Church Meets Cook and Levin

Damiano Mazza
CNRS, UMR 7030, LIPN, Université Paris 13, Sorbonne Paris Cité
Damiano.Mazza@lipn.univ-paris13.fr

Abstract

The Cook-Levin theorem (the statement that SAT is NP-complete) is a central result in structural complexity theory. Is it possible to prove it using the lambda-calculus instead of Turing machines? We address this question via the notion of affine approximation, which offers the possibility of using order-theoretic arguments, in contrast to the machine-level arguments employed in standard proofs. However, due to the size explosion problem in the lambda-calculus (a linear number of reduction steps may generate exponentially big terms), a naive transliteration of the proof of the Cook-Levin theorem fails. We propose to fix this mismatch using the author's recently introduced parsimonious lambda-calculus, reproving the Cook-Levin theorem and several related results in this higher-order framework. We also present an interesting relationship between approximations and intersection types, and discuss potential applications.

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—Reducibility and completeness

1. Introduction

The Cook-Levin theorem and its avatars. The Cook-Levin theorem is the statement that SAT (the satisfiability problem for propositional formulas) is NP-complete. The key to its proof is a simple and yet deep observation which complexity theorists usually describe with the slogan "computation is local" (Arora and Barak 2009). In fact, the core of the proof of the Cook-Levin theorem may be reused almost identically to show that:

1. CIRCUIT SAT is NP-complete;[1]
2. CIRCUIT VALUE is P-complete;[2]
3. P has polysize circuits.[3]

[1] CIRCUIT SAT is the following problem: given a Boolean circuit C with k inputs, does there exist $x \in \{0,1\}^k$ such that $C(x) = 1$? The complexity parameter is the size of C.

[2] CIRCUIT VALUE is the following problem: given a Boolean circuit C whose inputs are all fixed, is its output 1?

[3] Formally: for every $L \in \mathsf{P}$, there exists a family $(C_n)_{n\in\mathbb{N}}$ of Boolean circuits and a polynomial p s.t. the size of C_n is $O(p(n))$ and $C_n(x) = 1$ iff $x \in L$, for all $x \in \{0,1\}^n$, $n \in \mathbb{N}$ (*i.e.*, the family decides L).

LICS '16, July 05 - 08, 2016, New York, NY, USA
Copyright © 2016 ACM ISBN 978-1-4503-4391-6/16/07…$15.00
DOI: http://dx.doi.org/10.1145/http://dx.doi.org/10.1145/2933575.2934541

If we take the locality of computation to be the essence of the NP-completeness of SAT, then any one of the above results could be called "the Cook-Levin theorem".

Let us look more closely at the proof of these results. A deterministic Turing machine whose runtime is bounded by $p(n)$ induces, on inputs of size n, a square array $a_{i,j}$ of size $p(n)^2$ consisting of the "space-time" of the execution of the machine, the i-th line representing the contents of the tape at step i, plus information about the position of the head and the state. The key observation now is that line $i+1$ is induced by line i by means of purely local rules: $a_{i+1,j}$ is entirely determined by $a_{i,j-1}$, $a_{i,j}$ and $a_{i,j+1}$. This dependency is implementable by a constant-size Boolean circuit (depending on the machine). Therefore, each $n \in \mathbb{N}$ induces a circuit C of size $O(p(n)^2)$ with n inputs and 1 output (which isolates the answer of the machine), such that, given $x \in \{0,1\}^n$, computing the value of C with its inputs set to the bits of x is tantamount to deciding acceptance/rejection of x.

Now, if p is a polynomial, the size of C is also polynomial, hence P has polysize circuits. Furthermore, the description of C may be given in logarithmic space (or even less), which shows the P-hardness of CIRCUIT VALUE. From here, it is not difficult to infer the NP-hardness of CIRCUIT SAT. Finally, the NP-hardness of SAT follows from a simple reduction from CIRCUIT SAT.

Church meets Cook and Levin (and they don't get along). Let us try to look at the Cook-Levin theorem from a higher-order perspective. The first thing we need to do is to find an adequate notion of "higher-order Boolean circuit". We believe that this should be taken to coincide with *linear λ-terms* (multiplicative proof nets would be even better, but let us stick to λ-calculus). There are several reasons for this choice: like Boolean circuits, linear λ-terms may only compute finite functions; like Boolean circuits, the (sequential) runtime of a linear λ-term coincides with its size; finally, Boolean circuits may be seen as morphisms of a free symmetric monoidal category (in fact, a PROP), while (normal) linear λ-terms are morphisms in a free *closed* symmetric monoidal category.

In fact, for technical reasons it is better to consider *affine* λ-terms (erasing is permitted, not duplication), which are a minor variant still adhering to the above picture. Let us therefore take the following "equation" as our starting point:

$$\frac{\text{affine } \lambda\text{-terms}}{\lambda\text{-terms}} \quad = \quad \frac{\text{Boolean circuits}}{\text{Turing machines}}$$

The relationship between affine λ-terms and general λ-terms may be refined by formalizing the intuition (already present in (Girard 1987)) that the affine λ-calculus is "dense" in the full λ-calculus. This is the object of (Mazza 2012), which may be informally summarized as follows:

- there is a notion of *affine approximation* $t \sqsubset M$ of a λ-term by affine terms t, such that M may be seen as the limit of its affine approximations;
- reduction is continuous: if $M \to^* N$, then for all $u \sqsubset N$ there exists $t \sqsubset M$ such that $t \to^* u$.

Therefore, in perfect analogy with the fact that a Turing machine induces a family of circuits, we have that a higher-order program induces a family of higher-order circuits. The upshot is that this family is the result of a formal notion of continuous approximation, replacing the intuitive notion available with Turing machines.

Can we go further? Can we prove the Cook-Levin theorem or one of its avatars? First of all, it is fair to replace CIRCUIT VALUE with its higher-order version, let us call it AFFINE NORMAL FORM: given an affine λ-term, does it reduce to the Church Boolean $\underline{1}$? We could obtain a higher-order version of CIRCUIT SAT in a similar way, but let us stop and state already two higher-order avatars of the Cook-Levin theorem which we would like to prove:

1. AFFINE NORMAL FORM is P-complete;
2. polystep λ-terms induce polysize higher-order circuits.

Here is where we run into trouble. Indeed, (2) is just false in general: approximating λ-calculus computations of linear length may require exponentially big terms. Retrospectively, this was to be expected: computation in the λ-calculus is *not* local, the size of terms may grow exponentially with reduction, as in the reducts of DDA where $D := \lambda d.\lambda a.dd(aa)$. For what concerns (1), it is formally true: AFFINE NORMAL FORM is obviously in P and (Mairson 2004) showed how to logspace-reduce CIRCUIT VALUE to AFFINE NORMAL FORM (morally, this is because higher-order circuits subsume usual, first-order circuits). However, the failure of (2) makes it impossible to prove a P-hardness result in the style of the Cook-Levin theorem, *i.e.*, we do not know how to prove the P-completeness of AFFINE NORMAL FORM without knowing the existence of another non-trivial P-complete problem.

Enter parsimony. As stated above, λ-terms are limits of affine terms. In (Mazza 2014), we found a condition on such limits ensuring their good behavior with respect to approximating reductions. This led to the *parsimonious λ-calculus*, or $\mathrm{p}\Lambda$, recently developed in (Mazza and Terui 2015; Mazza 2015). In logical terms, $\mathrm{p}\Lambda$ corresponds to a variant of (affine) linear logic in which the exponential modality verifies the isomorphism $!A \cong A \otimes !A$. Its untyped version is Turing-complete and may be taken as the ground model in our higher-order analysis of the Cook-Levin theorem.

