Church $\Rightarrow$ Scott $=$ Ptime: an application of resource sensitive realizability

Aloïs Brunel
ENS Lyon
alois.brunel@ens-lyon.org

Kazushige Terui
RIMS, Kyoto University
terui@kurims.kyoto-u.ac.jp

We introduce a variant of linear logic with second order quantifiers and type fixpoints, both restricted to purely linear formulas. The Church encodings of binary words are typed by a standard non-linear type 'Church,' while the Scott encodings (purely linear representations of words) are by a linear type 'Scott.' We give a characterization of polynomial time functions, which is derived from (Leivant and Marion 93): a function is computable in polynomial time if and only if it can be represented by a term of type Church $\Rightarrow$ Scott.

To prove soundness, we employ a resource sensitive realizability technique developed by Hofmann and Dal Lago.

1 Introduction

The field of implicit computational complexity aims to provide abstract, qualitative, machine-independent characterizations of complexity classes such as polynomial time and polynomial space functions. Along its development, two crucial factors for bounding complexity of programs have been identified:

Linearity: In the higher order setting, non-linear use of function variables often causes an exponential growth of execution time. Hence a natural approach is to restrict use of higher order variables, often using types, in order to capture the desired complexity classes. Examples are light linear/affine logics [11, 2], their variant dual light affine logic [3], soft linear logic [15], and mixtures of linear higher order types with safe recursion (eg., [5], [13]). These logics all capture polynomial time functions, while there are also systems corresponding to polynomial space [10] and elementary functions [11].

Data tiering: Another source of exponential explosion lies in nested use of recursion, as observed by [4, 16]. Hence one naturally restricts the structure of primitive recursive programs by data tiering. This approach is most extensively pursued by a series of papers by Leivant and Marion on tiered recursion (ramified recurrence) [17, 18, 19, 20, 21]. In tiered recursion, one has a countable number of copies of the binary word algebra, distinguished by tiers. Then a bad nesting of primitive recursion is avoided by requiring that the output of the defined function has a lower tier than the variable it recurses on.

Data tiering and higher order functionals. Along their development of ramified recurrence, Leivant and Marion have made an interesting observation in [18], which reveals an intimate relationship between data tiering and higher order functionals. They consider a simply typed λ-calculus over a first order word algebra, called $\mathbf{1}\lambda^{\mathbf{p}}(\mathbf{W})$. The system is inherently equipped with two "tiers": the first order word algebra (of base type o) as the lower "tier," and the Church encodings of words (of higher order type $(\tau \to \tau)^2 \to \tau \to \tau$) as the higher one. First order words are just bit strings, while Church words internalize

P. Baillot (Ed.): Workshop on Developments in
Implicit Computational complExity (DICE 2010).
EPTCS 23, 2010, pp. 31–46, doi:10.4204/EPTCS.23.3

the iteration scheme. Due to this inherent tiering, the programs from the Church words to the first order words capture the polynomial time functions.

What their work reveals is a rather logical nature of tiers; in the end, tiering is nothing but the distinction between first order and higher order data. It is then natural to go one step further towards the logical direction, by replacing the first order algebra with the *linear* lambda terms, and by identifying the higher order data with *non-linear* terms. Our intuition is backed up by the fact that the linear encoding of words, often attributed to Scott (cf. [1]), behaves very similarly to the first order words; for instance, they admit constant time successor, predecessor and discriminator, while they are not enhanced with the power of iteration in their own.

To identify the set of Scott words, it is useful to introduce a type system with linearity and type fixpoints. We therefore introduce a variant of linear logic, called $\mathbf{DIAL}_{lin}$, as a typing system for the pure λ-terms. This system distinguishes non-linear and linear arrows and has second order quantifiers and type fixpoints, both restricted to linear types. Morally, the base type of $\mathbf{1}\lambda^{\mathbf{P}}(\mathbf{W})$ corresponds to the hereditarily linear formulas of $\mathbf{DIAL}_{lin}$, and the higher types of $\mathbf{1}\lambda^{\mathbf{P}}(\mathbf{W})$ to the non-linear formulas. We then characterize the class of polynomial time functions as those represented by terms of type: 'Church' (nonlinear words) ⇒ 'Scott' (linear words). The two types for binary words play the role of the two tiers. Our work thus exhibits a connection between the two factors controlling complexity: linearity and data tiering.

Resource sensitive realizability. Following some preceding works [12, 13, 14], Dal Lago and Hofmann have introduced in [6] a realizability semantics which is useful to reason about the complexity bounds for various systems uniformly. In their framework, the realizers are pure λ-terms (values, to be more precise) under the weak call-by-value semantics, and they come equipped with the resource bounds expressed by elements of a resource monoid. Various systems are then dealt with by choosing a suitable resource monoid, while the basic realizability constructions are unchanged. This framework has offered new and uniform proofs of the soundness theorems for LAL, EAL, LFPL, SAL and BLL with respect to the associated complexity classes [6, 7].

We here apply their technique to prove that all terms of type Church ⇒ Scott in the system $\mathbf{DIAL}_{lin}$ are polytime. The main novelty is that we build a suitable (partial) resource monoid based on higher order polynomials. Also, we do not require that realizers are values. This allows us to directly infer the complexity bounds of arbitrary λ terms (not restricted to values).

Outline. Section 2 introduces the system $\mathbf{DIAL}_{lin}$ and states the main results. Section 3 introduces the realizability semantics and proves the adequacy theorem. Section 4 applies these tools to derive the soundness theorem. Section 5 concludes this work.

2 System $\mathbf{DIAL}_{lin}$

In this section, we recall the weak call-by-value λ-calculus with the time cost measure of [9], and then introduce the type system $\mathbf{DIAL}_{lin}$ derived from second order affine linear logic with type fixpoints. The system emulates the two tiers of $\mathbf{1}\lambda^{\mathbf{P}}(\mathbf{W})$ by distinguishing linear and non-linear types.

2.1 Weak call-by-value lambda calculus with time measure

We assume that a set of variables $x, y, z, \ldots$ are given. As usual, the *λ-terms* t, u are defined by the grammar: $t, u ::= x \mid \lambda x.t \mid tu$. The set of λ-terms is denoted by Λ. Terms of the form x or $\lambda x.t$ are called

values. We denote by $FV(t)$ the set of the free variables of t and by $[\![t]\!]_\beta$ the β-normal form of t. The *size* $|t|$ of a term t is defined by:

$$|x| = 1, \qquad |\lambda x.t| = |t| + 1, \qquad |tu| = |t| + |u|.$$

As with [9], we adopt the *weak call-by-value* reduction strategy, which is defined by:

$$\frac{}{(\lambda x.t)v \to t[v/x]} \qquad \frac{t_1 \to t_2}{t_1 u \to t_2 u} \qquad \frac{t_1 \to t_2}{u t_1 \to u t_2}$$

where v denotes a value. We write $t \Downarrow$ if t evaluates to a value v: $t \to^* v$. The value v is unique whenever $t \to^* v$, so we write $[\![t]\!] = v$. It should not be confused with the β-normal form $[\![t]\!]_\beta$ of t.

The cost of evaluation is specified by a ternary relation $t \overset{n}{\twoheadrightarrow} u$, meaning that t reduces to u with cost n, defined as follows:

$$\frac{}{t \overset{0}{\twoheadrightarrow} t} \qquad \frac{t \to u \quad n = max\{|u| - |t|, 1\}}{t \overset{n}{\twoheadrightarrow} u} \qquad \frac{s \overset{n}{\twoheadrightarrow} t \quad t \overset{m}{\twoheadrightarrow} u}{s \overset{n+m}{\twoheadrightarrow} u}$$

The definition takes into account the cost of duplications. In particular we have:

Lemma 2.1. *Suppose that $(\lambda x.t)v \overset{n}{\twoheadrightarrow} t[v/x]$ and x occurs c times in t. Then $n = 1$ if $c \leq 1$, and $n \leq (c-1)|v|$ if $c \geq 2$.*

Proof. In the first case, $|t[v/x]| < |(\lambda x.t)v|$. In the second case, $|t[v/x]| - |(\lambda x.t)v| \leq |t| + c|v| - (|t| + 1 + |v|) \leq (c-1)|v|$. □

A distinctive feature of the above cost model is that the cost n is unique: $t \overset{n}{\twoheadrightarrow} v$ and $t \overset{m}{\twoheadrightarrow} v$ imply $n = m$ [9]. So we may define $Time(t) = n$ without ambiguity ($Time(t)$ is undefined if $t \not\Downarrow$). Finally, let $TS(t) = Time(t) + |t|$. (It should be noticed that $TS(t)$ is denoted as $Time(t)$ in [9]; our notation is due to [8].)

It is proved in [9] that this cost model is invariant, which means that λ-calculus and Turing machines simulate each other with a polynomial time overhead. In particular, we have:

Theorem 2.2. *There exists a Turing machine M_{eval} with the following property: given a λ-term t such that $t \Downarrow$ and $TS(t) = Time(t) + |t| = n$, M_{eval} computes $[\![t]\!]$ in time $O(n^4)$.*

The following facts (cf. [9]) will be useful below.

