.\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}} z, c_2^{\mathbb{U}} \rangle\!\rangle &\rightsquigarrow c_2^{\mathbb{U}}; \\ c_1^{\mathbb{C}} \overline{t} c_2^{\mathbb{C}}\langle\!\langle \lambda y.\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}} z, c_2^{\mathbb{U}} \rangle\!\rangle &\rightsquigarrow^* \big(\lambda y.\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}} z\big)\ \overline{t}\ c_2^{\mathbb{C}} \\ &\qquad \big(\overline{t}\langle\!\langle \lambda y.\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}} z, c_2^{\mathbb{U}} \rangle\!\rangle\big)\ c_2^{\mathbb{U}} \\ &\rightsquigarrow^* \big(\lambda y.\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}} z\big)\ \overline{t}\ c_2^{\mathbb{C}}\ t\ c_2^{\mathbb{U}} \\ &\rightsquigarrow^* c_1^{\mathbb{U}} t; \\ \textbf{Extract}\ \overline{t} &\rightsquigarrow \overline{t}\langle\!\langle \lambda y.\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}} z, c_2^{\mathbb{U}} \rangle\!\rangle \\ &\rightsquigarrow^* t; \\ c_2^{\mathbb{U}}\langle\!\langle M_1, M_2 \rangle\!\rangle &\rightsquigarrow^* c_1^{\mathbb{C}} c_2^{\mathbb{C}} c_2^{\mathbb{C}}; \\ c_1^{\mathbb{U}}\ t\langle\!\langle M_1, M_2 \rangle\!\rangle &\rightsquigarrow^* c_1^{\mathbb{C}} \overline{t}\ \overline{t}\{\!\{\lambda z.\lambda q.c_1^{\mathbb{C}}\big(c_1^{\mathbb{C}} z c_2^{\mathbb{C}}\big)\big(c_1^{\mathbb{C}} q c_2^{\mathbb{C}}\big), c_2^{\mathbb{C}}\}\!\} \\ &\rightsquigarrow^* c_1^{\mathbb{C}}\big(c_1^{\mathbb{C}} \overline{t} c_2^{\mathbb{C}}\big)\big(c_1^{\mathbb{C}} \overline{t} c_2^{\mathbb{C}}\big); \\ \textbf{Duplicate}\ t &\rightsquigarrow t\langle\!\langle M_1, M_2 \rangle\!\rangle \\ &\rightsquigarrow^* c_1^{\mathbb{C}} \overline{t}\ \overline{t}. \end{aligned}$$

Observe that, for every natural number n,

$$\begin{aligned}\vdash_{\mathsf{H}(\emptyset)}: \textbf{Extract} &: \mathbb{C}^n \multimap \mathbb{U}^n;\\ \vdash_{\mathsf{H}(\emptyset)}: \textbf{Duplicate} &: \mathbb{U}^n \multimap \mathbb{C}^n;\\ \vdash_{\mathsf{RH}(\emptyset)}: \textbf{Extract} &: \mathbb{C}^{n+1} \multimap \mathbb{U}^n;\\ \vdash_{\mathsf{RH}(\emptyset)}: \textbf{Duplicate} &: \mathbb{U}^{n+1} \multimap \mathbb{C}^n.\end{aligned}$$

Now let us define

$$[M]^w_{x,y} \equiv (\textbf{Duplicate}\ w)\{\!\{\lambda z.\lambda q.(\lambda x.\lambda y.M)(\textbf{Extract}\ z)(\textbf{Extract}\ q), \lambda x.\lambda y.M\}\!\}.$$

Indeed, for every $t \in \mathscr{E}_{\mathbb{U}}$:

$$\begin{aligned}[M]^w_{x,y}\{t/w\} &\rightsquigarrow^* (c_1^{\mathbb{C}}\overline{t}\overline{t})\{\!\{\lambda z.\lambda q.(\lambda x.\lambda y.M)(\textbf{Extract}\ z)(\textbf{Extract}\ q), \lambda x.\lambda y.M\}\!\}\\ &\rightsquigarrow^* (\lambda x.\lambda y.M)(\textbf{Extract}\ \overline{t})(\textbf{Extract}\ \overline{t})\\ &\rightsquigarrow^* (\lambda x.\lambda y.M)\ t\ t\\ &\rightsquigarrow^* M\{t/x, t/y\}.\end{aligned}$$

Observe that the requirement of typings for $[M]^w_{x,y}$ can be easily verified. □

The previous lemma suffices to prove every primitive recursive function to be representable inside $\mathsf{H}(\emptyset)$.