The main achievement of this paper is to reconcile, so to speak, Church with Cook and Levin. Indeed, we reprove the Cook-Levin theorem and all of its avatars mentioned above within the higher-order framework provided by $\mathrm{p}\Lambda$. With respect to Turing machines, λ-calculi have a superior algebraic structure and our proofs fully exploit it, relying on standard order-theoretic properties such as bounded completeness and Scott-continuity.

For example, let us sketch the proof that P has polysize higher-order circuits (*i.e.*, item (2) above, corresponding to Theorem 13 below). We denote by $(\ell\Lambda, \sqsubseteq)$ the poset of affine λ-terms under the approximation ordering, which extends to the approximation relation $\sqsubset$ on $\ell\Lambda \times \mathrm{p}\Lambda$ (remember that the terms of $\mathrm{p}\Lambda$ may be regarded as suprema of certain directed sets of $\ell\Lambda$). The key elements are:

1. monotonicity of reduction in $\ell\Lambda$ (Lemma 3.1): given $t \in \ell\Lambda$ s.t. $t \to^* u$, for all $t' \in \ell\Lambda$ s.t. $t \sqsubseteq t'$, we have $t' \to^* u'$ s.t. $u \sqsubseteq u'$ (*i.e.*, over-approximations do not lose information);
2. bounded completeness (Lemma 10): if $t, t' \sqsubset M$, then their supremum $t \sqcup t'$ exists;
3. rank and size bounds: there is a structural parameter of terms of $\ell\Lambda$, called *rank*, which verifies the following:
   - $\mathrm{rk}(t \sqcup t') = \max(\mathrm{rk}(t), \mathrm{rk}(t'))$ (Lemma 1);
   - for fixed M, the size $|t|$ of $t \sqsubset M$ is polynomial in $\mathrm{rk}(t)$ (Lemma 8);
4. (quantitative) Scott-continuity of reduction in $\mathrm{p}\Lambda$ (Proposition 7): given $M \in \mathrm{p}\Lambda$ s.t. $M \to N$, for all $u \sqsubset N$ there exists $t \sqsubset M$ s.t. $t \to u$ and $\mathrm{rk}(t) \leq \mathrm{rk}(u) + 1$ (*i.e.*, to know an affine approximation of the result, an affine approximation of the initial term suffices, whose rank increases by a constant);
5. discreteness of data: Church encodings of binary strings and Booleans in $\mathrm{p}\Lambda$ are actually terms of $\ell\Lambda$ and admit themselves as their only approximation. Moreover, their rank is 0.

Let now $M \in \mathrm{p}\Lambda$ decide a language in polynomially many reduction steps, that is, for all $w \in \{0,1\}^*$, $M\underline{w} \to^l \underline{b_w}$ with $l = O(|w|^c)$ for some constant c ($\underline{w}$ is the Church encoding of w and $\underline{b_w}$ is a Church Boolean). By continuity (4) and discreteness (5), there exists $v_w \sqsubset M\underline{w}$ s.t. $v_w \to^l \underline{b_w}$ and $\mathrm{rk}(v_w) = O(|w|^c)$. The definition of approximation (Fig. 5) implies that $v_w = t_w\underline{w}$ with $t_w \sqsubset M$; the definition of rank (Definition 1) and discreteness (5) give us $\mathrm{rk}(t_w) = \mathrm{rk}(v_w)$. For $n \in \mathbb{N}$, let $t_n := \bigsqcup_{|w|=2^n} t_w$, which exists by bounded completeness (2). Moreover, by the rank bound (3), $\mathrm{rk}(t_n) = O(n^c)$. Observe now that, since t_n is a least upper bound, we have $t_w \sqsubseteq t_n \sqsubset M$ for all $w \in \{0,1\}^n$. The first relation, by monotonicity (1), assures us that $t_n\underline{w} \to^* \underline{b_w}$ for all $w \in \{0,1\}^n$, *i.e.*, t_n decides the same language as M on strings of length n; the second relation combined with the size bound (3) gives us that $|t_n| = O(n^k)$ for some constant k. Hence, a language decided by a polystep parsimonious term is also decided by a family of affine terms of polynomial size, as desired.

We hope that the above proof conveys the conceptual improvement offered by the λ-calculus approach: the low-level simulation of a Turing machine by a Boolean circuit is replaced by a high-level argument based on continuity and compatible suprema. Furthermore, it turns out that approximations are related to a certain system of intersection types, where Scott-continuity corresponds to subject expansion. This is an interesting perspective that we mention at the end of the paper (Sect. 6).

Why the Cook-Levin theorem? The present work is motivated by research in implicit computational complexity (ICC), which is about replacing clocks with certificates: instead of defining a complexity class $\mathcal{C}$ by means of explicit resource bounds, one seeks a (decidable) property $\Phi_{\mathcal{C}}$ of programs such that $\Phi_{\mathcal{C}}(P)$ certifies that the program P decides a language in $\mathcal{C}$ (soundness) and, conversely, given $L \in \mathcal{C}$ there exists at least one P deciding L such that $\Phi_{\mathcal{C}}(P)$ holds (completeness, which makes ICC non-trivial). Seminal examples of this methodology are (Bellantoni and Cook 1992; Leivant and Marion 1993; Girard 1998; Jones 1999).

Note that the soundness and completeness of $\Phi_{\mathcal{C}}$ with respect to $\mathcal{C}$ morally correspond to membership to $\mathcal{C}$ and $\mathcal{C}$-hardness of a $\mathcal{C}$-complete problem, so ICC may also be understood as offering an alternative to the notion of complete problem, which is one of the hinges of traditional structural complexity theory. However, in this respect ICC has not been nearly as successful as complete problems: despite its fruitful applications to static analysis, from the standpoint of computational complexity ICC is virtually useless. This paper may be seen as a first attempt at uplifting the proof-theoretic roots of ICC beyond the mere characterization of complexity classes, towards an understanding of fundamental complexity phenomena which is complementary (and, hopefully, equally interesting) to that offered by traditional structural complexity. Given the connection between implicit characterizations and complete problems, it seemed natural to turn our attention to the primordial completeness result.

Outline. The rest of the paper is articulated as follows. In Sect. 2 we introduce the calculus $\ell\Lambda$ of affine terms, as well as the notion of approximation for usual λ-terms, concluding that the Cook-Levin theorem cannot be reformulated in this context. Sect. 3 is where we introduce the parsimonious λ-calculus and prove the main approximation properties, which are used in Sect. 4 to prove the higher-order version of all the avatars of the Cook-Levin theorem mentioned above. In Sect. 5 we go back to the first-order world and

prove the actual Cook-Levin theorem, *i.e.*, we show how the results of Sect. 4 may be used to infer that CIRCUIT SAT is NP-complete. It is known since the work of Terui (Terui 2004) that proof nets offer a convenient bridge between Boolean circuits and the higher-order setting, so we follow the same path. The heart of Sect. 5 is a compilation of higher-order circuits, under the guise of *Boolean proof nets*, into usual circuits, which serves as the reduction needed to show NP-hardness of CIRCUIT SAT. This compilation is of independent interest because it avoids the non-standard reachability gate employed by Terui (although his results provide a finer analysis of parallelism). Unsurprisingly, it is a "machine-level" construction, employing no order-theoretic property. Therefore, to be fair, our λ-calculus-based proof of the Cook-Levin theorem still contains low-level arguments; however, they are only used in building a reduction, *i.e.*, the conceptual part of the theorem (the existence of a non-trivial NP-complete problem) is entirely abstract.