Lemma 2.3. *The following hold when $t \Downarrow$.*

(size) $|[\![t]\!]| \leq TS(t)$.

(exchange) *If $t = (\lambda x_1 x_2.s)u_1 u_2$ and $t' = (\lambda x_2 x_1.s)u_2 u_1$, then $TS(t') = TS(t)$.*

(contraction) *If $t = (\lambda x_1 x_2.s)uu$ and $t' = (\lambda x.s[x/x_1, x/x_2])u$, then $TS(t') \leq TS(t)$.*

(weakening) *If $t' = (\lambda x.t)u$, $x \notin FV(t)$ and $u \Downarrow$, then $TS(t') = TS(t) + TS(u) + 2$.*

(concatenation) *If $t = s_1((\lambda x.s_2)u)$ and $t' = (\lambda x.s_1 s_2)u$ ($x \notin FV(s_1)$), then $TS(t') = TS(t)$.*

(identity) *If $t' = (\lambda x.x)t$, then $TS(t') = TS(t) + 3$.*

Proof. For (size), it is sufficient to prove that if $t \to u$ then $TS(t) \geq TS(u)$. If $|u| - |t| \geq 1$, we have $TS(t) = Time(t) + |t| = (|u| - |t| + Time(u)) + |t| = Time(u) + |u| = TS(u)$. Otherwise, $TS(t) = (1 + Time(u)) + |t| \geq Time(u) + |u| = TS(u)$.

For (weakening), we have $t' \xrightarrow{n} (\lambda x.t)[\![u]\!] \xrightarrow{1} t$ with $n = Time(u)$. Hence $TS(t') = Time(t') + |t'| = (n+1+Time(t)) + (|t| + |u| + 1) = TS(t) + TS(u) + 2$.

For (identity), we have $t' \xrightarrow{n} (\lambda x.x)[\![t]\!] \xrightarrow{1} [\![t]\!]$ with $n = Time(t)$. Hence $TS(t') = Time(t') + |t'| = (n+1) + (|t| + 2) = TS(t) + 3$.

For (contraction), we have $t \xrightarrow{2n} (\lambda x_1 x_2.s)[\![u]\!][\![u]\!] \xrightarrow{m} s[[\![u]\!]/x_1, [\![u]\!]/x_2] = t_0$ and $t' \xrightarrow{n} (\lambda x.s[x/x_1, x/x_2])[\![u]\!] \xrightarrow{k} t_0$. Consider the case when each of x_1 and x_2 occurs more than once in s. Then $m = |t_0| - |(\lambda x_1 x_2.s)[\![u]\!][\![u]\!]|$ and $k = |t_0| - |(\lambda x.s[x/x_1, x/x_2])[\![u]\!]|$. Hence we have:

$$\begin{aligned} TS(t) &= 2n + (|t_0| - |(\lambda x_1 x_2.s)[\![u]\!][\![u]\!]|) + Time(t_0) + |t| \\ &= 2n + TS(t_0) + 2|u| - 2|[\![u]\!]|; \\ TS(t') &= n + (|t_0| - |(\lambda x.s[x/x_1, x/x_2])[\![u]\!]|) + Time(t_0) + |t'| \\ &= n + TS(t_0) + |u| - |[\![u]\!]|. \end{aligned}$$

By (size), we have $|[\![u]\!]| \leq TS(u) = n + |u|$, hence we conclude $TS(t') \leq TS(t)$. The calculation is similar when either z_1 or z_2 occurs at most once.

The equations for (exchange) and (concatenation) are easily verified. □

2.2 The dual type system

We now introduce the system **DIAL**$_{lin}$: the dual intuitionistic affine logic with linear quantifiers and type fixpoints. It is based on intuitionistic linear logic with unrestricted weakening (thus "linear" actually means "affine"). It does not possess the ! connective but distinguishes linear and non-linear function spaces as in [3]. It has the second order quantifier and the type fixpoint operator, but both are restricted to purely linear formulas.

Given a set of propositional variables $\alpha, \beta, \ldots$, the (general) *formulas* $A, B, \ldots$ and the *linear formulas* $L, M, \ldots$ are defined by the following grammar:

$$L, M ::= \alpha \mid \forall \alpha L \mid \mu \alpha L^{(*)} \mid L \multimap M, \qquad A, B ::= L \mid \forall \alpha A \mid L \multimap B \mid A \Rightarrow B.$$

$(*)$: we add the condition that we can build $\mu \alpha L$ only if α occurs only positively in L. This is a common restriction that makes it easier to interpret fixpoint types in realizability semantics.

Thus the linear formulas are the formulas that do not contain any $\Rightarrow$.

We handle judgments of the form $\Gamma; \Delta \vdash t : A$, where Δ consists of assignments of the form $(x : L)$ with L a linear formula, and Γ consists of $(x : A)$ with A an arbitrary formula. We assume that variables in Γ and Δ are distinct. The variables in Δ are intended to be affine linear: each of them occurs at most once in t, in contrast to those in Γ which may have multiple occurrences. The typing rules are defined in Figure 1. Notice that L always denotes a linear formula.

We say that a term t is *of type* A in **DIAL**$_{lin}$ if $\vdash t : A$ is derivable by the typing rules in Figure 1. Below are some remarks.

- The intended meaning of judgment $\Gamma; \Delta \vdash t : A$ is $!\Gamma^*, \Delta \vdash t : A^*$, where Γ^*, A^* are translations into linear logic given by $(B \Rightarrow C)^* = !B^* \multimap C^*$. Hence the rule (Contr) can be applied only to variables in Γ.

$$\frac{}{x:A;\vdash x:A}\,(ax1) \qquad \frac{}{;x:L\vdash x:L}\,(ax2)$$

$$\frac{\Gamma;\Delta\vdash t:\mu\alpha L}{\Gamma;\Delta\vdash t:L[\mu\alpha L/\alpha]}\,(\mu_e) \qquad \frac{\Gamma;\Delta\vdash t:L[\mu\alpha L/\alpha]}{\Gamma;\Delta\vdash t:\mu\alpha L}\,(\mu_i)$$

$$\frac{\Gamma;\Delta\vdash t:A \qquad \alpha\notin FV(\Gamma;\Delta)}{\Gamma;\Delta\vdash t:\forall\alpha A}\,(\forall_i) \qquad \frac{\Gamma;\Delta\vdash t:\forall\alpha A}{\Gamma;\Delta\vdash t:A[L/\alpha]}\,(\forall_e)$$

$$\frac{\Gamma_1;\Delta\vdash t:A\Rightarrow B \qquad \Gamma_2;\vdash u:A}{\Gamma_1,\Gamma_2;\Delta\vdash tu:B}\,(\Rightarrow_e) \qquad \frac{\Gamma,z:A;\Delta\vdash t:B}{\Gamma;\Delta\vdash \lambda zt:A\Rightarrow B}\,(\Rightarrow_i)$$

$$\frac{\Gamma_1;\Delta_1\vdash t:L\multimap B \qquad \Gamma_2;\Delta_2\vdash u:L}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2\vdash tu:B}\,(\multimap_e) \qquad \frac{\Gamma;\Delta,z:L\vdash t:B}{\Gamma;\Delta\vdash \lambda zt:L\multimap B}\,(\multimap_i)$$

$$\frac{\Gamma,x:A,y:A;\Delta\vdash t:B}{\Gamma,z:A;\Delta\vdash t[z/x,z/y]:B}\,(Contr) \qquad \frac{\Gamma;\Delta,x:L\vdash t:B}{\Gamma,x:L;\Delta\vdash t:B}\,(Derel) \qquad \frac{\Gamma;\Delta\vdash t:B}{\Gamma,\Gamma';\Delta,\Delta'\vdash t:B}\,(Weak)$$

Figure 1: Typing rules of $\mathbf{DIAL}_{lin}$

- The $\Rightarrow_e$ rule implicitly performs the ! promotion on A, so the judgment for u should not contain a linear variable.
- We only allow substitution of linear formulas for propositional variables (in rules $(\forall_e)$, (μ_i) and (μ_e)). One can check that such a substitution in a formula always results in a formula. This restriction is strictly necessary, since the exponential function would be typed otherwise (see below).
- One unpleasant restriction is that the premise L of a linear implication $L\multimap B$ has to be linear. It does not seem essential for complexity, but our realizability argument forces it.
- The type system enjoys the subject reduction property with respect to the β-reduction.

2.3 Church and Scott data types

In $\mathbf{DIAL}_{lin}$, data may be represented in two ways, either in the Church style or in the Scott style. Figure 2 illustrates the two encodings for natural numbers n and binary words $w\in\{0,1\}^*$, together with some basic functions defined on them. In the definition of $\mathsf{w}^\bullet$, w is assumed to be $i_1\cdots i_n$ where each i_k is either 0 or 1.