THEOREM 6.2. *For every primitive recursive function $f : \mathbb{N}^n \to \mathbb{N}$ there is a term M_f such that $\vdash_{\mathsf{H}(\emptyset)} M_f : \mathbb{U}^0 \overset{n}{\multimap} \mathbb{U}^0$ and M_f represents f.*

PROOF. Base functions are the constant $0 : \mathbb{N} \to \mathbb{N}$, the successor $s : \mathbb{N} \to \mathbb{N}$, and for every n, i, the projections $u_i^n : \mathbb{N}^n \to \mathbb{N}$. It can be easily checked that these functions are represented by

$$\begin{aligned}M_0 &\equiv \lambda x.c_2^{\mathbb{U}};\\ M_s &\equiv \lambda x.c_1^{\mathbb{U}}x;\\ M_{u_i^n} &\equiv \lambda x_1.\lambda x_2.\dots.\lambda x_n.x_i.\end{aligned}$$

Observe that

$$\begin{aligned}\vdash_{\mathsf{H}(\emptyset)} M_0 &: \mathbb{U}^0 \multimap \mathbb{U}^0;\\ \vdash_{\mathsf{H}(\emptyset)} M_s &: \mathbb{U}^0 \multimap \mathbb{U}^0;\\ \vdash_{\mathsf{H}(\emptyset)} M_{u_i^n} &: \mathbb{U}^0 \overset{n}{\multimap} \mathbb{U}^0.\end{aligned}$$

We now need some additional notation. Given a term M such that $x_1 : \mathbb{U}^0, \dots, x_n : \mathbb{U}^0 \vdash M : A$, we will define terms $M_i^{x_1,\dots,x_n}$ such that $x_i : \mathbb{U}^0 \vdash M_i^{x_1,\dots,x_n} : A$ for every i:

$$\begin{aligned}M_1^{x_1,\dots,x_n} &\equiv (\lambda x_1.\dots.\lambda x_n.M)x_1;\\ \forall i \geq 1.M_{i+1}^{x_1,\dots,x_n} &\equiv \left[M_i^{x_1,\dots,x_n}x_{i+1}\right]^{x_{i+1}}_{x_i,x_{i+1}}.\end{aligned}$$

We can prove the following by induction on i.

$$M_i^{x_1,\dots,x_n}\{t/x_i\} \rightsquigarrow^* (\lambda x_1.\dots.\lambda x_n.M)\underbrace{t\dots t}_{i\text{ times}}.$$

Indeed,

$$
\begin{array}{rcl}
M_1^{x_1,\dots,x_n}\{t/x_1\} & \equiv & (\lambda x_1.\dots.\lambda x_n.M)t; \\
\forall i \geq 1. M_{i+1}^{x_1,\dots,x_n}\{t/x_{i+1}\} & \equiv & [M_i^{x_1,\dots,x_n} x_{i+1}]_{x_i,x_{i+1}}^{x_{i+1}}\{t/x_{i+1}\} \\
& \rightsquigarrow^* & (M_i^{x_1,\dots,x_n} x_{i+1})\{t/x_i, t/x_{i+1}\} \\
& \rightsquigarrow^* & (M_i^{x_1,\dots,x_n}\{t/x_i\})t \\
& \rightsquigarrow^* & ((\lambda x_1.\dots.\lambda x_n.M)\underbrace{t\dots t}_{i \text{ times}})t \\
& \equiv & (\lambda x_1.\dots.\lambda x_n.M)\underbrace{t\dots t}_{i+1 \text{ times}}.
\end{array}
$$

In this way we can get a generalized variant of Lemma 6.1 by putting $\langle M\rangle_{x_1,\dots,x_n}^{z} \equiv (\lambda x_n.M_n^{x_1,\dots,x_n})z$. Indeed,

$$
\begin{array}{rcl}
\langle M\rangle_{x_1,\dots,x_n}^{z}\{t/z\} & \equiv & (\lambda x_n.M_n^{x_1,\dots,x_n})t \\
& \rightsquigarrow & M_n^{x_1,\dots,x_n}\{t/x_n\} \\
& \rightsquigarrow^* & (\lambda x_1.\dots.\lambda x_n.M)\underbrace{t\dots t}_{n \text{ times}} \\
& \rightsquigarrow^* & M\{t/x_1,\dots t/x_n\}.
\end{array}
$$

We are now ready to prove that composition and recursion can be represented in $\mathsf{H}(\emptyset)$. Suppose $f:\mathbb{N}^n \to \mathbb{N}$, $g_1,\dots,g_n:\mathbb{N}^m \to \mathbb{N}$ and let $h:\mathbb{N}^m \to \mathbb{N}$ be the function obtained by composing f with $g_1,\dots,g_n$, namely,

$$h(n_1,\dots,n_m) = f(g_1(n_1,\dots,n_m),\dots,g_n(n_1,\dots,n_m)).$$

We define

$$
\begin{array}{rcl}
N & \equiv & \lambda x_1^m.\dots.\lambda x_n^m.\dots.\lambda x_1^1.\dots.\lambda x_n^1.M_f\left(M_{g_1}x_1^1\dots x_1^m\right) \\
& & \qquad\qquad\qquad\qquad\qquad\qquad \dots \\
& & \qquad\qquad\qquad\qquad\qquad\qquad (M_{g_n}x_n^1\dots x_n^m); \\
M_h^m & \equiv & \left\langle N x_1^m\dots x_n^m\right\rangle_{x_1^m,\dots,x_n^m}^{y_m}; \\
\forall i < m. M_h^i & \equiv & \left\langle \lambda y_{i+1}.\left(M_h^{i+1}x_1^i\dots x_n^i\right)\right\rangle_{x_1^i,\dots,x_n^i}^{y_i}; \\
M_h & \equiv & \lambda y_1.M_h^1.
\end{array}
$$