2. The Polyadic Affine Calculus

Terms. We fix two disjoint, countably infinite sets of *affine variables*, ranged over by a, b, c, and *exponential variables*, ranged over by x, y, z. The terms of the polyadic affine calculus belong to the following grammar:

$$t, u ::= a \mid \lambda a.t \mid tu \mid x_i \mid !\bot \mid t :: u \mid t[!x := u]$$

There are two binders: the usual λa and $t[!x := u]$, which binds x in t. This is indeed a compact form of the more traditional notation $\mathsf{let}\ !x = u\ \mathsf{in}\ t$. The set of free variables of t is denoted by $\mathrm{fv}(t)$. The notion of α-equivalence is defined as usual and terms are always considered up to it. We also sometimes appeal to *Barendregt's convention*, under which $\mathrm{fv}(t)$ and the set of bound variables of t are disjoint and distinct binders of t bind distinct variables.

A *proper term* must meet the following linearity restriction: modulo Barendregt's convention, every affine variable appears at most once and each occurrence of exponential variable appears with a distinct index. We will only ever consider proper terms, the set of which we denote by $\ell\Lambda$. When putting together two proper terms t, u, forming *e.g.* tu, we will always tacitly assume the result to be proper.

The affine part of $\ell\Lambda$ is standard: it is a term language for the proofs of intuitionistic multiplicative affine logic with the connective $\multimap$ only (*i.e.*, without $\otimes$). The exponential part essentially corresponds to adding a $\otimes$ connective of arbitrary arity. We use the notation $\langle u_1, \dots, u_n \rangle$ for the term $u_1 :: (u_2 :: \cdots :: (u_n :: !\bot) \cdots)$, which we call a *sequence* of length n.

DEFINITION 1 (Size, depth, rank, homogeneity). *The* size *of $t \in \ell\Lambda$, denoted by $|t|$, is the number of nodes in its syntactic tree, counting leaves of the form x_i as of size $i + 1$.*

The depth *of t, denoted by $\mathrm{d}(t)$, is the maximum nesting level of the sequences it contains, e.g. $\mathrm{d}(\langle x_1, \langle y_0 \rangle \rangle) = 2$.*

The rank *of t, denoted by $\mathrm{rk}(t)$, is the length of the longest sequence appearing in t. We say that t is* homogeneous *if all of its sequences have the same length. If t contains no sequence, it is homogeneous of every rank.*

Reduction. A *context*, generically denoted by C, is a term with exactly one occurrence of a special constant called *hole*, denoted by $\langle\cdot\rangle$. We write $\mathsf{C}\langle t \rangle$ for the term obtained by replacing $\langle\cdot\rangle$ with t in C. We will also employ a kind of generalized context, in which the hole appears *at most* once (instead of exactly once). We call such contexts *affine* and denote them by A. Thus, when we write $t = \mathsf{A}\langle x_0 \rangle$, we mean that x_0 may or may not appear (once) in t.

A special class of contexts is that of *substitution contexts*, ranged over by the symbol $[-]$ and generated as follows:

$$[-] ::= \langle\cdot\rangle \mid [-][!x := t]$$

Because of their particular shape, for substitution contexts we use the notation $t[-]$ instead of $[-]\langle t \rangle$.

Given $k \in \mathbb{Z}$, we write t^{x+k} to denote the term obtained from t by replacing every free occurrence x_i with x_{i+k}. When $k = 1$ or $k = -1$, we write t^{x++} and t^{x--}, respectively. The notations $t^{\pm k}$, t^{++} and t^{--} are defined similarly, except that *all* free occurrences of exponential variables are re-indexed. The same notations also apply to contexts.

The (one step) reduction relation on $\ell\Lambda$ is defined in Fig. 1, where Barendregt's convention is assumed. Reduction in $\ell\Lambda$ trivially confluent, because of affinity: the absence of duplication immediately implies that $\rightarrow^{=}$ (the reflexive closure of reduction) has the diamond property, from which confluence of $\rightarrow^{*}$ (the reflexive-transitive closure of reduction) follows by standard diagram chasing. Also observe that the size of terms strictly decreases by reduction, so every term of $\ell\Lambda$ (strongly) normalizes in a number of steps bounded by its size.

Ordering. We introduce a binary relation $\sqsubseteq$ on $\ell\Lambda$ by means of the derivation system of Fig. 2. Intuitively, $t \sqsubseteq t'$ iff t is obtained from t' by replacing some subterms of the form $u :: v$ with $!\bot$.

LEMMA 1. *$(\ell\Lambda, \sqsubseteq)$ is a finitely bounded-complete poset, i.e., if $F \subseteq \ell\Lambda$ is finite, non-empty and has an upper bound, then its supremum $\bigsqcup F$ exists. Moreover, $\mathrm{rk}(\bigsqcup F) = \max_{t \in F} \mathrm{rk}(t)$.*

PROOF. Reflexivity, transitivity and antisymmetry of $\sqsubseteq$ are readily proved by induction on derivations. The bounded completeness property follows from the case in which which F has cardinality 2, so let $t, t' \in \ell\Lambda$. We define $t \sqcup t'$ by induction on t:

- $t = a$ or $t = x_i$: then $t \sqcup t' := t$ if $t' = t$, otherwise it is undefined;
- $t = \lambda a.t_1$: then $t \sqcup t' = \lambda a.(t_1 \sqcup t'_1)$ if $t' = \lambda a.t'_1$, otherwise it is undefined;
- $t = t_1 t_2$: then $t \sqcup t' := (t_1 \sqcup t'_1)(t_2 \sqcup t'_2)$ if $t' = t'_1 t'_2$, or undefined otherwise;
- $t = !\bot$: then $t \sqcup t' := t'$ if $t' = !\bot$ or $t' = t'_1 :: t'_2$, or undefined otherwise;
- $t = t_1 :: t_2$: then $t \sqcup t' = t$ if $t' = !\bot$, or $t \sqcup t' = (t_1 \sqcup t'_1) :: (t_2 \sqcup t'_2)$ if $t' = t'_1 :: t'_2$, or undefined otherwise;
- $t = t_1[!x := t_2]$: then $t \sqcup t' := t_1 \sqcup t'_1[!x := t_2 \sqcup t'_2]$ if $t' = t'_1[!x := t'_2]$, or undefined otherwise.

An easy induction on t establishes that, if $t \sqcup t'$ is defined, then $\mathrm{rk}(t \sqcup t') = \max\{\mathrm{rk}(t), \mathrm{rk}(t')\}$. It is also straightforward to show, by induction on u, that whenever $t, t' \sqsubseteq u$, $t \sqcup t'$ and $t' \sqcup t$ are both defined and equal and $t, t' \sqsubseteq t \sqcup t' \sqsubseteq u$, proving that it is indeed the supremum. □

The ordering relation may be extended to contexts by adding to Fig. 2 a nullary rule deriving $\langle\cdot\rangle \sqsubseteq \langle\cdot\rangle$. In the following, we use ξ, ξ' to denote objects which may be either terms or contexts of $\ell\Lambda$.

LEMMA 2. *1. $\xi \sqsubseteq \xi'$ implies $\mathrm{fv}(\xi) = \mathrm{fv}(\xi')$;*
2. $\xi \sqsubseteq \xi'$ iff $\xi^{x+i} \sqsubseteq (\xi')^{x+i}$ for every x and $i \in \mathbb{N}$;
3. $t \sqsubseteq \mathsf{C}'\langle u' \rangle$ iff $t = \mathsf{C}\langle u \rangle$ for some $u \sqsubseteq u'$ and $\mathsf{C} \sqsubseteq \mathsf{C}'$;
4. $\mathsf{C}\langle u \rangle \sqsubseteq t'$ iff $t' = \mathsf{C}'\langle u' \rangle$ for some $u \sqsubseteq u'$ and $\mathsf{C} \sqsubseteq \mathsf{C}'$.