The first thing to be verified is the following:

Proposition 2.4. *For every term t in β-normal form, $\vdash t:\mathsf{N}^\bullet$ if and only if t is a Church numeral $\mathsf{n}^\bullet$ (or $\lambda x.x$, that is η-equivalent to $\mathsf{1}^\bullet$). $\vdash t:\mathsf{N}^\circ$ if and only if t is a Scott numeral n°. Similarly for Church and Scott words.*

Proof. The claim is standard for Church numerals. So let us focus on Scott numerals. The following

derivations show that $\vdash \mathsf{n}^\circ : \mathsf{N}^\circ$ for every natural number n.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{;y:\alpha \vdash y:\alpha}}{;x:\mathsf{N}^\circ \multimap \alpha, y:\alpha \vdash y:\alpha}}{\vdash \mathsf{0}^\circ : (\mathsf{N}^\circ \multimap \alpha) \multimap (\alpha \multimap \alpha)}}{\vdash \mathsf{0}^\circ : \forall\alpha.(\mathsf{N}^\circ \multimap \alpha) \multimap (\alpha \multimap \alpha)}}{\vdash \mathsf{0}^\circ : \mathsf{N}^\circ} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{;x:\mathsf{N}^\circ \multimap \alpha, y:\alpha \vdash x:\mathsf{N}^\circ \multimap \alpha} \quad \vdash \mathsf{n}^\circ : \mathsf{N}^\circ}{;x:\mathsf{N}^\circ \multimap \alpha, y:\alpha \vdash x\mathsf{n}^\circ : \alpha}}{\vdash (\mathsf{n}+1)^\circ : (\mathsf{N}^\circ \multimap \alpha) \multimap (\alpha \multimap \alpha)}}{\vdash (\mathsf{n}+1)^\circ : \forall\alpha.(\mathsf{N}^\circ \multimap \alpha) \multimap (\alpha \multimap \alpha)}}{\vdash (\mathsf{n}+1)^\circ : \mathsf{N}^\circ}$$

For the other direction, we proceed by induction on the size of t. Suppose that $\vdash t : \mathsf{N}^\circ$. Since t is in β-normal form, the last part of the derivation must be necessarily of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{}{;x:\mathsf{N}^\circ \multimap \alpha, y:\alpha \vdash t_0 : \alpha}}{\vdash \lambda xy.t_0 : (\mathsf{N}^\circ \multimap \alpha) \multimap (\alpha \multimap \alpha)}}{\vdash \lambda xy.t_0 : \forall\alpha.(\mathsf{N}^\circ \multimap \alpha) \multimap (\alpha \multimap \alpha)}}{\vdash \lambda xy.t_0 : \mathsf{N}^\circ}$$

and $t = \lambda xy.t_0$. Since t_0 is no more an abstraction, it must be either y or of the form xt_1 with $\vdash t_1 : \mathsf{N}^\circ$. In the former case we have $t = \mathsf{0}^\circ$, while in the latter case we may apply the induction hypothesis to obtain $t_1 = \mathsf{n}^\circ$ for some n. Hence $t = \lambda xy.x\mathsf{n}^\circ = (\mathsf{n}+1)^\circ$. □

Let us come back to Figure 2. As usual, Church numerals $\mathsf{n}^\bullet, \mathsf{m}^\bullet$ can be multiplied by composition $\mathsf{n}^\bullet \circ \mathsf{m}^\bullet = \lambda f.\mathsf{n}^\bullet(\mathsf{m}^\bullet f)$. This can be repeated arbitrary many but fixed times, so we naturally obtain terms $\mathsf{mult}^\bullet$ and $\mathsf{mon}^\bullet_n$ representing multiplication and monomial $x \mapsto x^n$ of degree n. On the other hand, it is not possible to encode exponentiation, since it requires of instantiation of α with a non-linear formula such as $\mathsf{N}^\bullet$, that is not allowed in $\mathbf{DIAL}_{lin}$.

Turning on to the Scott numerals and words, observe that they are affine linear, and admit constant time successor succ° and predecessor pred° in contrast to Church.

Every finite set of cardinality n can be represented by B°_n, and the tensor product of two linear formulas by $L \otimes M$. These allow us to *linearly* represent the decomposer dec°, which works as follows: $\mathsf{dec}^\circ(\mathsf{iw}^\circ) = \mathsf{b}^\circ_i \otimes \mathsf{w}^\circ$ for $i \in \{0,1\}$ and $\mathsf{dec}^\circ(\varepsilon^\circ) = \mathsf{b}^\circ_2 \otimes \varepsilon^\circ$.

Given these building blocks, it is routine to encode the transition function of a Turing machine by a term of *linear* type $L \multimap L$. It can then be iterated by means of $\mathsf{iter}^\bullet : \mathsf{N}^\bullet \Rightarrow (L \multimap L) \Rightarrow (L \multimap L)$. Combining it with $\mathsf{mon}^\bullet_n$ and other "administrative" operations, we obtain an encoding of arbitrary polynomial time Turing machines.

Theorem 2.5 (**FP**-completeness)**.** *For every polynomial time function $f : \{0,1\}^* \to \{0,1\}^*$, there exists a λ-term t_f of type $\mathsf{W}^\bullet \Rightarrow \mathsf{W}^\circ$ in $\mathbf{DIAL}_{lin}$. Given $w \in \{0,1\}^*$, we have $[\![t_f \mathsf{w}^\bullet]\!]_\beta = \mathsf{f(w)}^\circ$.*

A couple of remarks are in order.

- Both Church and Scott numerals/words can be generalized to lists, trees and their combinations. It is indeed an advantage of the polymorphic setting that there is a generic means to build various data types. Moreover, we may consider for instance the Church lists of Scott numerals.
- In view of the fact that our system is derived from $\mathbf{1}\lambda^{\mathbf{p}}(\mathbf{W})$ of [18], one may wonder whether it is possible to give a direct translation of $\mathbf{1}\lambda^{\mathbf{p}}(\mathbf{W})$ into $\mathbf{DIAL}_{lin}$ for proving **FP**-completeness. It is, however, not straightforward because $\mathbf{1}\lambda^{\mathbf{p}}(\mathbf{W})$ is not sensitive to the distinction between linear and non-linear arrows, that is crucial for our system. In particular, our Church numerals only allow iteration of linear functions $L \multimap L$, while the Church numerals of $\mathbf{1}\lambda^{\mathbf{p}}(\mathbf{W})$ allow iteration of non-linear functions as well.

Church numerals and words :

$$\begin{array}{rclrcl}
\mathsf{N}^\bullet & \equiv & \forall\alpha(\alpha\multimap\alpha)\Rightarrow(\alpha\multimap\alpha) & \mathsf{W}^\bullet & \equiv & \forall\alpha(\alpha\multimap\alpha)\Rightarrow(\alpha\multimap\alpha)\Rightarrow(\alpha\multimap\alpha)\\
\mathsf{n}^\bullet & = & \lambda f x.\underbrace{f(\ldots f}_{n\ times}(x)\ldots) & \mathsf{w}^\bullet & = & \lambda f_0.\lambda f_1.\lambda x.f_{i_1}(f_{i_2}(\ldots(f_{i_n}(x)\ldots)))\\
\mathsf{mult}^\bullet & \equiv & \lambda xy\lambda f.x(yf):\mathsf{N}^\bullet\Rightarrow\mathsf{N}^\bullet\Rightarrow\mathsf{N}^\bullet & \mathsf{mon}^\bullet_n & \equiv & \lambda x\lambda f.\underbrace{x(\cdots(x}_{n\ times}f)\cdots):\mathsf{N}^\bullet\Rightarrow\mathsf{N}^\bullet
\end{array}$$

Scott numerals and words :

$$\begin{array}{rclrcl}
\mathsf{N}^\circ & \equiv & \mu\beta\forall\alpha(\beta\multimap\alpha)\multimap(\alpha\multimap\alpha) & \mathsf{W}^\circ & \equiv & \mu\beta\forall\alpha(\beta\multimap\alpha)\multimap(\beta\multimap\alpha)\multimap(\alpha\multimap\alpha)\\
0^\circ & = & \lambda xy.y & \varepsilon^\circ & = & \lambda xyz.z\\
(\mathsf{n}+1)^\circ & = & \lambda xy.x(\mathsf{n}^\circ) & (0\mathsf{w})^\circ & = & \lambda xyz.x(\mathsf{w}^\circ)\\
 & & & (1\mathsf{w})^\circ & = & \lambda xyz.y(\mathsf{w}^\circ)\\
\mathsf{succ}^\circ & = & \lambda z.\lambda xy.xz:\mathsf{N}^\circ\multimap\mathsf{N}^\circ & \mathsf{pred}^\circ & = & \lambda z.z(\lambda x.x)(0^\circ):\mathsf{N}^\circ\multimap\mathsf{N}^\circ
\end{array}$$