Indeed:

$$
\begin{array}{rcl}
M_h\ulcorner n_1\urcorner\dots\ulcorner n_m\urcorner & \rightsquigarrow^* & \left(M_h^1\{\ulcorner n_1\urcorner/y_1\}\right)\ulcorner n_2\urcorner\dots\ulcorner n_m\urcorner \\
& \rightsquigarrow^* & \left(\lambda y_2.M_h^2\ulcorner n_1\urcorner\dots\ulcorner n_1\urcorner\right)\ulcorner n_2\urcorner\dots\ulcorner n_m\urcorner \\
& \rightsquigarrow & \left(M_h^2\{\ulcorner n_2\urcorner/y_2\}\ulcorner n_1\urcorner\dots\ulcorner n_1\urcorner\right)\ulcorner n_3\urcorner\dots\ulcorner n_m\urcorner \\
& \rightsquigarrow^* & \dots \\
& \rightsquigarrow^* & (\dots((N\ulcorner n_m\urcorner\dots\ulcorner n_m\urcorner)\ulcorner n_{m-1}\urcorner\dots\ulcorner n_{m-1}\urcorner)\dots)\ulcorner n_1\urcorner\dots\ulcorner n_1\urcorner \\
& \rightsquigarrow^* & M_f(M_{g_1}\ulcorner n_1\urcorner\dots\ulcorner n_m\urcorner)\dots(M_{g_n}\ulcorner n_1\urcorner\dots\ulcorner n_m\urcorner) \\
& \rightsquigarrow^* & \ulcorner f(g_1(n_1,\dots,n_m),\dots,g_n(n_1,\dots,n_m))\urcorner.
\end{array}
$$

Now, suppose $f : \mathbb{N}^m \to \mathbb{N}$ and $g : \mathbb{N}^{m+2} \to \mathbb{N}$ and let $h : \mathbb{N}^{m+1} \to \mathbb{N}$ be the function obtained by f and g by primitive recursion. We have

$$\begin{aligned} h(0, n_1, \ldots, n_m) &= f(n_1, \ldots, n_m); \\ h(n+1, n_1, \ldots, n_m) &= g(n, h(n, n_1, \ldots, n_m), n_1, \ldots, n_m). \end{aligned}$$

We define

$$\begin{aligned} N &\equiv \lambda x_m.\lambda y_m.\ldots.\lambda x_1.\lambda y_1.\lambda y.\lambda w.M_g y(wx_1 \ldots x_m)y_1 \ldots y_m; \\ M_h^m &\equiv [N x_m y_m]^{z_n}_{x_m, y_m}; \\ \forall i < m.M_h^i &\equiv \left[\lambda z_{i+1}.M_h^{i+1} x_i y_i\right]^{z_i}_{x_i, y_i}; \\ N_h &\equiv \lambda y.\lambda w.\lambda z_1.\ldots.\lambda z_m.M_h^1 z_2 \ldots z_m y w; \\ M_h &\equiv \lambda x.x\langle\langle N_h, M_f\rangle\rangle. \end{aligned}$$

Notice that

$$\begin{aligned} \ulcorner 0\urcorner\langle\langle N_h, M_f\rangle\rangle &\rightsquigarrow M_f \equiv V_0; \\ \ulcorner n+1\urcorner\langle\langle N_h, M_f\rangle\rangle &\rightsquigarrow N_h\ulcorner n\urcorner(\ulcorner n\urcorner\langle\langle N_h, M_f\rangle\rangle) \\ &\rightsquigarrow^* \lambda z_1.\ldots.\lambda z_m.M_h^1 z_2 \ldots z_n\ulcorner n\urcorner(\ulcorner n\urcorner\langle\langle N_h, M_f\rangle\rangle) \equiv V_n. \end{aligned}$$

Moreover, for every $\ulcorner n_1\urcorner, \ldots, \ulcorner n_m\urcorner$:

$$\begin{aligned} V_0\ulcorner n_1\urcorner \ldots \ulcorner n_m\urcorner &\rightsquigarrow^* M_f\ulcorner n_1\urcorner \ldots \ulcorner n_m\urcorner \rightsquigarrow \ulcorner h(0, n_1, \ldots, n_m)\urcorner; \\ V_{n+1}\ulcorner n_1\urcorner \ldots \ulcorner n_m\urcorner &\rightsquigarrow^* M_h^1\{\ulcorner n_1\urcorner/z_1\}\ulcorner n_2\urcorner \ldots \ulcorner n_m\urcorner\ulcorner n\urcorner(\ulcorner n\urcorner\langle\langle N_h, M_f\rangle\rangle) \\ &\rightsquigarrow^* \left(\lambda z_2.M_h^2\ulcorner n_1\urcorner\ulcorner n_1\urcorner\right)\ulcorner n_2\urcorner \ldots \ulcorner n_m\urcorner\ulcorner n\urcorner(\ulcorner n\urcorner\langle\langle N_h, M_f\rangle\rangle) \\ &\rightsquigarrow^* \left(\lambda z_3.M_h^3\ulcorner n_2\urcorner\ulcorner n_2\urcorner\ulcorner n_1\urcorner\ulcorner n_1\urcorner\right)\ulcorner n_3\urcorner \ldots \ulcorner n\urcorner(\ulcorner n\urcorner\langle\langle N_h, M_f\rangle\rangle) \\ &\rightsquigarrow^* \ldots \\ &\rightsquigarrow^* N\ulcorner n_m\urcorner\ulcorner n_m\urcorner \ldots \ulcorner n_1\urcorner\ulcorner n_1\urcorner\ulcorner n\urcorner(\ulcorner n\urcorner\langle\langle N_h, M_f\rangle\rangle) \\ &\rightsquigarrow^* N\ulcorner n_m\urcorner\ulcorner n_m\urcorner \ldots \ulcorner n_1\urcorner\ulcorner n_1\urcorner\ulcorner n\urcorner V_n \\ &\rightsquigarrow^* M_g\ulcorner n\urcorner(V_n\ulcorner n_1\urcorner \ldots \ulcorner n_m\urcorner)\ulcorner n_1\urcorner \ldots \ulcorner n_m\urcorner \\ &\rightsquigarrow^* M_g\ulcorner n\urcorner\ulcorner h(n, n_1, \ldots, n_m)\urcorner\ulcorner n_1\urcorner \ldots \ulcorner n_m\urcorner \\ &\rightsquigarrow^* \ulcorner g(n, h(n, n_1, \ldots, n_m), n_1, \ldots, n_m)\urcorner \\ &\equiv \ulcorner h(n+1, n_1, \ldots, n_m)\urcorner. \end{aligned}$$

This concludes the proof. □

The following two results show that functions representable in $\mathsf{RH}(\emptyset)$ (respectively, $\mathsf{RH}(\mathsf{A})$) combinatorially saturate **FP** (respectively, **FE**).

LEMMA 6.3. *There are terms* **Coerc**, **Add**, **Square** *such that for every* n, m

$$\begin{aligned} \mathbf{Coerc}\,\ulcorner n\urcorner &\rightsquigarrow^* \ulcorner n\urcorner; \\ \mathbf{Add}\,\ulcorner n\urcorner\,\ulcorner m\urcorner &\rightsquigarrow^* \ulcorner n+m\urcorner; \\ \mathbf{Square}\,\ulcorner n\urcorner &\rightsquigarrow^* \ulcorner n^2\urcorner. \end{aligned}$$

Moreover, for every $i \in \mathbb{N}$

$$\begin{aligned}\vdash_{\mathsf{RH}(\emptyset)} \mathbf{Coerc} &: \mathbb{U}^{i+1} \multimap \mathbb{U}^{i};\\ \vdash_{\mathsf{RH}(\emptyset)} \mathbf{Add} &: \mathbb{U}^{i+1} \multimap \mathbb{U}^{i} \multimap \mathbb{U}^{i};\\ \vdash_{\mathsf{RH}(\emptyset)} \mathbf{Square} &: \mathbb{U}^{i+3} \multimap \mathbb{U}^{i}.\end{aligned}$$

PROOF. **Coerc** is $\lambda x.x\langle\!\langle \lambda y.\lambda w.c_1^{\mathbb{U}}w, c_2^{\mathbb{U}}\rangle\!\rangle$. **Add** is $\lambda x.\lambda y.(x\langle\!\langle M_1, M_2\rangle\!\rangle)y$, where M_1 is $\lambda w.\lambda z.\lambda q.c_1^{\mathbb{U}}(zq)$ and M_2 is $\lambda z.z$. **Square** is

$$\lambda x.\big[\mathbf{Add}((\mathbf{Coerc}\ x_1)\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)((\mathbf{Predecessor}\ x_2)\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)\big]^{x}_{x_1,x_2}$$

where **Predecessor** is $\lambda x.x\{\!\!\{\lambda y.y, c_2^{\mathbb{U}}\}\!\!\}$ and **Adds** is $\lambda x.\mathbf{Add}(c_1^{\mathbb{U}}x)$. Indeed:

$$\begin{aligned}
c_2^{\mathbb{U}}\langle\!\langle \lambda y.c_1^{\mathbb{U}}y, c_2^{\mathbb{U}}\rangle\!\rangle &\rightsquigarrow c_2^{\mathbb{U}};\\
c_1^{\mathbb{U}}t\langle\!\langle \lambda y.\lambda w.c_1^{\mathbb{U}}w, c_2^{\mathbb{U}}\rangle\!\rangle &\rightsquigarrow (\lambda y.\lambda w.c_1^{\mathbb{U}}w)t(t\langle\!\langle \lambda y.\lambda w.c_1^{\mathbb{U}}w, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\rightsquigarrow^* (\lambda y.\lambda w.c_1^{\mathbb{U}}w)tt\\
&\rightsquigarrow^* c_1^{\mathbb{U}}t;\\
\mathbf{Coerc}\ t &\rightsquigarrow t\langle\!\langle \lambda y.\lambda w.c_1^{\mathbb{U}}w, c_2^{\mathbb{U}}\rangle\!\rangle \rightsquigarrow^* t;\\
\ulcorner 0\urcorner\langle\!\langle M_1, M_2\rangle\!\rangle &\rightsquigarrow \lambda y.y \equiv V_0;\\
\ulcorner n+1\urcorner\langle\!\langle M_1, M_2\rangle\!\rangle &\rightsquigarrow M_1\ulcorner n\urcorner(\ulcorner n\urcorner\langle\!\langle M_1, M_2\rangle\!\rangle)\\
&\rightsquigarrow (\lambda z.\lambda q.c_1^{\mathbb{U}}(zq))V_n \rightsquigarrow \lambda q.c_1^{\mathbb{U}}(V_n q) \equiv V_{n+1};\\
V_0\ulcorner m\urcorner &\rightsquigarrow \ulcorner m\urcorner;\\
V_{n+1}\ulcorner m\urcorner &\rightsquigarrow c_1^{\mathbb{U}}(V_n\ulcorner m\urcorner) \rightsquigarrow^* c_1^{\mathbb{U}}\ulcorner n+m\urcorner \equiv \ulcorner (n+1)+m\urcorner;\\
\mathbf{Add}\ulcorner n\urcorner\ulcorner m\urcorner &\rightsquigarrow (\ulcorner n\urcorner\langle\!\langle M_1, M_2\rangle\!\rangle)\ulcorner m\urcorner\\
&\rightsquigarrow^* V_n\ulcorner m\urcorner \rightsquigarrow^* \ulcorner n+m\urcorner;\\
\ulcorner 0\urcorner\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle &\rightsquigarrow \ulcorner 0\urcorner \equiv \ulcorner 0(0+1)/2\urcorner;\\
\ulcorner n+1\urcorner\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle &\rightsquigarrow \mathbf{Adds}\ulcorner n\urcorner(\ulcorner n\urcorner\langle\!\langle \mathbf{Add}, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\rightsquigarrow \mathbf{Add}\ulcorner n+1\urcorner(\ulcorner n(n+1)/2\urcorner)\\
&\rightsquigarrow^* \ulcorner n+1+n(n+1)/2\urcorner \equiv \ulcorner (n+1)(n+2)/2\urcorner;\\
\mathbf{Square}\ulcorner 0\urcorner &\rightsquigarrow^* \mathbf{Add}((\mathbf{Coerc}\ulcorner 0\urcorner)\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\qquad ((\mathbf{Predecessor}\ulcorner 0\urcorner)\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\rightsquigarrow^* \mathbf{Add}(\ulcorner 0\urcorner\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)(\ulcorner 0\urcorner\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\rightsquigarrow^* \mathbf{Add}\ulcorner 0\urcorner\ulcorner 0\urcorner\\
&\rightsquigarrow^* \ulcorner 0\urcorner \equiv \ulcorner 0^2\urcorner;\\
\mathbf{Square}\ulcorner n+1\urcorner &\rightsquigarrow^* \mathbf{Add}((\mathbf{Coerc}\ulcorner n+1\urcorner)\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\qquad ((\mathbf{Predecessor}\ulcorner n+1\urcorner)\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\rightsquigarrow^* \mathbf{Add}(\ulcorner n+1\urcorner\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)(\ulcorner n\urcorner\langle\!\langle \mathbf{Adds}, c_2^{\mathbb{U}}\rangle\!\rangle)\\
&\rightsquigarrow^* \mathbf{Add}\ulcorner (n+1)(n+2)/2\urcorner\ulcorner n(n+1)/2\urcorner\\
&\rightsquigarrow^* \ulcorner (n+1)(n+2)/2+n(n+1)/2\urcorner \equiv \ulcorner (n+1)^2\urcorner.
\end{aligned}$$

This concludes the proof. □

In presence of ramification, an exponential behavior can be obtained by exploiting contraction on tree-algebraic types.

LEMMA 6.4. *There is a term* **Exp** *such that for every* n

$$\mathbf{Exp}\ulcorner n\urcorner \rightsquigarrow^* \ulcorner 2^n\urcorner.$$

Moreover, for every $i \in \mathbb{N}$

$$\vdash_{\mathsf{RH(A)}} \mathbf{Exp} : \mathbb{U}^{i+2} \multimap \mathbb{U}^i.$$

PROOF. For every $n \in \mathbb{N}$, we will denote by $ct(n)$ the complete binary tree of height n in $\mathscr{E}_{\mathbb{C}}$. we have

$$\begin{aligned} ct(0) &= c_2^{\mathbb{C}};\\ ct(n+1) &= c_1^{\mathbb{C}}(ct(n))(ct(n)). \end{aligned}$$

For every n, there are 2^n occurrences of $c_2^{\mathbb{C}}$ inside $ct(n)$. We will now define two terms **Blowup** and **Leaves** such that

$$\begin{aligned} \mathbf{Blowup}\ulcorner n\urcorner &\rightsquigarrow^* ct(n);\\ \mathbf{Leaves}\,(ct(n)) &\rightsquigarrow^* \ulcorner 2^n\urcorner. \end{aligned}$$