PROOF. A straightforward induction on derivations (points 1 and 2), on C' (point 3) and on C (point 4). □

LEMMA 3 (Monotonicity and continuity). *Let $t \in \ell\Lambda$ and $t \rightarrow u$.*
1. for all t' s.t. $t \sqsubseteq t'$, there exists u' s.t. $u \sqsubseteq u'$ and $t' \rightarrow u'$;
2. for all u' s.t. $u' \sqsubseteq u$, there exists t' s.t. $t' \sqsubseteq t$ and $t' \rightarrow u'$.

PROOF. By definition, $t = \mathsf{C}\langle t_0 \rangle$ and $u = \mathsf{C}\langle u_0 \rangle$ with t_0 and u_0 matching the left and right hand side of one of the rules of Fig. 1. Both points are proved by induction on C. The only interesting case

$$(\lambda a.\mathsf{A}\langle a\rangle)[-]u \to_{\mathsf{b}} \mathsf{A}\langle u\rangle[-]$$
$$\mathsf{A}\langle x_0\rangle[!x := (u :: v)[-]] \to_{!} \mathsf{A}^{x--}\langle u\rangle[!x := v][-]$$
$$t[!x := (!\bot)[-]] \to_{\mathsf{w}} t[-] \qquad \text{if } x \notin \mathrm{fv}(t)$$

Figure 1. Reduction rules for the polyadic affine calculus. Reduction is the closure of these rules under arbitrary contexts.

$$\frac{}{a \sqsubseteq a} \qquad \frac{t \sqsubseteq t'}{\lambda a.t \sqsubseteq \lambda a.t'} \qquad \frac{t \sqsubseteq t' \quad u \sqsubseteq u'}{tu \sqsubseteq t'u'} \qquad \frac{}{x_i \sqsubseteq x_i} \qquad \frac{}{!\bot \sqsubseteq !\bot} \qquad \frac{}{!\bot \sqsubseteq t :: u} \qquad \frac{t \sqsubseteq t' \quad u \sqsubseteq u'}{t :: u \sqsubseteq t' :: u'} \qquad \frac{t \sqsubseteq t' \qquad u \sqsubseteq u'}{t[!x := u] \sqsubseteq t'[!x := u']}$$

Figure 2. The ordering on $\ell\Lambda$. In the nullary rule deriving $!\bot \sqsubseteq t :: u$, t and u are arbitrary.

is $\mathsf{C} = \langle\cdot\rangle$, the rest is straightforward (using Lemma 2). Moreover, we only prove point 1, because we never need point 2 and, actually, we prove a very similar result with essentially the same proof in the context of the parsimonious λ-calculus (Proposition 7).

The first case is $t_0 = (\lambda a.\mathsf{A}\langle a\rangle)[-]v$ and $u_0 = \mathsf{A}\langle v\rangle[-]$. Then, Lemma 2.4 gives us $t' = (\lambda a.\mathsf{A}'\langle a\rangle)[-]'v'$ with $\mathsf{A} \sqsubseteq \mathsf{A}'$, $[-] \sqsubseteq [-]'$ and $v \sqsubseteq v'$, so we conclude by setting $u' := \mathsf{A}'\langle v'\rangle[-]'$ (using again Lemma 2.4).

The second case is $t_0 = \mathsf{A}\langle x_0\rangle[!x := (v :: w)[-]]$ and $u_0 = \mathsf{A}^{x--}\langle v\rangle[!x := w][-]$. Lemma 2.4 gives us $t' = \mathsf{A}'\langle x_0\rangle[!x := (v' :: w')[-]']$ with $v \sqsubseteq v'$, $w \sqsubseteq w'$, $[-] \sqsubseteq [-]'$ and $\mathsf{A} \sqsubseteq \mathsf{A}'$. From this, we get $\mathsf{A}^{x++} \sqsubseteq (\mathsf{A}')^{x++}$ (Lemma 2.2) whence $\mathsf{A}^{x++}\langle v\rangle \sqsubseteq (\mathsf{A}')^{x++}\langle v'\rangle$ (Lemma 2.4 again), so we concluding by letting $u' := (\mathsf{A}')^{x--}\langle v'\rangle[!x := w'][-]'$.

The third (and last) case is $t_0 = v[!x := (!\bot)[-]]$ with $x \notin \mathrm{fv}(v)$ and $u_0 = v[-]$. Lemma 2.4 gives us $t' = v'[!x := (!\bot)[-]']$ with $[-] \sqsubseteq [-]'$, $v \sqsubseteq v'$ and $x \notin \mathrm{fv}(v')$ by Lemma 2.1, so it is enough to set $u' := v'[-]'$ to conclude. □

The proofs of the two parts of Lemma 3 look a lot like those of the subject reduction (point 1) and subject expansion (point 2) property of a type system. In fact, approximations and type systems are related, as we will see in Sect. 6.

Approximating lambda-terms. Observe that the poset $(\ell\Lambda, \sqsubseteq)$ is *not* a dcpo: typically, the chain $!\bot \sqsubseteq \langle x_0\rangle \sqsubseteq \langle x_0, x_1\rangle \sqsubseteq \langle x_0, x_1, x_2\rangle \sqsubseteq \cdots$ has no supremum. Suprema of directed sets of polyadic affine terms may be presented as infinitary affine λ-terms, in which usual (non-affine) λ-terms may be faithfully embedded. This idea was developed in (Mazza 2012); here, we avoid infinitary calculi and define approximations directly.

Let us denote by Λ the set of usual λ-terms.

DEFINITION 2 (Approximations in the λ-calculus). *Given $t \in \ell\Lambda$ and $M \in \Lambda$, we say that t* approximates *M if the judgment $t \sqsubset M$ may be derived from the rules of Fig. 3 (which also define an auxiliary relation denoted by $\sqsubset^!$).*

The proof of the following result, which we omit because it is essentially a rephrasing of a result of (Mazza 2012), may also be seen as a subject expansion property, in analogy with Lemma 3.2.

PROPOSITION 4 (Continuity of β-reduction). *Let $M \in \Lambda$ and let $M \to_\beta N$. Then, for all $u \sqsubset N$, there exists $t \sqsubset M$ such that $t \to^+ u$.*

So, if $M \to_\beta^n N$ (reduction in n steps) with N containing no applications (for instance, N is a Church Boolean) and u is the unique approximation of N, we have that there exists $t \sqsubset M$ such that $t \to^m u$. Reduction in $\ell\Lambda$ strictly reduces the size of terms, so $m \leq |t|$. But how big may $|t|$ be, as a function of n? The next definition, which is intentionally vague as to the meaning of "calculus", is meant to give a general setting to deal with the above question.

DEFINITION 3. *We say that a calculus $(\mathcal{C}, \to)$ has* affine approximations *if there is an approximation relation $\sqsubset$ with $\ell\Lambda$ such that:*

1. *reduction is continuous (i.e., Proposition 4 holds);*
2. *each $M \in \mathcal{C}$ admits a homogeneous rank 1 approximation, denoted by $\lfloor M\rfloor_1$, of size linear in $|M|$;*
3. *for each $M \in \mathcal{C}$, there is $d \in \mathbb{N}$ such that $t \sqsubset M$ implies $\mathrm{d}(t) \leq d$; the least such d is said to be the* depth *of M.*

In the following, calculi will be assumed to have affine approximations.

We say that a calculus $(\mathcal{C}, \to)$ enjoys polysize reduction *if, for every family of terms $M_n \in \mathcal{C}$ of depth $O(1)$, there exists $k \in \mathbb{N}$ such that, whenever $M_n \to^{l(n)} N_n$, we have $|N_n| = O(|M_n| l(n)^k)$.*

We say that a calculus $(\mathcal{C}, \to)$ has polynomial approximations *if, for every family of terms $M_n \in \mathcal{C}$ of depth $O(1)$, there exists $k \in \mathbb{N}$ such that, whenever $M_n \to^{l(n)} N_n$, there exist $t_n \sqsubset M_n$ such that $t_n \to^* \lfloor N_n\rfloor_1$ and $|t_n| = O(|M_n| l(n)^k)$.*

Observe that the λ-calculus has affine approximations: the only non-obvious point is (3), which is however not hard to verify (the depth of M turns out to be equal to its applicative depth, *e.g.*, the depth of $I(II)$ is 2).