Finite sets and tensor product :

$$\begin{array}{rclrcl}
\mathsf{B}^\circ_n & \equiv & \forall\alpha.\underbrace{\alpha\multimap\ldots\alpha\multimap}_{n\ times}\alpha & L\otimes M & \equiv & \forall\alpha.(L\multimap M\multimap\alpha)\multimap\alpha\\
\mathsf{b}^\circ_i & \equiv & \lambda x_0\cdots x_{n-1}.x_i & t\otimes u & \equiv & \lambda x.xtu\quad(t:L,\ u:M)
\end{array}$$

Decomposer and iteration :

$$\begin{array}{rcl}
\mathsf{dec}^\circ & = & \lambda z.z(\lambda y.\mathsf{b}^\circ_0\otimes y)(\lambda y.\mathsf{b}^\circ_1\otimes y)(\mathsf{b}^\circ_2\otimes\varepsilon^\circ)\ :\ \mathsf{W}^\circ\multimap\mathsf{B}^\circ_3\otimes\mathsf{W}^\circ\\
\mathsf{iter}^\bullet & = & \lambda xfg.xfg\ :\ \mathsf{N}^\bullet\Rightarrow(L\multimap L)\Rightarrow(L\multimap L)
\end{array}$$

Figure 2: Basic encodings

The rest of this paper is concerned with the converse of Theorem 2.5. Namely, we prove:

Theorem 2.6 (**FP**-soundness)**.** *For every λ-term t of type $\mathsf{W}^\bullet\Rightarrow\mathsf{W}^\circ$, the associated function $f_t:\{0,1\}^*\to\{0,1\}^*$ defined by $f_t(w_1)=w_2\Leftrightarrow[\![t\mathsf{w}_1{}^\bullet]\!]_\beta=\mathsf{w}_2{}^\circ$ is a polynomial time function.*

Altogether, these two theorems ensure that the terms of type $\mathsf{W}^\bullet\Rightarrow\mathsf{W}^\circ$ in $\mathbf{DIAL}_{lin}$ precisely capture the class **FP** of polynomial time functions.

3 Resource sensitive realizability

We now develop a resource sensitive realizability semantics for $\mathbf{DIAL}_{lin}$ inspired by [8]. It concerns with the realizability relation $t,p\Vdash_\eta A$, where A is a formula to be realized, η is a valuation of propositional variables, and t is a λ-term, called a *realizer*, that embodies the computational content of a given proof. The second component p is a higher order (additive) polynomial, called a *majorizer*, that imposes a resource bound on t. Since we do not intend our model to be categorical, we do not include the denotation of t in the realizability relation (in contrast to the length space of [8]).

We then show the adequacy theorem, ensuring that $\mathbf{DIAL}_{lin}$ is sound with respect to the realizability semantics.

3.1 Higher order polynomials

We begin with the description of majorizers, namely higher order polynomials. Actually they are just monotone additive terms (without multiplication), but we nevertheless call them polynomials, since they will indeed serve as polynomials bounding the runtime of realizers (see Theorem 4.2). Using higher

order polynomials rather than first order ones will allow us to capture the difference between linear and non-linear formulas.

Definition 3.1 (Higher order polynomials)**.** We consider simple types $\sigma, \tau, \ldots$ defined by $\sigma ::= o \mid \tau \to \tau$, where o is the only base type. A *higher order polynomial* p is a λ-term built from constants $n : o$ (for every natural number n) and $+ : o \to o \to o$. More precisely, given a set $V(\sigma)$ of variables for each simple type σ, they are built as follows:

$$\frac{x \in V(\sigma)}{x : \sigma} \qquad \frac{p : \sigma \to \tau \quad q : \sigma}{pq : \tau} \qquad \frac{x \in V(\sigma) \quad p : \tau}{\lambda x.p : \sigma \to \tau} \qquad \frac{n \in \mathbb{N}}{n : o} \qquad \frac{}{+ : o \to o \to o}$$

We denote by Π the set of closed higher order polynomials.

The role of higher order polynomials is to impose a static, quantitative bound on realizers. Hence we identify them by $\alpha\beta\eta$-equivalence and natural arithmetical equivalences. For instance, we identify $x+y = y+x$ and $2+3 = 5$. We often write $p(q_1, \ldots, q_n)$ for $pq_1 \cdots q_n$. If $p : o$ and $c \in \mathbb{N}$, we write cp for $p + \cdots + p$ (c times).

We extend addition to higher-order terms so that one can sum up two terms at least when one of the summands is of base type o. Formally, let $\tau = \tau_1 \to ... \to \tau_k \to o$ and $p : \tau$. If $q : o$, we denote by $p+q$ the term $\lambda x_1 \cdots x_k.(p(x_1, ..., x_k) + q)$.

We also define a lowering operator which brings a higher order term down to a base type one. It will allow majorizers of higher order type to bound concrete resources such as time and size.

$$\begin{array}{rcll} 0_\tau & = & \lambda x_1 \cdots x_k.0, & \text{where } \tau = \tau_1 \to ... \to \tau_k \to o; \\ \downarrow p & = & p0_{\tau_1} \cdots 0_{\tau_k}, & \text{where } p : \tau = \tau_1 \to ... \to \tau_k \to o. \end{array}$$

Observe that $\downarrow p$ is a natural number if p is a closed higher order polynomial. Notice also that $\downarrow p = p$ if $p : o$.

Formulas of $\mathbf{DIAL}_{lin}$ are mapped to types of higher order polynomials as follows:

$$o(L) = o, \qquad o(L \multimap A) = o(A), \qquad o(A \Rightarrow B) = o(A) \to o(B), \qquad o(\forall \alpha A) = o(A).$$

Thus all linear formulas collapse to o, while non-linear formulas retain the structure given by non-linear arrows.

Remark 3.2. Consider $\mathscr{M} = (\Pi, +, \leq, D)$ where $p_1 + p_2$ is a partial operation defined only when one of the p_i is of type o, $p \leq q$ iff $\downarrow p \leq \downarrow q$ and $D(p,q) = \downarrow q - \downarrow p$. Then $\mathscr{M}$ gives rise to a *partial resource monoid*, namely a partial monoid that satisfies all the axioms of resource monoids given by [8].

It would be desirable to have a *total* resource monoid so that the basic results of [8] would be reused for our purpose. However, we have no idea how to do that coherently. This problem is related to the above mentioned restriction on $\mathbf{DIAL}_{lin}$ that the premise of a linear implication must be a linear formula.

3.2 Realizability relation

We are now ready to introduce the realizability relation. Intuitively, $t, p \Vdash A$ signifies that A is the specification of t and p majorizes the potential cost for evaluating t when it is applied to some arguments.

Let us begin with some notations.

- $\overline{x}$, $\overline{t}$, $\overline{A}$ stand for (possibly empty) lists of variables, terms and formulas, respectively.
- $t\langle u_1/x_1, \ldots, u_n/x_n\rangle$ denotes the term $(\lambda x_1 \cdots x_n.t)u_1 \cdots u_n$.

- θ, ξ stand for lists of binding expressions; for instance, $\theta = u_1/x_1, \ldots, u_n/x_n$ with $x_1, \ldots, x_n$ distinct. This allows us to concisely write $t\langle\theta\rangle$ for $t\langle u_1/x_1, \ldots, u_n/x_n\rangle = (\lambda x_1 \cdots x_n.t)u_1 \cdots u_n$.

Definition 3.3. (Saturated sets) Let τ be a type for higher order polynomials. A nonempty set $X \subseteq \Lambda \times \Pi$ is a *saturated set of type* τ if whenever $(t, p) \in X$, we have $t \Downarrow$, p is a closed higher order polynomial of type τ and the following hold:

(bound) $TS(t) \leq_{\downarrow} p$.

(monotonicity) $(t,\ p+n) \in X$ for every $n \in \mathbb{N}$.

(exchange) If $t = t_0\langle\theta, v_1/y_1, v_2/y_2, \xi\rangle\overline{u}$, then $(t_0\langle\theta, v_2/y_2, v_1/y_1, \xi\rangle\overline{u},\ p) \in X$.

(weakening) If $t = t_0\langle\theta\rangle\overline{u}$, $z \notin FV(t_0)$ and $w \Downarrow$, then $(t_0\langle\theta, w/z\rangle\overline{u},\ p + TS(w) + 2) \in X$.

(contraction) If $t = t_0\langle\theta, w/z_1, w/z_2\rangle\overline{u}$, then $(t_0[z/z_1, z/z_2]\langle\theta, w/z\rangle\overline{u},\ p) \in X$.

(concatenation) If $t = (t_0\langle\theta\rangle)(t_1\langle\xi\rangle)\overline{u}$, then $((t_0t_1)\langle\theta, \xi\rangle\overline{u},\ p) \in X$.

(identity) If $t = t_0\overline{u}$, then $((x\langle t_0/x\rangle)\overline{u},\ p+3) \in X$.

By Lemma 2.3 (size), condition (bound) implies that $|[\![t]\!]| \leq_{\downarrow} p$. Note that condition (weakening) asks for an additional cost $TS(w) + 2$. This is due to our computational model: weak call-by-value reduction $(\lambda x.t)w \to t[w/x]$ requires that w is a value, even when $x \notin FV(t)$.

We have to show that there exists at least one saturated set. The following proposition gives the canonical one.