We define

$$\begin{aligned} \mathbf{Blowup} &\equiv \lambda x.x\langle\!\langle \lambda y.\lambda w.c_1^{\mathbb{C}}ww, c_2^{\mathbb{C}}\rangle\!\rangle;\\ \mathbf{Leaves} &\equiv \lambda x.(x\langle\!\langle M_1, M_2\rangle\!\rangle)c_2^{\mathbb{U}};\\ \mathbf{Exp} &\equiv \lambda x.\mathbf{Leaves}(\mathbf{Blowup}\,x); \end{aligned}$$

where

$$\begin{aligned} M_1 &\equiv \lambda y.\lambda w.\lambda z.\lambda q.\lambda r.z(qr);\\ M_2 &\equiv \lambda x.c_1^{\mathbb{U}}x. \end{aligned}$$

Indeed,

$$\begin{aligned} \ulcorner 0\urcorner\langle\!\langle \lambda y.c_1^{\mathbb{C}}yy, c_2^{\mathbb{C}}\rangle\!\rangle &\rightsquigarrow c_2^{\mathbb{C}} \equiv ct(0);\\ \ulcorner n+1\urcorner\langle\!\langle \lambda y.\lambda w.c_1^{\mathbb{C}}ww, c_2^{\mathbb{C}}\rangle\!\rangle &\rightsquigarrow \big(\lambda y.\lambda w.c_1^{\mathbb{C}}ww\big)\ulcorner n\urcorner\big(\ulcorner n\urcorner\langle\!\langle \lambda y.\lambda w.c_1^{\mathbb{C}}ww, c_2^{\mathbb{C}}\rangle\!\rangle\big)\\ &\rightsquigarrow^* \big(\lambda y.\lambda w.c_1^{\mathbb{C}}ww\big)\ulcorner n\urcorner(ct(n))\\ &\rightsquigarrow^* c_1^{\mathbb{C}}(ct(n))(ct(n)) \equiv ct(n+1) \end{aligned}$$

$$
\begin{array}{rcl}
\mathbf{Blowup}\ulcorner n\urcorner & \rightsquigarrow & \ulcorner n\urcorner\langle\!\langle \lambda y.c_1^{\mathbb{C}} y\,y, c_2^{\mathbb{C}}\rangle\!\rangle \rightsquigarrow (ct(n)); \\
(ct(0))\langle\!\langle M_1, M_2\rangle\!\rangle & \rightsquigarrow & \lambda x.c_1^{\mathbb{U}} x \equiv V_0; \\
(ct(n+1))\langle\!\langle M_1, M_2\rangle\!\rangle & \rightsquigarrow^* & \lambda r.V_n(V_n r) \equiv V_{n+1}; \\
V_0\ulcorner m\urcorner & \rightsquigarrow & \ulcorner 1+m\urcorner \equiv \ulcorner 2^0+m\urcorner; \\
V_{n+1}\ulcorner m\urcorner & \rightsquigarrow & V_n(V_n\ulcorner m\urcorner) \rightsquigarrow^* V_n\ulcorner 2^n+m\urcorner \\
 & \rightsquigarrow^* & \ulcorner 2^n+2^n+m\urcorner \equiv \ulcorner 2^{n+1}+m\urcorner; \\
\mathbf{Leaves}(ct(n)) & \rightsquigarrow & \big((ct(n))\langle\!\langle \lambda y.\lambda w.\lambda z.\lambda q.\lambda r.z(qr).\lambda x.c_1^{\mathbb{U}} x\rangle\!\rangle\big)c_2^{\mathbb{U}} \\
 & \rightsquigarrow^* & V_n\ulcorner 0\urcorner \rightsquigarrow^* \ulcorner 2^n\urcorner; \\
\mathbf{Exp}\ulcorner n\urcorner & \rightsquigarrow & \mathbf{Leaves}(\mathbf{Blowup}\ulcorner n\urcorner) \rightsquigarrow^* \mathbf{Leaves}(ct(n)) \\
 & \rightsquigarrow^* & \ulcorner 2^n\urcorner.
\end{array}
$$

This concludes the proof. □

The last two lemmas are not completeness results, but help in the so-called *quantitative* part of the encoding of Turing machines. Indeed, **FP** can be embedded into $\mathsf{RH}(\emptyset)$, while **FE** can be embedded into $\mathsf{RH}(\mathsf{A})$.

THEOREM 6.5. *For every polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ there are a term M_f and an integer n_f such that $\vdash_{\mathsf{RH}(\emptyset)} M_f : \mathbb{B}^{n_f} \to \mathbb{B}^0$ and M_f represents f. For every elementary-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ there are a term M_f and an integer n_f such that $\vdash_{\mathsf{RH}(\mathsf{A})} M_f : \mathbb{B}^{n_f} \to \mathbb{B}^0$ and M_f represents f.*