PROPOSITION 5. *If a calculus $(\mathcal{C}, \to)$ has polynomial approximations then it enjoys polysize reduction.*

PROOF. We assume that $\mathcal{C}$ has affine approximations (otherwise the result is vacuously true) and prove the contrapositive. Suppose that there is a sequence of terms M_n of depth $O(1)$ such that, for all $k \in \mathbb{N}$, there are reductions $M_n \to^{l(n)} N_n$ such that $|N_n| = \omega(|M_n| l(n)^k)$. Now, take any family $t_n \sqsubset M_n$ such that $t_n \to^* \lfloor N_n\rfloor_1$. Since reduction in $\ell\Lambda$ shrinks the size, we have $|t_n| \geq |\lfloor N_n\rfloor_1| = \Theta(|N_n|) = \omega(|M_n| l(n)^k)$. □

The λ-calculus does not enjoy polysize reduction: if we set $M = DDI$ with $D := \lambda d.\lambda a.dd(aa)$, we have a constant family M such that $M \to_\beta^n N_n$ with $|N_n| = \Theta(2^n)$ (N_n contains 2^n copies of I). Therefore, as anticipated above, the λ-calculus does not have polynomial approximations.

Enjoying polysize reduction is interesting because, modulo some mild assumptions on the calculus (such as the fact that one reduction step $M \to N$ is itself implementable in time polynomial in $|M|$), it immediately implies that reduction in the calculus may be simulated by Turing machines with a polynomial slowdown.

However, we stress that the polysize reduction property, albeit sufficient, is *not* necessary for the unitary cost model to be polynomially related to Turing machines: the λ-calculus itself is a prominent example, as shown by (Accattoli and Dal Lago 2014).

$$\frac{}{x_i \sqsubset x} \qquad \frac{t \sqsubset M}{\lambda a.t[!x := a] \sqsubset \lambda x.M} \qquad \frac{t \sqsubset M \quad u \sqsubset^! N}{tu \sqsubset MN} \qquad \frac{}{!\bot \sqsubset^! M} \qquad \frac{t \sqsubset M \quad u \sqsubset^! M}{t :: u \sqsubset^! M}$$

Figure 3. The approximation relation for the λ-calculus. In the top left rule, $i \in \mathbb{N}$ is arbitrary.

3. The Parsimonious Lambda-Calculus

Terms. We consider the same sets of affine and exponential variables as for the polyadic affine calculus. The terms of the parsimonious λ-calculus belong to the following grammar:[4]

$$M, N ::= a \mid \lambda a.M \mid MN \mid x_i \mid !M \mid M[!x := N].$$

The only difference with respect to $\ell\Lambda$ is the ! operator, which degenerates to the constant $!\bot$ in $\ell\Lambda$. A *parsimonious term* is one verifying the following constraints:

- **affinity**: modulo Barendregt's convention, every affine variable appears at most once and each occurrence of exponential variable appears with a distinct index;
- **boxes**: in each subterm $!M$ (called a *box*), all variables of $\mathrm{fv}(M)$, which are said to be *locally free*, are exponential;
- **parsimony:** each exponential variable appears locally free in at most one box (counting nesting) and at most once therein; moreover, if x_i is such an occurrence and x_j is any other occurrence, then $i > j$.

We denote by $\mathrm{p}\Lambda$ the set of parsimonious terms.

For example, $\Delta := \lambda a.x_0!x_1[!x := a]$ and $\Delta!\Delta$ are parsimonious. On the other hand, the following are not parsimonious: $\lambda a.!a$ (the affine variable a appears locally free inside a box); $a!x_7!x_2$ or $(!!x_3)[!x := a]$ (the exponential variable x appears locally free in two boxes, nested in the second case; by contrast $!(!x_3[!x := y_2])$ is parsimonious); $!(x_0x_1)$ (a box containing two locally free occurrences of the same exponential variable); $x_3!x_1$ (the index of the occurrence of x in the box is not maximal in the term).

Using the syntax and the reindexing notations introduced for $\ell\Lambda$, the intuition is that $!M$ is equivalent to $M :: !M^{++}$ and therefore to the infinite stream $M :: M^{+1} :: M^{+2} :: \cdots$. This justifies the restriction on the locally free variables of boxes: if $a \in \mathrm{fv}(M)$, $M :: !M^{++}$ would not be affine. It also explains the parsimony constraints: if any of them is violated, one may obtain a term breaking the affinity condition by sufficiently expanding $!M$.

The *size* of $M \in \mathrm{p}\Lambda$, denoted by $|M|$, is defined as in Definition 1. The *exponential depth* of M is defined to be the maximum nesting level of its boxes and denoted by $\mathrm{d}(M)$.

Reduction. Substitution contexts are defined exactly as in $\ell\Lambda$. In $\mathrm{p}\Lambda$, we use the terminology *affine context*, still denoted by A, to mean a context with *at most one* occurrence of $\langle\cdot\rangle$ *which does not appear in a box*. If the hole does appear, the context is called *linear* (this definition is vacuous in $\ell\Lambda$ because all non-affine contexts there are linear). Thus, writing $M = \mathsf{A}\langle x_0\rangle$ is a succinct way of expressing that, if x_0 appears in M, then it is not locally free in a box.

The reduction rules for the parsimonious λ-calculus are defined in Fig. 4. These are closed under *linear* contexts only.

Polynomial approximations. $\mathrm{p}\Lambda$ has affine approximations.

DEFINITION 4 (Approximations in $\mathrm{p}\Lambda$). *If $t \in \ell\Lambda$ and $M \in \mathrm{p}\Lambda$, we say that t* approximates *M if the judgment $t \sqsubset M$ may be derived from the rules of Fig. 5. The approximation relation is extended to contexts by adding $\langle\cdot\rangle \sqsubset \langle\cdot\rangle$ as a nullary rule.*

[4] The one presented here is actually a subset of the parsimonious λ-calculus as introduced in (Mazza and Terui 2015; Mazza 2015). This fragment suffices for our present purposes and we adopt it for simplicity.

We observe that point (3) of Definition 3 is verified: the depth of $M \in \mathrm{p}\Lambda$ coincides with its exponential depth. Point (2) is obviously verified. We will now check point (1).

LEMMA 6. *Let M, C and Ξ be a term, a linear context and either a term or a linear context of $\mathrm{p}\Lambda$, respectively, and let ξ be either a term or a context of $\ell\Lambda$.*

1. *$\xi \sqsubset \Xi$ implies $\mathrm{fv}(\xi) = \mathrm{fv}(\Xi)$;*
2. *$\xi \sqsubset \Xi$ iff $\xi^{x+i} \sqsubset \Xi^{x+i}$ for every x and $i \in \mathbb{N}$;*
3. *$t \sqsubset \mathsf{C}\langle M\rangle$ iff $t = \mathsf{C}_0\langle u\rangle$ for some $u \sqsubset M$ and $\mathsf{C}_0 \sqsubset \mathsf{C}$;*

PROOF. Essentially identical to Lemma 2. □

PROPOSITION 7 (Quantitative continuity). *Let $M \in \mathrm{p}\Lambda$ and let $M \to N$. Then, for all $u \sqsubset N$ there exists $t \sqsubset M$ such that $t \to u$. Moreover, $\mathrm{rk}(t) \leq \mathrm{rk}(u) + 1$.*

PROOF. By definition, we have $M = \mathsf{C}\langle M_0\rangle$ and $N = \mathsf{C}\langle N_0\rangle$, with C a linear context and M_0, N_0 matching the left and right hand side, respectively, of one of the rewriting rules of Fig. 4. The proof is by induction on C. The only interesting case is $\mathsf{C} = \langle\cdot\rangle$, the rest is straightforward, using Lemma 6.3.