Proposition 3.4. *$X_0 = \{(t, n) : t \Downarrow$ and $TS(t) \leq n\}$ is the greatest saturated set of type o.*

Proof. Conditions (bound) and (monotonicity) hold by definition. The other conditions follow from Lemma 2.3. X_0 is obviously greatest. □

A *valuation* η maps each propositional variable α to a saturated set $\eta(\alpha)$ of type o. $\eta\{\alpha \leftarrow X\}$ stands for a valuation which agrees with η except that it assigns X to α.

Definition 3.5. (Realizability) We define the relation $t, p \Vdash_\eta A$, where $t \in \Lambda$ (called *realizer*), p is a closed higher order polynomial of type $o(A)$ (called *majorizer*) and η is a valuation. It induces the set $\hat{A}_\eta = \{(t, p) : t, p \Vdash_\eta A\}$. The definition proceeds by induction on A.

- $t, n \Vdash_\eta \alpha$ iff $(t, n) \in \eta(\alpha)$.
- $t,\ p \Vdash_\eta L \multimap A$ iff $TS(t) \leq_{\downarrow} p$ and $u, m \Vdash_\eta L$ implies $tu, p + m \Vdash_\eta A$ for every u, m.
- $t,\ p \Vdash_\eta B \Rightarrow A$ iff $TS(t) \leq_{\downarrow} p$ and $u, q \Vdash_\eta B$ implies $tu, p(q) \Vdash_\eta A$ for every u, q.
- $t,\ p \Vdash_\eta \forall\alpha A$ iff $t,\ p \Vdash_{\eta\{\alpha \leftarrow X\}} A$ for every saturated set X of type o.
- $t,\ p \Vdash_\eta \mu\alpha L$ iff $(t, p) \in X$ for every saturated set X of type o such that $\hat{L}_{\eta\{\alpha \leftarrow X\}} \subseteq X$.

Lemma 3.6.

1. *For every formula A, $\hat{A}_\eta = \{(t, p) : t, p \Vdash_\eta A\}$ is a saturated set of type $o(A)$.*
2. *For every A and L, we have $t, p \Vdash_\eta A[L/\beta]$ iff $t, p \Vdash_{\eta\{\beta \leftarrow \hat{L}_\eta\}} A$.*
3. *If $t, p \Vdash_\eta \forall\alpha A$, then $t, p \Vdash_\eta A[L/\alpha]$ for every linear formula L.*
4. *$\mu\hat{\alpha}L_\eta$ is the least fixpoint of $f(X) = \hat{L}_{\eta\{\alpha \leftarrow X\}}$.*
5. *$t, p \Vdash_\eta \mu\alpha L$ iff $t, p \Vdash_\eta L[\mu\alpha L/\alpha]$.*

Proof. 1 and 2. By induction on A, noting that any non-empty intersection of saturated sets is again saturated.
3. By 1 and 2.
4. Notice that f is a monotone function since α occurs only positively in L. Call a saturated set X of type o a *prefixpoint* of f if $f(X) \subseteq X$. Then $\mu\hat{\alpha}L_\eta$ is the infimum of all prefixpoints of f. So $\mu\hat{\alpha}L_\eta \subseteq X$ for every prefixpoint X, and by monotonicity $f(\mu\hat{\alpha}L_\eta) \subseteq f(X) \subseteq X$. Since $\mu\hat{\alpha}L_\eta$ is the infimum of all such X's, we obtain $f(\mu\hat{\alpha}L_\eta) \subseteq \mu\hat{\alpha}L_\eta$. Applying monotonicity again, we get $f(f(\mu\hat{\alpha}L_\eta)) \subseteq f(\mu\hat{\alpha}L_\eta)$, so $f(\mu\hat{\alpha}L_\eta)$ is a prefixpoint of f. Hence $\mu\hat{\alpha}L_\eta \subseteq f(\mu\hat{\alpha}L_\eta)$.

5. By 2 and 4, $t, p \Vdash_\eta \mu\alpha L$ iff $(t,p) \in \mu\hat{\alpha}L_\eta$ iff $(t,p) \in f(\mu\hat{\alpha}L_\eta)$ iff $t, p \Vdash_{\eta\{\alpha \leftarrow \mu\hat{\alpha}L_\eta\}} L$ iff $t, p \Vdash_\eta L[\mu\alpha L/\alpha]$. □

3.3 Adequacy theorem

The adequacy theorem is the crux of this paper. It states that **DIAL**$_{lin}$ is sound with respect to the realizability semantics we have introduced.

Theorem 3.7 (Adequacy)**.** *Suppose that $\overline{x} : \overline{C}; \overline{y} : \overline{M} \vdash t : A$ is derivable. Then there exists a higher order polynomial $p(\overline{x}) : o(A)$ with variables $\overline{x}$ of type $o(\overline{C})$ such that for any valuation η we have the following:*

$$\overline{u}, \overline{q} \Vdash_\eta \overline{C},\ \overline{s}, \overline{m} \Vdash_\eta \overline{M} \Longrightarrow t\langle \overline{u}/\overline{x}, \overline{s}/\overline{y} \rangle, p(\overline{q}) + \overline{m} \Vdash_\eta A.$$

Moreover, if $\overline{x} = x_1, \ldots, x_a$ and each x_i occurs c_i times in t, then

$$(*) \quad |t| + c_1 \downarrow q_1 + \cdots + c_a \downarrow q_a \leq \downarrow p(\overline{q}).$$

We call the above $p(\overline{x})$ a *majorizer* of $\overline{x} : \overline{C}; \overline{y} : \overline{M} \vdash t : A$.

Proof. By induction on the length of the derivation. We omit the cases for $\forall$ and μ, since they easily follow from Lemma 3.6. Accordingly, we do not specify the valuation η, simply writing $\Vdash$ for $\Vdash_\eta$. We distinguish the last inference rule of the derivation.

Case (ax1): For $x : A; \vdash x : A$, take $p(x) = x + 3$ as the majorizer. Condition (*) obviously holds.

If $u, q \Vdash A$, then condition (identity) for saturated sets implies $x\langle u/x \rangle, q + 3 \Vdash A$, namely $x\langle u/x \rangle, p(q) \Vdash A$.

Case (ax2): For $; y : L \vdash y : L$, take $p = 3$ as the majorizer.

Case ($\multimap_e$):
$$\frac{\overline{x}_1 : \overline{C}_1; \overline{y}_1 : \overline{M}_1 \vdash t_1 : L \multimap A \quad \overline{x}_2 : \overline{C}_2; \overline{y}_2 : \overline{M}_2 \vdash t_2 : L}{\overline{x}_1 : \overline{C}_1, \overline{x}_2 : \overline{C}_2; \overline{y}_1 : \overline{M}_1, \overline{y}_2 : \overline{M}_2 \vdash t_1 t_2 : A}$$

By the induction hypothesis, we have majorizers $p_1(\overline{x}_1) : o(L \multimap A)$ and $p_2(\overline{x}_2) : o(L)$ of the left and right premises, respectively. We claim that $p(\overline{x}_1, \overline{x}_2) = p_1(\overline{x}_1) + p_2(\overline{x}_2)$ is the suitable majorizer of the conclusion. Notice that $p_2(\overline{x}_2)$ is of type o, so that the addition is well defined. Condition (*) follows by the induction hypothesis.

Suppose that $\overline{u}_i, \overline{q}_i \Vdash \overline{C}_i$ and $\overline{s}_i, \overline{m}_i \Vdash \overline{M}_i$ for $i = 1, 2$ and write θ_i for the list $\overline{u}_i/\overline{x}_i, \overline{s}_i/\overline{y}_i$.

Then the induction hypothesis yields $t_1\langle \theta_1 \rangle, p_1(\overline{q}_1) + \overline{m}_1 \Vdash L \multimap A$ and $t_2\langle \theta_2 \rangle, p_2(\overline{q}_2) + \overline{m}_2 \Vdash L$. Hence by the definition of realizability, $t_1\langle \theta_1 \rangle t_2\langle \theta_2 \rangle, p_1(\overline{q}_1) + p_2(\overline{q}_2) + \overline{m}_1 + \overline{m}_2 \Vdash A$, so by conditions (concatenation) and (exchange), $(t_1 t_2)[\overline{u}_1/\overline{x}_1, \overline{u}_2/\overline{x}_2, \overline{s}_1/\overline{y}_1, \overline{s}_2/\overline{y}_2], p(\overline{q}_1, \overline{q}_2) + \overline{m}_1 + \overline{m}_2 \Vdash A$ as required.

Case ($\Rightarrow_e$):
$$\frac{\overline{x}_1 : \overline{C}_1; \overline{y} : \overline{M} \vdash t_1 : B \Rightarrow A \quad \overline{x}_2 : \overline{C}_2; \vdash t_2 : B}{\overline{x}_1 : \overline{C}_1, \overline{x}_2 : \overline{C}_2; \overline{y} : \overline{M} \vdash t_1 t_2 : A}$$

By the induction hypothesis, we have majorizers $\lambda z.p_1(\overline{x}_1,z) : o(B) \to o(A)$ and $p_2(\overline{x}_2) : o(B)$ of the left and right premises, respectively. We claim that $p(\overline{x}_1,\overline{x}_2) = p_1(\overline{x}_1, p_2(\overline{x}_2))$ is the suitable majorizer of the conclusion. As before, condition (*) holds.