PROOF. First of all, we can observe that for ever polynomial $p : \mathbb{N} \to \mathbb{N}$, there are another polynomial $\overline{p} : \mathbb{N} \to \mathbb{N}$, an integer n_p, and term $M_{\overline{p}}$ such that

$$
\begin{array}{rcl}
\forall n \in \mathbb{N}.\overline{p}(n) & \geq & p(n); \\
\forall n \in \mathbb{N}.\ \vdash_{\mathsf{RH}(\emptyset)} & M_{\overline{p}} : \mathbb{B}^{n_p+n} \multimap \mathbb{B}^n; &
\end{array}
$$

and $M_{\overline{p}}$ represents $\overline{p}$. $\overline{p}$ is simply p where all monomials x^k are replaced by x^{2^l} (where $k \leq 2^l$) and $M_{\overline{p}}$ is built up from terms in $\mathscr{E}_{\mathbb{U}}$, **Add**, **Square**, and **Coerc** (see Lemma 6.3). Analogously, for every elementary function $p : \mathbb{N} \to \mathbb{N}$, there are another elementary function $\overline{p} : \mathbb{N} \to \mathbb{N}$, an integer n_p, and term $M_{\overline{p}}$ such that

$$
\begin{array}{rcl}
\forall n \in \mathbb{N}.\overline{p}(n) & \geq & p(n); \\
\forall n \in \mathbb{N}.\ \vdash_{\mathsf{RH}(\mathsf{A})} & M_{\overline{p}} : \mathbb{B}^{n_p+n} \multimap \mathbb{B}^n; &
\end{array}
$$

and $M_{\overline{p}}$ represents $\overline{p}$. This time, $\overline{p}$ is a tower

$$
\overline{p}(n) = 2^{\cdot^{\cdot^{2^n}}} \Big\}_k \ \textit{times}
$$

obtained from p by applying a classical result on elementary functions, while $M_{\overline{p}}$ is built up from terms in $\mathscr{E}_{\mathbb{U}}$, **Add**, **Coerc**, and **Exp** (see Lemma 6.4).

Now, consider a Turing machine $\mathcal{M}$ working in polynomial time. Configurations for $\mathcal{M}$ are quadruples (*state*, *left*, *right*, *current*), where *state* belongs to a finite set of states, *left*, *right* $\in \Sigma^*$ (where Σ is a finite alphabet) are the contents of the left and right portion of the tape, and *current* $\in \Sigma$ is the symbol currently read by the head. It it not difficult to encode configurations of $\mathcal{M}$ by terms in $\mathscr{E}_{\mathbb{D}}$ in such a way that terms M_{init}, M_{final}, M_{trans} exist such that:

(1) $M_{init}\ulcorner s \urcorner$ rewrites to the term encoding the initial configuration on s, M_{final} extracts the result from a final configuration, and M_{trans} represents the transition function of $\mathcal{M}$.

(2) For every n,

$$\begin{aligned} \vdash_{\mathsf{RH}(\emptyset)} M_{init} &: \mathbb{B}^{n+1} \multimap \mathbb{D}^n; \\ \vdash_{\mathsf{RH}(\emptyset)} M_{final} &: \mathbb{D}^{n+1} \multimap \mathbb{B}^n; \\ \vdash_{\mathsf{RH}(\emptyset)} M_{trans} &: \mathbb{D}^n \multimap \mathbb{D}^n. \end{aligned}$$

Moreover, there is a term M_{length} such that $M_{length}\ulcorner s \urcorner \rightsquigarrow^* \ulcorner |s| \urcorner$ for every $s \in \{0,1\}^*$. Let now $p : \mathbb{N} \to \mathbb{N}$ be a polynomial bounding the running time of $\mathcal{M}$. The function computed by $\mathcal{M}$ is the one represented by the term

$$M_{\mathcal{M}} \equiv \lambda x.\langle M_{final}(((M_{\overline{p}}(M_{length}\, y))\langle\!\langle \lambda x.\lambda y.\lambda w.y(M_{trans}w), \lambda x.x\rangle\!\rangle)(M_{init}z))\rangle^x_{y,z}$$

where $\langle M\rangle^x_{y,z}$ is the generalization of $[M]^z_{x,y}$ to the algebra $\mathbb{B}$.

If $\mathcal{M}$ works in elementary time, we can proceed in the same way. $\square$

7. CONCLUSIONS

We introduced a typed lambda calculus equivalent to Gödel System T and a new context-based semantics for it. We then characterized the expressive power of various subsystems of the calculus, all of them being obtained by imposing linearity and ramification constraints. To the author's knowledge, the only fragment whose expressive power has been previously characterized is $\mathsf{RH}(\mathsf{W})$ [Hofmann 2000; Bellantoni et al. 2000; Dal Lago et al. 2003]. In studying the combinatorial dynamics of normalization, the semantics has been exploited in an innovative way.

There are other systems to which our semantics can be applied. This, in particular, includes nonsize-increasing polynomial-time computation [Hofmann 1999] and the calculus capturing **NC** by Aehlig et al. [2001]. Moreover, we believe higher-order contraction can be accommodated in the framework by techniques similar to the ones in Gonthier et al. [1992].

The most interesting development, however, consists in studying the applicability of our semantics to the automatic extraction of runtime bounds from programs. This, however, goes beyond the scope of this article and is left to future investigations.

ACKNOWLEDGMENTS

The author would like to thank S. Martini and L. Roversi for their support and the anonymous referees for many useful comments.

REFERENCES

Aehlig, K., Johannsen, J., Schwichtenberg, H., and Terwijn, S. A. 2001. Linear ramified higher type recursion and parallel complexity. In *Proof Theory in Computer Science*. Lecture Notes in Computer Science, vol. 2183, 1–21.