Let $M_0 \to_{\mathsf{b}} N_0$, which implies that $M_0 = (\lambda a.P)[-]Q$ and $u \sqsubset P\{Q/a\}[-]$. Actually, since there is at most one free occurrence of a in P, we may write $P = \mathsf{A}\langle a\rangle$ and $P\{Q/a\} = \mathsf{A}\langle Q\rangle$. Hence, by Lemma 6.3, $u = \mathsf{A}'\langle q\rangle[-]'$ with $q \sqsubset Q$, $\mathsf{A}' \sqsubset \mathsf{A}$ and $[-]' \sqsubset [-]$. Then, if we let $t := (\lambda a.\mathsf{A}'\langle a\rangle)[-]'q$, we have $t \sqsubset M_0$ (Lemma 6.3 again), $t \to_{\mathsf{b}} u$, and $\mathrm{rk}(t) = \mathrm{rk}(u)$.

Let $M_0 \to_! N_0$, which implies $M_0 = \mathsf{A}\langle x_0\rangle[!x := (!P)[-]]$ and $u \sqsubset \mathsf{A}^{x--}\langle P\rangle[!x := !P^{++}][-]$. By definition of approximation and Lemma 6.3, we have $u = \mathsf{A}'\langle v\rangle[!x := w][-]'$ with $\mathsf{A}' \sqsubset \mathsf{A}^{x--}$, $v \sqsubset P$, $w \sqsubset !P^{++}$ and $[-]' \sqsubset [-]$. Now, by Lemma 6.2, $(\mathsf{A}')^{x++} \sqsubset \mathsf{A}$, so $(\mathsf{A}')^{x++}\langle x_0\rangle \sqsubset \mathsf{A}\langle x_0\rangle$ by Lemma 6.3. Then, if we let $t := (\mathsf{A}')^{x++}\langle x_0\rangle[!x := (v_0 :: w)[-]']$, we have $t \sqsubset M_0$, $t \to_! u$ and $\mathrm{rk}(t) = \mathrm{rk}(u) + 1$.

Let $M_0 \to_{\mathsf{w}} N_0$, which means that $M_0 = P[!x := (!Q)[-]]$ with $x \notin \mathrm{fv}(P)$ and $u \sqsubset P[-]$. By Lemma 6.3, $u = p[-]'$ with $p \sqsubset P$ and $[-]' \sqsubset [-]$. In this case, it is enough to consider $t := p[!x := (!\bot)[-]']$, which obviously satisfies $t \sqsubset M_0$ and, by Lemma 6.1, $t \to_{\mathsf{w}} u$, with $\mathrm{rk}(t) = \mathrm{rk}(u)$. □

Proposition 7 implies that $\mathrm{p}\Lambda$ actually has polynomial approximations, via the following lemma:

LEMMA 8 (Size bound). *For all $M \in \mathrm{p}\Lambda$ and $t \in \ell\Lambda$, $t \sqsubset M$ implies $|t| \leq |M|(\mathrm{rk}(t) + 1)^{\mathrm{d}(M)}$.*

PROOF. By induction on M. We only check the case $M = !N$, the rest is straightforward. We claim that $t \sqsubset !N$ implies $t = \langle u_0, \ldots, u_{n-1}\rangle$ with $u_i \sqsubset N^{+i}$ for all $0 \leq i < n$, which is itself easily shown by induction (on the derivation of $t \sqsubset !N$). Using the induction hypothesis, the fact that $\mathrm{rk}(t)$ bounds n as well as all $\mathrm{rk}(u_i)$, and that $|M| = |N| + 1$ and $\mathrm{d}(M) = \mathrm{d}(N) + 1$, we have

$$|t| = 1 + \sum_{i=0}^{n-1} |u_i| \leq 1 + \sum_{i=0}^{n-1} |N|(\mathrm{rk}(u_i) + 1)^{\mathrm{d}(N)}$$
$$\leq 1 + |N|(\mathrm{rk}(t) + 1)^{\mathrm{d}(N)+1} \leq |M|(\mathrm{rk}(t) + 1)^{\mathrm{d}(M)},$$

as desired. □

$$(\lambda a.\mathsf{A}\langle a\rangle)[-]N \to_{\mathsf{b}} \mathsf{A}\langle N\rangle[-]$$
$$\mathsf{A}\langle x_0\rangle[!x := (!N)[-]] \to_{!} \mathsf{A}^{x--}\langle N\rangle[!x := !N^{++}][-]$$
$$M[!x := (!N)[-]] \to_{\mathsf{w}} M[-] \qquad x \notin \mathrm{fv}(M)$$

Figure 4. Reduction rules for the parsimonious λ-calculus. Reduction is the closure of these rules under *linear* contexts.

$$\frac{}{a \sqsubset a} \qquad \frac{t \sqsubset M}{\lambda a.t \sqsubset \lambda a.M} \qquad \frac{t \sqsubset M \quad u \sqsubset N}{tu \sqsubset MN} \qquad \frac{}{x_i \sqsubset x_i} \qquad \frac{}{!\bot \sqsubset !M} \qquad \frac{t \sqsubset M \quad u \sqsubset !M^{++}}{t :: u \sqsubset !M} \qquad \frac{t \sqsubset M \qquad u \sqsubset N}{t[!x := u] \sqsubset M[!x := N]}$$

Figure 5. The approximation relation for the parsimonious λ-calculus.

COROLLARY 9. *The parsimonious λ-calculus has polynomial approximations.*

PROOF. Let $M_n \in \mathrm{p}\Lambda$ be a family of terms whose depths are bounded by d and let $M_n \to^{l(n)} N_n$ be reductions. If $l(n)$ is identically null, we may trivially conclude so let us assume that $l(n) = \Omega(1)$. By Proposition 7, we have $t_n \sqsubset M_n$ such that $t_n \to^{l(n)} \lfloor N_n \rfloor_1$ and $\mathrm{rk}(t_n)+1 \leq l(n)+2 = \Theta(l(n))$. Therefore, using Lemma 8, we have $|t_n| = O(|M_n| l(n)^d)$. □

Parsimonious terms may be seen as particular elements of the ideal completion of $(\ell\Lambda, \sqsubseteq)$ (this is a rephrasing of the the topological approach of (Mazza 2012)), which is a bounded complete dcpo. The following result is therefore not surprising:

LEMMA 10 (Bounded completeness). *Let $M \in \mathrm{p}\Lambda$, $n > 0$ and $t_1, \ldots, t_n \sqsubset M$. Then, $\bigsqcup_i t_i$ exists and $\bigsqcup_i t_i \sqsubset M$.*

PROOF. Analogous to Lemma 1 (M plays the role of u). □

Although the set of approximations of $M \in \mathrm{p}\Lambda$ does not have a supremum in $\ell\Lambda$ (the supremum being M itself), there is a greatest element if we only consider approximations up to a given rank k, namely the homogeneous approximation of M of rank k, denoted by $\lfloor M \rfloor_k$. It may be defined inductively as follows: $\lfloor a \rfloor_k := a$; $\lfloor \lambda a.N \rfloor_k := \lambda a.\lfloor N \rfloor_k$; $\lfloor NP \rfloor_k := \lfloor N \rfloor_k \lfloor P \rfloor_k$; $\lfloor x_i \rfloor_k := x_i$; $\lfloor !N \rfloor_k := \langle \lfloor N \rfloor_k, \ldots, \lfloor N^{+(k-1)} \rfloor_k \rangle$; $\lfloor N[!x := P] \rfloor_k := \lfloor N \rfloor_k [!x := \lfloor P \rfloor_k]$.