Suppose that $\overline{u}_i,\overline{q}_i \Vdash \overline{C}_i$ for $i = 1,2$, $\overline{s},\overline{m} \Vdash \overline{M}$, and write $\theta_1 = \overline{u}_1/\overline{x}_1,\overline{s}/\overline{y}$ and $\theta_2 = \overline{u}_2/\overline{x}_2$. Then the induction hypothesis yields $t_1\langle\theta_1\rangle, \lambda z.p_1(\overline{q}_1,z)+\overline{m} \Vdash B \Rightarrow A$ and $t_2\langle\theta_2\rangle, p_2(\overline{q}_2) \Vdash B$. Hence $t_1\langle\theta_1\rangle t_2\langle\theta_2\rangle$, $p_1(\overline{q}_1, p_2(\overline{q}_2)) + \overline{m} \Vdash A$, so by conditions (concatenation) and (exchange), $(t_1t_2)\langle\overline{u}_1/\overline{x}_1,\overline{u}_2/\overline{x}_2,\overline{s}/\overline{y}\rangle)$, $p(\overline{q}_1,\overline{q}_2)+\overline{m} \Vdash A$ as required.

Case ($\multimap_i$): $$\frac{\overline{x} : \overline{C};\overline{y} : \overline{M}, z : L \vdash t : A}{\overline{x} : \overline{C};\overline{y} : \overline{M} \vdash \lambda z.t : L \multimap A}$$

By the induction hypothesis, we have a majorizer $p_0(\overline{x}) : o(A)$ of the premise. We claim that $p(\overline{x}) = p_0(\overline{x}) + d$ with constant d specified below is the suitable majorizer of the conclusion. Condition (*) holds if $d \geq 1$.

Suppose that $\overline{u},\overline{q} \Vdash \overline{C}$, $\overline{s},\overline{m} \Vdash \overline{M}$ and write $\theta = \overline{u}/\overline{x},\overline{s}/\overline{y}$. Then whenever $w,k \Vdash L$, the induction hypothesis gives us $t\langle\theta,w/z\rangle, p_0(\overline{q})+\overline{m}+k \Vdash A$. By (monotonicity), $t\langle\theta,w/z\rangle, p(\overline{q})+\overline{m}+k \Vdash A$, namely, $w,k \Vdash L$ implies $(\lambda z.t)\langle\theta\rangle w,\ p(\overline{q})+\overline{m}+k \Vdash A$.

Hence it just remains to verify that $((\lambda z.t)\langle\theta\rangle,\ p(\overline{q})+\overline{m})$ satisfies condition (bound). Suppose that $\overline{x} = x_1,\ldots,x_a$ and each x_i occurs at most c_i times in t. We assume that $c_i \geq 2$ for $i = 1,\ldots,a$; the case $c_i = 0,1$ can be easily treated by choosing d large enough. We have

$$(\lambda z.t)\langle\theta\rangle = (\lambda\overline{xy}z.t)\overline{us} \overset{n_1}{\twoheadrightarrow} (\lambda\overline{xy}z.t)[\![\overline{u}]\!][\![\overline{s}]\!] \overset{n_2}{\twoheadrightarrow} (\lambda\overline{y}z.t[[\![\overline{u}]\!]/\overline{x}])[\![\overline{s}]\!] \overset{n_3}{\twoheadrightarrow} \lambda z.t[[\![\overline{u}]\!]/\overline{x},[\![\overline{s}]\!]/\overline{y}],$$

where $n_1 = Time(\overline{u}) + Time(\overline{s})$, $n_3 \leq d_1 =$ the length of the list $\overline{y}$, and $n_2 = |\lambda\overline{y}z.t[[\![\overline{u}]\!]/\overline{x}]| - |(\lambda\overline{xy}z.t)[\![\overline{u}]\!]|$ $\leq (c_1-1)|[\![u_1]\!]| + \cdots + (c_a-1)|[\![u_a]\!]| \leq (c_1-1)TS(u_1)+\cdots+(c_a-1)TS(u_a)$ by Lemmas 2.1 and 2.3(1). Hence

$$\begin{aligned} TS((\lambda z.t)\langle\theta\rangle) &= Time((\lambda\overline{xy}z.t)\overline{us}) + |(\lambda\overline{xy}z.t)\overline{us}| \\ &\leq Time(\overline{u}) + Time(\overline{s}) + (c_1-1)TS(u_1)+\cdots+(c_a-1)TS(u_a) + d_1 + |t| + |\overline{u}| + |\overline{s}| + d_2 \\ &= |t| + c_1TS(u_1)+\cdots+c_aTS(u_a) + TS(\overline{s}) + d_1 + d_2, \end{aligned}$$

where d_2 is the length of $\overline{xy}z$. Because of $TS(\overline{u}) \leq\downarrow \overline{q}$, $TS(\overline{s}) \leq \overline{m}$ and condition (*), we obtain $TS((\lambda z.t)\langle\theta\rangle)$ $\leq p(\overline{q})+\overline{m}$ by letting $d = d_1 + d_2$. We therefore conclude $(\lambda z.t)\langle\theta\rangle,\ p(\overline{q})+\overline{m} \Vdash L \multimap A$.

Case ($\Rightarrow_i$): $$\frac{\overline{x} : \overline{C}, z : B;\overline{y} : \overline{M} \vdash t : A}{\overline{x} : \overline{C};\overline{y} : \overline{M} \vdash \lambda z.t : B \Rightarrow A}$$

By the induction hypothesis, we have a majorizer $p_0(\overline{x},z) : o(A)$ of the premise. We claim that $p(\overline{x}) = \lambda z.p_0(\overline{x},z) + d : o(B) \to o(A)$ with d a large enough constant is the suitable majorizer of the conclusion. The proof is just the same as above.

Case (*Contr*): $$\frac{z_1 : B, z_2 : B, \overline{x} : \overline{C};\overline{y} : \overline{M} \vdash t : A}{z : B, \overline{x} : \overline{C};\overline{y} : \overline{M} \vdash t[z/z_1, z/z_2] : A}$$

By the induction hypothesis, we have a majorizer $p_0(z_1,z_2,\overline{x}) : o(A)$ of the premise. We can prove that $p(z,\overline{x}) = p_0(z,z,\overline{x})$ is the suitable majorizer of the conclusion by using condition (contraction).

Case (*Weak*): $$\frac{\overline{x} : \overline{C};\overline{y} : \overline{M} \vdash t : A}{\overline{x} : \overline{C};\overline{y} : \overline{M}, z : L \vdash t : A}$$

By the induction hypothesis, we have a majorizer $p_0(\overline{x}) : o(A)$ of the premise. We can prove that $p_0(\overline{x})+2 : o(A)$ works for the conclusion by using condition (weakening).

Case (*Derel*): $\dfrac{\overline{x} : \overline{C}; \overline{y} : \overline{M}, z : L \vdash t : A}{\overline{x} : \overline{C}, z : L; \overline{y} : \overline{M} \vdash t : A}$

By the induction hypothesis, we have a majorizer $p_0(\overline{x}) : o(A)$ of the premise. Then it is easy to see that $p(\overline{x}, z) = p_0(\overline{x}) + z$ works as a majorizer of the conclusion. □

4 Polynomial time soundness

In this section, we apply the adequacy theorem to prove that every term t of type $\mathsf{W}^\bullet \Rightarrow L$ (with L a Scott data type) represents a polynomial time function. There is, however, a technical problem due to the use of the weak call-by-value strategy. Since it does not reduce under λ, if t is of the form $t = \lambda xy.t'$, then the evaluation of $t\,\mathsf{w}^\bullet$ gets stuck after the first reduction.

The problem can be settled by a little trick when L is fixpoint-free, eg., $L = \mathsf{B}_2^\circ$ (Subsection 4.1). However, the case when $L = \mathsf{W}^\circ$ is not so easy. The main difficulty is that, although each *bit* of the output Scott word can be computed in polynomial time, its *length* is not yet ensured to be polynomial. The length cannot be detected by weak call-by-value; it rather depends on the size of the β-normal form. We are thus compelled to develop another realizability argument based on the β-normal form, which indeed ensures that the output is of polynomial length (Subsection 4.2). We will then be able to prove the polynomial time soundness for the Scott words (Subsection 4.3).

4.1 Polynomial time soundness for predicates

We first observe that Church numerals and words are bounded by linear majorizers.

Lemma 4.1.