BAILLOT, P. AND PEDICINI, M. 2001. Elementary complexity and geometry of interaction. *Fundam. Inf. 45,* 1-2, 1–31.

BELLANTONI, S. AND COOK, S. 1992. A new recursion-theoretic characterization of the polytime functions. *Comput. Complexity 2*, 97–110.

BELLANTONI, S., NIGGL, K. H., AND SCHWICHTENBERG, H. 2000. Higher type recursion, ramification and polynomial time. *Ann. Pure App. Logic 104*, 17–30.

BONFANTE, G., MARION, J.-Y., AND MOYEN, J.-Y. 2004. On complexity analysis by quasi-interpretations. *Theor. Comput. Sci*. To appear.

DAL LAGO, U. 2006. Context semantics, linear logic and computational complexity. In *Proceedings of the 21th IEEE Syposium on Logic in Computer Science,* 169–178.

DAL LAGO, U. AND MARTINI, S. 2006. An invariant cost model for the lambda calculus. In *Proceedings of the. 2nd Conference on Computability in Europe*. Lecture Notes in Computer Science, vol. 3988. 105–114.

DAL LAGO, U., MARTINI, S., AND ROVERSI, L. 2003. Higher order linear ramified recurrence. In *Types for Proofs and Programs, Post-Workshop Proceedings*. Lecture Notes in Computer Science, vol. 3085. 178–193.

DANOS, V., HERBELIN, H., AND REGNIER, L. 1996. Game semantics & abstract machines. In *Proceedings of the 11th IEEE Syposium on Logic in Computer Science,* 394–405.

DANOS, V. AND REGNIER, L. 1999. Reversible, irreversible and optimal lambda-machines. *Theor. Comput. Sci. 227,* 1-2, 79–97.

GHICA, D. 2005. Slot games: A quantitative model of computation. In *Proceedings of the 32nd ACM Symposium on Principles of Programming Languages,* 85–97.

GIRARD, J.-Y. 1988. Geometry of interaction 2: Deadlock-Free algorithms. In *Proceedings of the Conference on Computer Logic*. Lecture Notes in Computer Science, vol. 417. 76–93.

GIRARD, J.-Y. 1989. Geometry of interaction 1: Interpretation of system F. In *Proceedings of the Logic Colloquium 88*, 221–260.

GIRARD, J.-Y. 1998. Light linear logic. *Inf. Comput. 143, 2,* 175–204.

GONTHIER, G., ABADI, M., AND LÉVY, J.-J. 1992. The geometry of optimal lambda reduction. In *Proceedings of the 12th ACM Symposium on Principles of Programming Languages*, 15–26.

HOFMANN, M. 1999. Linear types and non-size-increasing polynomial time computation. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*, 464–473.

HOFMANN, M. 2000. Safe recursion with higher types and BCK-algebra. *Ann. Pure Appl. Logic 104*, 113–166.

JOACHIMSKI, F. AND MATTHES, R. 2003. Short proofs of normalization for the simply-typed lambda-calculus, permutative conversions and Gödel's T. *Archive Math. Logic 42,* 1, 59–87.

LAFONT, Y. 2004. Soft linear logic and polynomial time. *Theor. Comput. Sci. 318*, 163–180.

LEIVANT, D. 1993. Stratified functional programs and computational complexity. In *Proceedings of the 20th ACM Symposium on Principles of Programming Languages*, 325–333.

LEIVANT, D. 1999a. Applicative control and computational complexity. In *Proceedings of the 13th International Workshop on Computer Science Logic*. Lecture Notes in Computer Science, vol. 1685. 82–95.

LEIVANT, D. 1999b. Ramified recurrence and computational complexity III: Higher type recurrence and elementary complexity. *Ann. Pure Appl. Logic 96*, 209–229.

LEIVANT, D. 2002. Intrinsic reasoning about functional programs I: First order theories. *Ann. Pure Appl. Logic 114,* 1-3, 117–153.

LEIVANT, D. 2004. Intrinsic reasoning about functional programs II: Unipolar induction and primitive-recursion. *Theor. Comput. Sci. 318,* 1-2, 181–196.

LEIVANT, D. AND MARION, J.-Y. 1994. Ramified recurrence and computational complexity II: Substitution and poly-space. In *Proceedings of the 8th International Workshop on Computer Science Logic*. Lecture Notes in Computer Science, vol. 933, 486–500.

MAIRSON, H. 2002. From Hilbert spaces to Dilbert spaces: Context semantics made simple. In *Proceedings of the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science*. Lecture Notes in Computer Science, vol. 2556, 2–17.

OSTRIN, G. AND WAINER, S. 2002. Proof theoretic complexity. In *Proof and System Reliability*, H. Schwichtenberg and R. Steinbrüggen, Eds. NATO Science Series, vol. 62. Kluwer, 369–398.

SANDS, D. 1991. Operational theories of improvement in functional languages. In *Proceedings of the Glasgow Workshop on Functional Programming*, 298–311.
SIMMONS, H. 2005. Tiering as a recursion technique. *Bull. Symbol. Logic 11,* 3, 321–350.

Received February 2006; revised October 2006; accepted February 2007