LEMMA 11. *Given $M \in \mathrm{p}\Lambda$ and $k \in \mathbb{N}$, $\lfloor M \rfloor_k$ is the greatest element of the set $\{t \in \ell\Lambda \mid t \sqsubset M,\ \mathrm{rk}(t) \leq k\}$.*

PROOF. A straightforward induction on M. □

4. Reconciling Church with Cook and Levin

The parsimonious λ-calculus is a general model of computation: it is Turing-complete. This is not obvious a priori because, as far as we know, there is no direct encoding of the λ-calculus in $\mathrm{p}\Lambda$. For instance, $\mathrm{p}\Lambda$ apparently lacks a general fixpoint combinator. However, it does have a *linear* fixpoint combinator, given by $\mathsf{Y}_\ell := \mathsf{X}!\mathsf{X}$, where $\mathsf{X} := \lambda a.\lambda f.y_0(x_0!x_1!y_1)[!x := a][!y := f]$. Intuitively, a recursive definition is linear when it is of the form (in Caml syntax) `let rec` $f(x_1, \ldots, x_n) = \Phi$ with f occurring once in Φ. The chief example of linear (tail) recursive definitions are while loops, which are enough to achieve Turing-completeness.

The encoding of binary strings in $\mathrm{p}\Lambda$ is also a variant of the usual Church encoding: the string $w = b_0 \cdots b_{n-1} \in \{0,1\}^n$ is encoded by

$$\underline{w} := \lambda s^0.\lambda s^1.\lambda z.x_0^{b_0}(\ldots x_{n-1}^{b_{n-1}} z \ldots)[!x^0 := s^0][!x^1 := s^1].$$

The Church Booleans are $\underline{0} := \lambda a.\lambda b.b$ and $\underline{1} := \lambda a.\lambda b.a$, as usual.

DEFINITION 5. *We let $\mathrm{p}\lambda\mathsf{TIME}(f)$ be the class of languages $L \subseteq \{0,1\}^*$ such that there exists a closed $M \in \mathrm{p}\Lambda$ such that, for all $n \in \mathbb{N}$ and $w \in \{0,1\}^n$, $M\underline{w} \to^l \underline{b}$ with $b \in \{0,1\}$ (a Church Boolean), $l \leq f(n)$ and $w \in L$ iff $b = 1$.*

The above definition mimics the definition of the usual deterministic time class $\mathsf{TIME}(f)$, so the following is expected:

PROPOSITION 12. $\mathsf{TIME}(f) \subseteq \mathrm{p}\lambda\mathsf{TIME}(\mathsf{poly}(f))$ *and* $\mathrm{p}\lambda\mathsf{TIME}(f) \subseteq \mathsf{TIME}(\mathsf{poly}(f))$.

PROOF. The first is shown by simulating Turing machines in $\mathrm{p}\Lambda$ with a polynomial slowdown, which is unsurprising. The converse is a consequence of the polysize reduction property of $\mathrm{p}\Lambda$ (which holds by Corollary 9 and Proposition 5). □

DEFINITION 6. *A* higher order Boolean circuit *with n inputs is a term $t \in \ell\Lambda$ such that $\mathrm{fv}(t) = \{a^1, \ldots, a^n\}$ (all affine variables) and such that, for all Church Booleans $\underline{b_1}, \ldots, \underline{b_n}$, $t\{\underline{b_1}/a^1, \ldots, \underline{b_n}/a^n\}$ reduces to a Church Boolean.*

We say that a language $L \subseteq \{0,1\}^*$ has polysize higher order Boolean circuits *if there exists a family of higher order Boolean circuits $(t_n)_{n\in\mathbb{N}}$ such that t_n has n inputs and, for all $w = b_1 \cdots b_n \in \{0,1\}^n$, $t_n\{\underline{b_1}/a^1, \ldots, \underline{b_n}/a^n\} \to^* \underline{1}$ iff $w \in L$.*

THEOREM 13. P *has polysize higher order Boolean circuits.*

PROOF. By Proposition 12, if $L \in \mathsf{P}$ then there is $M \in \mathrm{p}\Lambda$ deciding L in polynomially many steps. Fix an enumeration $(w_i)_{i\in\mathbb{N}}$ of binary strings. By Corollary 9 applied to the family $M\underline{w_i}$, we have $v_i \sqsubset M\underline{w_i}$ such that v_i reduces to a Boolean $\underline{b}$ (because $\lfloor \underline{b} \rfloor_1 = \underline{b}$) with $b = 1$ iff $w_i \in L$ and $\mathrm{rk}(v_i)$ is polynomial in $|w_i|$. But $v_i = t_i\underline{w_i}$ with $t_i \sqsubset M$ and $\mathrm{rk}(t_i) = \mathrm{rk}(v_i)$ ($\underline{w_i}$ has no box, so it is the only approximation of itself and does not contribute to the rank). Now, for all $n \in \mathbb{N}$, let $i_1^n, \ldots, i_{2^n}^n$ be all the indices such that $|w_{i_j^n}| = n$ and let $u_n := \bigsqcup_j t_{i_j^n}$. By Lemma 10, this is defined and $u_n \sqsubset M$. Moreover, by the rank bound of Lemma 1, $\mathrm{rk}(u_n)$ is polynomial in n, which means that, by Lemma 8, $|u_n|$ is also polynomial in n ($|M|$ and $\mathrm{d}(M)$ are independent of n, of course). Furthermore, since $t_{i_j^n}\underline{w_{i_j^n}} \sqsubset u_n\underline{w_{i_j^n}}$, thanks to Lemma 3.1 we have that u_n still decides L on strings of length n. All that is left to do is to convert u_n into a higher order Boolean circuit (Definition 6). This is done by means of terms with the property that $\mathsf{boolToStr}_n\{\underline{b_1}/a^1, \ldots, \underline{b_n}/a^n\} \to^* \underline{b_1 \cdots b_n}$, *i.e.*, $\mathsf{boolToStr}_n$ builds a Church string of length n out the Boolean bits constituting it. These are easy to construct. Then, $(u_n\mathsf{boolToStr}_n)_{n\in\mathbb{N}}$ is a polysize family of higher order circuits deciding L. □

We let AFFINE NORMAL FORM be the following problem: given $t \in \ell\Lambda$, do we have $t \to^* \underline{1}$ (the Church Boolean)? This is tantamount to computing the value of a higher order Boolean circuit

with its inputs fixed, so we could also call this problem HO CIRCUIT VALUE.

THEOREM 14. AFFINE NORMAL FORM *is* P*-complete.*

PROOF. It is clear that AFFINE NORMAL FORM $\in \mathsf{P}$: reduction shrinks the size of terms and the naive algorithm which repeatedly scans for a redex and reduces it is obviously polynomial in the size of the initial term (it is actually quadratic).

For P-hardness, we need to find a logspace reduction from any $L \in \mathsf{P}$ to AFFINE NORMAL FORM. By Theorem 13, we know that there exists $M \in \mathrm{p}\Lambda$ such that it is enough to consider a family $(t_n)_{n\in\mathbb{N}} \sqsubset M$ of rank polynomial in n in order to decide $w \in L$ for all $w \in \{0,1\}^n$, $n \in \mathbb{N}$. Terms of the form $t_n\underline{w}$ are instances of AFFINE NORMAL FORM, so we are close to our goal. Unfortunately, though, we do not know how to compute t_n in logspace (w.r.t. n). However, if $k(n) := \mathrm{rk}(t_n)$, then $t_n \sqsubset \lfloor M \rfloor_{k(n)}$ (Lemma 11) and, by Lemma 3.1, $\lfloor M \rfloor_{k(n)}$ also decides L on strings of length n. For fixed M (as in our case), homogeneous approximations of rank k *can* be computed in space logarithmic in k (see the definition given before Lemma 11). In order to determine a given node in the syntactic tree of $\lfloor M \rfloor_{k(n)}$ (or the absence of such), all we need is a pointer to the syntactic tree of M and an integer counter bounded by $k(n) + m$, where m is the maximum index of exponential occurrences in M (a constant). Storing this counter in binary only occupies logarithmic space because $k(n)$ is polynomial in n. □

Let HO CIRCUIT SAT be the following problem: given a higher order Boolean circuit t with n inputs (Definition 6), are there $b_1, \ldots, b_n \in \{0,1\}$ such that $t\{\underline{b_1}/a^1, \ldots, \underline{b_n}/a^n\} \to^* \underline{1}$?