1. *For every $n \in \mathbb{N}$, we have $\mathsf{n}^\bullet, p_n \Vdash \mathsf{N}^\bullet$ with $p_n = \lambda z.n(z+3)+3 : o \to o$.*
2. *For every $w \in \{0,1\}^n$, we have $\mathsf{w}^\bullet, q_n \Vdash \mathsf{W}^\bullet$ with $q_n = \lambda z_0 z_1.n(z_0+z_1+3)+3 : o \to o \to o$.*

Proof. Since both are similar, we only prove the statement 1. We assume $n \geq 1$, since the case $n = 0$ is easy. Let η be a valuation, $u, m \Vdash_\eta \alpha \multimap \alpha$ and $v, k \Vdash_\eta \alpha$.

By condition (identity), we have $x\langle v/x\rangle, k+3 \Vdash_\eta \alpha$ and $f_i\langle u/f_i\rangle, m+3 \Vdash_\eta \alpha \multimap \alpha$ for any variable f_i. So $f_1\langle u/f_1\rangle(\cdots(f_n\langle u/f_n\rangle x\langle v/x\rangle)\cdots), n(m+3)+k+3 \Vdash_\eta \alpha$. By (concatenation) and (contraction), $f^n x\langle u/f, v/x\rangle, n(m+3)+k+3 \Vdash_\eta \alpha$. By noting that $f^n x\langle u/f, v/x\rangle = (\lambda fx.f^n x)uv$, we obtain $\lambda fx.f^n x$, $\lambda z.n(z+3)+3 \Vdash_\eta (\alpha \multimap \alpha) \Rightarrow (\alpha \multimap \alpha)$. □

These linear majorizers are turned into polynomial ones when applied to majorizers of higher order type. As a consequence, we obtain a polynomial bound on the execution time.

Theorem 4.2 (Weak soundness)**.** *Let L be a linear formula. If $\vdash t : \mathsf{W}^\bullet \Rightarrow L$, then there exists a polynomial P such that for every $w \in \{0,1\}^*$, $\mathit{Time}(t\mathsf{w}^\bullet) \leq P(|w|)$.*

Proof. By the adequacy theorem, we have a majorizer $\lambda x.p(x) : o(\mathsf{W}^\circ \Rightarrow L) = (o \to o \to o) \to o$ such that $t, \lambda x.p(x) \Vdash \mathsf{W}^\bullet \Rightarrow L$. Let $w \in \{0,1\}^n$. By the lemma above, we have $\mathsf{w}^\bullet, q_n \Vdash_\eta W$. Hence by the definition of realizability, $t\mathsf{w}^\bullet, p(q_n) \Vdash L$.

We prove that $p(q_n) : o$ is a polynomial in n by induction on the structure of the term $p(x)$. We suppose that $p(x)$ is in β normal form.

If $p(x) = k$, then $p(q_n) = k$ is a constant and obviously a polynomial in n. If $p(x) = p_1(x) + p_2(x)$, then by the induction hypothesis $p_1(q_n)$ and $p_2(q_n)$ are polynomials in n, so is $p(q_n)$. Otherwise, $p(x)$ must be of the form $xp_1(x)p_2(x)$ since x is the only free variable and of type $o \to o \to o$. By the induction hypothesis, $p_1(q_n)$ and $p_2(q_n)$ are polynomials in n. So $p(q_n) = q_n(p_1(q_n), p_2(q_n)) = n(p_1(q_n) + p_2(q_n) + 3) + 3$ is still a polynomial in n.

By condition (bound), we conclude that $Time(t\mathsf{w}^\bullet)$ is bounded by $p(q_n)$, a polynomial in n. □

This in particular implies that every term of type $\mathsf{W}^\bullet \Rightarrow \mathsf{B}_2^\circ$ represents a polynomial time predicate.

Corollary 4.3 (**P**-soundness for predicates)**.** *If $t : \mathsf{W}^\bullet \Rightarrow \mathsf{B}_2^\circ$, then the predicate $f_t : \{0,1\}^* \to \{0,1\}$ defined by $f_t(w) = 1 \Leftrightarrow [\![t\mathsf{w}^\bullet]\!]_\beta = \mathsf{b}_1{}^\circ$ is a polynomial time predicate.*

Proof. Observe that $\lambda x.tx\mathsf{b}_0^\circ\mathsf{b}_1^\circ : \mathsf{W}^\bullet \Rightarrow \mathsf{B}_2^\circ$ and for every $w \in \{0,1\}^*$ the term $(\lambda x.tx\mathsf{b}_0^\circ\mathsf{b}_1^\circ)\mathsf{w}^\bullet$ reduces to either b_0° or b_1° by the weak call-by-value strategy (see the proof of Lemma 4.8). By the previous theorem, the runtime is bounded by a polynomial. □

4.2 Size realizability

As explained in the beginning of this section, the previous realizability semantics does not tell anything about the length of the output Scott words. We thus introduce another realizability semantics based on the (applicative) size of β-normal forms. Due to lack of space, we can only state the definitions and the result.

Let $\sharp t$ be the number of applications in t, which is more precisely defined by:

$$\sharp x = 0, \qquad \sharp(tu) = \sharp t + \sharp u + 1, \qquad \sharp\lambda x.t = \sharp t.$$

$\sharp t$ is not relevant for bounding the size of t in general (think of $t = \lambda x_1 \cdots x_{100}.x_i$; we have $\sharp t = 0$). However, when t is a Scott word, $\sharp t$ exactly corresponds to the length of the word represented by t.

Definition 4.4 (Size-saturated sets)**.** Let τ be a type for higher order polynomials. A nonempty set $X \subseteq \Lambda \times \Pi$ is *a* size-saturated set of type τ if whenever $(t,p) \in X$, t is normalizable, p is a closed higher order polynomial of type τ and the following hold:

(bound') $\sharp[\![t]\!]_\beta \leq_\downarrow p$.

(weak') if $t = t_0\langle\theta\rangle\overline{u}$ and $z \notin FV(t_0)$, then $(t_0\langle\theta, w/z\rangle\overline{u},\ p) \in X$.

(identity') if $t = t_0\overline{u}$, then $((x\langle t_0/x\rangle)\overline{u},\ p) \in X$.

We also require conditions (monotonicity), (exchange), (contraction), (concatenation) of Definition 3.5, and finally,

(variable) $(\maltese, 0_\tau) \in X$, where $\maltese$ is a fixed variable.

Condition (variable) employs a fixed variable $\maltese$ (considered as an inert object), that helps us to deal with open terms. Notice that it contradicts the previous condition (bound); that is one reason why we have to consider size realizability separately from the previous one.

As before, we have the greatest size-saturated set of type o: $X_s = \{(t,n) : \sharp[\![t]\!]_\beta \leq n\}$. A *valuation* η is now supposed to map each propositional variable to a size-saturated set of type o.

Definition 4.5 (Size realizability)**.** We define the relation $t, p \Vdash^s_\eta A$ as in Definition 3.5, except that

- $t, p \Vdash^s_\eta L \multimap A$ iff either $t = \maltese$ (and p is arbitrary), or $u, m \Vdash^s_\eta L$ implies $tu, p + m \Vdash^s_\eta A$ for every u, m.
- $t, p \Vdash^s_\eta B \Rightarrow A$ iff either $t = \maltese$, or $u, q \Vdash^s_\eta B$ implies $tu, p(q) \Vdash^s_\eta A$ for every u, q.

One can then verify that for every formula A and valuation η, the set $\hat{A}_\eta = \{(t,p) : t, p \Vdash_\eta A\}$ is a size-saturated set of type $o(A)$.

Theorem 4.6 (Size adequacy)**.** *Suppose that $\overline{x} : \overline{C}; \overline{y} : \overline{M} \vdash t : A$ is derivable. Then there exists a higher order polynomial $p(\overline{x}) : o(A)$ such that for any valuation η we have the following:*

$$\overline{u}, \overline{q} \Vdash^s_\eta \overline{C},\ \overline{v}, \overline{m} \Vdash^s_\eta \overline{M} \Longrightarrow t\langle \overline{u}/\overline{x}, \overline{v}/\overline{y}\rangle, p(\overline{q}) + \overline{m} \Vdash^s_\eta A.$$

Theorem 4.7 (Size soundness)**.** *If $\vdash t : \mathsf{W}^\bullet \Rightarrow L$, then there exists a polynomial P such that $\sharp[\![t\mathsf{w}^\bullet]\!]_\beta \leq P(|w|)$ for every $w \in \{0,1\}^*$.*

4.3 Polynomial time soundness for words

As in the proof of Corollary 4.3, we use a little trick. First note that we have a predecessor $\mathsf{p}^\circ = \lambda z.z(\lambda x.x)(\lambda x.x)(\varepsilon^\circ) : \mathsf{W}^\circ \multimap \mathsf{W}^\circ$. By employing it, we define

$$\mathsf{q}^\circ = \lambda x.x(\lambda y.\mathsf{b}^\circ_0)(\lambda y.\mathsf{b}^\circ_1)\mathsf{b}^\circ_2 : \mathsf{W}^\circ \multimap \mathsf{B}^\circ_3, \qquad \mathsf{bit}^\circ_i = \lambda x.\mathsf{q}^\circ(\underbrace{\mathsf{p}^\circ \cdots \mathsf{p}^\circ}_{i \text{ times}}(x)) : \mathsf{W}^\circ \multimap \mathsf{B}^\circ_3.$$

Lemma 4.8. *Suppose that t is a closed term of type W° and $[\![t]\!]_\beta$ represents a word $w \in \{0,1\}^n$ of length n. Then for any $i < n$, $[\![\mathsf{bit}^\circ_i(t)]\!] = \mathsf{b}^\circ_0$ or b°_1, depending on the ith bit of w. If $i \geq n$, $[\![\mathsf{bit}^\circ_i(t)]\!] = \mathsf{b}^\circ_2$.*

Proof. The crucial fact is that given a closed term u of type W°, $\mathsf{q}^\circ u$ always evaluates to b°_j for some $j \in \{0,1,2\}$ by the weak call-by-value strategy. To see this, take a fresh propositional variable γ, variables z_0, z_1, z_2, and consider $\mathsf{q}^\circ_\gamma = \lambda x.x(\lambda y.z_0)(\lambda y.z_1)z_2 : \mathsf{W}^\circ \multimap \gamma$. Since $\mathsf{q}^\circ_\gamma u$ is of type γ, so is $[\![\mathsf{q}^\circ_\gamma u]\!]$ by the subject reduction property. Hence it cannot be an abstraction. It cannot either be an application, since the only possible head variables are z_0, z_1 and z_2 of atomic type γ. Therefore $[\![\mathsf{q}^\circ_\gamma u]\!] = z_j$ for some $j \in \{0,1,2\}$. By substituting b°_j for z_j, we obtain $[\![\mathsf{q}^\circ u]\!] = \mathsf{b}^\circ_j$. Now the claim is easily verified. □

Finally we are able to prove the polynomial time soundness for words.

Theorem 2.6 (FP-soundness). *For every λ-term t of type $\mathsf{W}^\bullet \Rightarrow \mathsf{W}^\circ$, the associated function $f_t : \{0,1\}^* \to \{0,1\}^*$ defined by $f_t(w_1) = w_2 \Leftrightarrow [\![t\mathsf{w}_1{}^\bullet]\!]_\beta = \mathsf{w}_2{}^\circ$ is a polynomial time function.*

Proof. By the adequacy theorem, we have $t,\ \lambda x.p(x) \Vdash \mathsf{W}^\bullet \Rightarrow \mathsf{W}^\circ$ for some $\lambda x.p(x) : o(\mathsf{W}^\bullet) \to o$.

We also have $\mathsf{p}^\circ, k \Vdash \mathsf{W}^\circ \multimap \mathsf{W}^\circ$ and $\mathsf{q}^\circ, k' \Vdash \mathsf{W}^\circ \multimap \mathsf{B}^\circ_3$ for some constants k, k', from which we easily obtain $\lambda x.\mathsf{bit}^\circ_i(tx),\ \lambda x.p(x) + ik + k'' \vdash \mathsf{W}^\bullet \Rightarrow \mathsf{B}^\circ_3$ for some constant $k'' \geq k'$.

By inspecting the proof of Theorem 4.2 we obtain a polynomial $P(x)$ such that $\mathit{Time}(\mathsf{bit}^\circ_i(t\mathsf{w}^\bullet)) \leq P(|w|) + ik$ for every $w \in \{0,1\}^n$ and every $i \in \mathbb{N}$. Furthermore, Theorem 4.7 gives a polynomial $Q(x)$ such that $\sharp[\![t\mathsf{w}^\bullet]\!]_\beta \leq Q(|w|)$, that implies that the Scott term $[\![t\mathsf{w}^\bullet]\!]_\beta$ represents a word of length at most $Q(|w|)$.

Now the desired word $f_t(w)$ can be obtained by computing the values of $\mathsf{bit}^\circ_0(t\mathsf{w}^\bullet), \mathsf{bit}^\circ_1(t\mathsf{w}^\bullet), \mathsf{bit}^\circ_2(t\mathsf{w}^\bullet)$, ... until we obtain $[\![\mathsf{bit}^\circ_m(t\mathsf{w}^\bullet)]\!] = \mathsf{b}^\circ_2$. We know that $m \leq Q(|w|)$. Hence the overall runtime is $R(|w|)$ with $R(x) = O((P(x) + Q(x))^4 \cdot Q(x))$ in view of Theorem 2.2. □

5 Concluding remarks

Inspired by [18], we have introduced a purely logical system $\mathbf{DIAL}_{lin}$ that captures precisely the class of polynomial time functions. To prove soundness, we have introduced a simple variant of the Hofmann-Dal Lago realizability. Here is a non-exhaustive list of the remaining open questions related to this work:

- Can we, instead of using a dual type system, directly deal with the !-connective? For the time being, it seems that it would considerably complicate the definition of the realizability relation.
- We are compelled to introduce two realizability interpretations, one for bounding the runtime, and the other for bounding the length of the output. Is it possible to integrate them into one realizability interpretation?
- Is it possible to relate our definition of realizability with the original one [6] more closely? We have observed that our higher order polynomials are equipped with the structure of partial resource monoid (Remark 3.2). Our definition of realizability is also derived from their notion of length space. Establishing an exact correspondence is, however, left to the future work.
- We have adapted the tiered recursion characterization of the **PTIME** functions. Can we find a suitable logical system as well corresponding to the tiered recursion characterizations of **PSPACE** and **ALOGTIME** in [21], [20] and [22]?

References

[1] M. Abadi, L. Cardelli & G. Plotkin (1993): *Types for the Scott numerals*. *Manuscript* .

[2] A. Asperti & L. Roversi (2002): *Intuitionistic Light Affine Logic (Proof-nets, Normalization Complexity, Expressive Power, Programming Notation)*. *ACM Transactions on Computational Logic* 3(1), pp. 137 – 175.

[3] P. Baillot & K. Terui (2009): *Light types for polynomial time computation in lambda calculus*. *Information and Computation* 207(1), pp. 41–62.

[4] S. Bellantoni & S. Cook (1992): *A new recursion-theoretic characterization of the polytime functions*. *Computational Complexity* 2(2), pp. 97–110.

[5] S. Bellantoni, K.-H. Niggl & H. Schwichtenberg (2000): *Ramification, Modality and Linearity in Higher Type Recursion*. *Annals of Pure and Applied Logic* 104, pp. 17–30.

[6] U. Dal Lago & M. Hofmann (2005): *Quantitative models and implicit complexity*. In: *FSTTCS*. pp. 189–200.

[7] U. Dal Lago & M. Hofmann (2009): *Bounded Linear Logic, Revisited*. In: *TLCA*. pp. 80–94.

[8] U. Dal Lago & M. Hofmann (2009): *A Semantic Proof of Polytime Soundness for Light Affine Logic*. *Theory of Computing Systems*, to appear.

[9] U. Dal Lago & S. Martini (2008): *The weak lambda calculus as a reasonable machine*. *Theoretical Computer Science* 398(1-3), pp. 32–50.

[10] M. Gaboardi, J.-Y. Marion & S. Ronchi Della Rocca (2008): *A logical account of PSPACE*. In: *POPL*. pp. 121–131.

[11] J.-Y. Girard (1998): *Light linear logic*. *Information and Computation* 143(2), pp. 175–204.

[12] M. Hofmann (1997): *An application of category-theoretic semantics to the characterisation of complexity classes using higher-order function algebras*. *Bulletin of Symbolic Logic*, pp. 469–486.

[13] M. Hofmann (2000): *Safe recursion with higher types and BCK-algebra*. *Annals of Pure and Applied Logic* 104(1-3), pp. 113–166.

[14] M. Hofmann (2003): *Linear types and non-size-increasing polynomial time computation*. *Information and Computation* 183(1), pp. 57–85.

[15] Y. Lafont (2004): *Soft linear logic and polynomial time*. *Theoretical Computer Science* 318, pp. 163–180.

[16] D. Leivant (1991): *A Foundational Delineation of Computational Feasiblity*. In: *LICS*. pp. 2–11.

[17] D. Leivant (1993): *Stratified functional programs and computational complexity*. In: *POPL*. pp. 325–333.

[18] D. Leivant & J.-Y. Marion (1993): *Lambda calculus characterizations of poly-time*. In: *TLCA*. pp. 274–288.

[19] D. Leivant & J.-Y. Marion (1994): *Ramified recurrence and computational complexity I: Word recurrence and poly-time*. In: P. Clote & J. Remmel, editors: *Feasible Mathematics II*. Birkhauser, pp. 320 – 343.

[20] D. Leivant & J.-Y. Marion (1995): *Ramified recurrence and computational complexity II: substitution and poly-space*. In: *CSL*. pp. 486–500.

[21] D. Leivant & J.-Y. Marion (1997): *Predicative Functional Recurrence and Poly-space*. In: *TAPSOFT*. pp. 369–380.

[22] D. Leivant & J.-Y. Marion (2000): *A characterization of alternating log time by ramified recurrence*. *Theoretical Computer Science* 236(1-2), pp. 192–208.