THEOREM 15. HO CIRCUIT SAT *is* NP*-complete.*

PROOF. Essentially analogous to showing that CIRCUIT SAT is NP-complete from the proof that CIRCUIT VALUE is P-complete (Papadimitriou 1994). □

5. Proof Nets and Satisfiability

Our aim now is to show that HO CIRCUIT SAT reduces to CIRCUIT SAT, thus completing the proof of the Cook-Levin theorem (modulo the standard reduction from CIRCUIT SAT to SAT). The idea is to implement reduction of polyadic affine terms with Boolean circuits. Following Terui (Terui 2004), we use proof nets as a convenient bridge between terms and circuits.

DEFINITION 7 (Net). *Let* $\Sigma := \{\mathsf{ax}, \mathsf{cut}, \otimes, \parr, \mathsf{w}, !_0, !, ?\}$. *Each symbol of* Σ *has an associated arity:* ax *and* cut *have arity* 2*;* $\otimes$*,* $\parr$*,* ! *and* ? *have arity* 3*;* w *and* $!_0$ *have arity* 1*. A* net π *is a tuple* $(\mathsf{P}_\pi, \mathsf{C}_\pi, \ell_\pi, \mathsf{p}_\pi)$ *where:*

- P_π *and* C_π *are finite sets, the elements of which are called* ports *and* cells*, respectively;*
- $\ell_\pi : \mathsf{C}_\pi \to \Sigma$ *is the* labelling *function;*
- p_π *is a function assigning with each cell* c *a tuple of ports, the arity of which is equal to the arity of* $\ell_\pi(c)$*. The ports of* $\mathsf{p}_\pi(c)$ *are divided into* conclusions *and* premises *and have a* polarity *(relative to* c*), all depending on* $\ell_\pi(c)$*; these are given in Fig. 6, as explained momentarily.*

In addition, a net π *must verify the following conditions:*

- *each port is conclusion of exactly one cell and premise of at most one cell;*
- *if a port is conclusion of* c *and premise of* c'*, then its polarities relative to* c *and* c' *are opposite.*

In Fig. 6, which gives the graphical representation of cells, we specify the polarities, as well as the conclusions and premises of cells: the former are displayed as outgoing, the latter as incoming. When writing $\mathsf{p}_\pi(c)$ *as a tuple, we separate conclusions/premises by means of a semicolon, with the tuple order matching the left-to-right order in graphical representations. So, for instance, if* $\ell_\pi(c) = \otimes$*, then* $\mathsf{p}_\pi(c) = (p; q, r)$*, with* p *the conclusion and* q, r *the positive and negative premise, respectively.*

The size *of a net is the number of its ports.*

Nets are usually required to satisfy some form of correctness, yielding *proof nets*. We will not specify any correctness criterion here, we will rather take "proof net" as a synonym of a net which is the (reduct of the) encoding of a polyadic affine term.

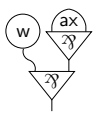
Cut-elimination steps are defined in Fig. 7. Polarities and orientations are omitted because they can be recovered without ambiguity from Fig. 6. In fact, such information will be omitted in all subsequent graphical representations. In the $\to_{\mathsf{ax}_-}$ step, the negative premise of the cut cell must not be an ax cell; in the $\to_{\mathsf{ax}_+}$ step, the negative conclusion of the ax cell must not be a cut cell. We denote by $\to$ the rewriting relation induced by cut-elimination, which is trivially confluent and strongly normalizing.

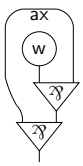
The definition of cut-elimination for axioms deserves a comment. The usual definition allows cuts on axioms to be eliminated without conditions, leading to critical pairs such as the following:

From the rewriting point of view, such critical pairs are harmless because they are trivially confluent. However, as first observed by (Terui 2004), they are problematic when one wants to implement proof net cut-elimination with Boolean circuits, as we are going to do below. It is immediate to check that, with our definition, there are no critical pairs. This solution, which differs from the one given in (Terui 2004), is possible because we are using polarized (*i.e.*, intuitionistic) nets. It exploits less the parallelism of circuits (which is not of concern here) but has the advantage that the implementation uses the standard basis $\{\neg, \wedge, \vee\}$ of fan-in 2, the one traditionally adopted for formulating CIRCUIT SAT, whereas (Terui 2004) employs a non-standard reachability gate.

Terms as proof nets. Terms of $\ell\Lambda$ may be encoded in proof nets. A term t with n occurrences of free variables induces a proof net $(\!|t|\!)$ with $n+1$ free ports, one labelled by a special symbol $\bullet$ (corresponding to the root of t) and the others labelled by the free occurrences themselves. The net $(\!|t|\!)$ is defined by induction on t, as as shown in Fig. 9. To avoid cluttering the pictures, only the conclusions strictly relevant to the inductive construction of the net are drawn. The multiplicative cases (top row) are standard. Of the exponential cases, only the encoding of $t[!x := u]$ deserves an explanation. For that, we first associate with every binary word $w \in \{0,1\}^*$ a net Δ_w, as described in Fig. 8. Now, if $x_{i_1}, \ldots, x_{i_n}$ are all the free occurrences of x in t, with $i_1 < \cdots < i_n$, we define the binary word $w = w_0 \cdots w_{i_n}$ as follows: $w_j = 1$ if $j = i_k$ for some $1 \le k \le n$, otherwise $w_j = 0$. In other words, the j-th bit of w is "turned on" iff x_j is free in t.

Two important examples, used in the sequel: if $\underline{0} := \lambda a.\lambda b.b$ and $\underline{1} := \lambda a.\lambda b.a$ are the Church Booleans, we have

$(\!|\underline{0}|\!) =$ 

$(\!|\underline{1}|\!) =$ 

LEMMA 16. *Let* $t \in \ell\Lambda$ *be closed. Then,* $t \to^* \underline{1}$ *(the Church Boolean) iff*

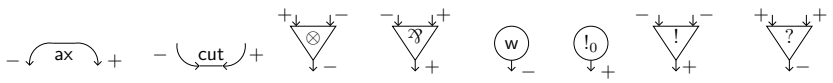


Figure 6. Cells for building nets, with their polarity assignment.

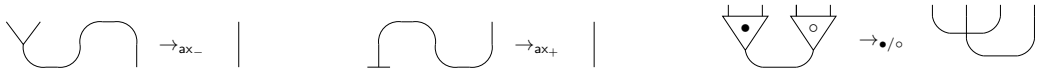


Figure 7. Cut-elimination steps on nets. In the bottom step, $\bullet = \otimes$ and $\circ = \parr$, or $\bullet = !$ and $\circ = ?$.

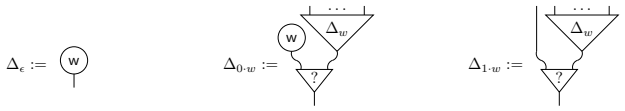


Figure 8. The inductive definition of the net Δ_w, with $w \in \{0,1\}^*$.

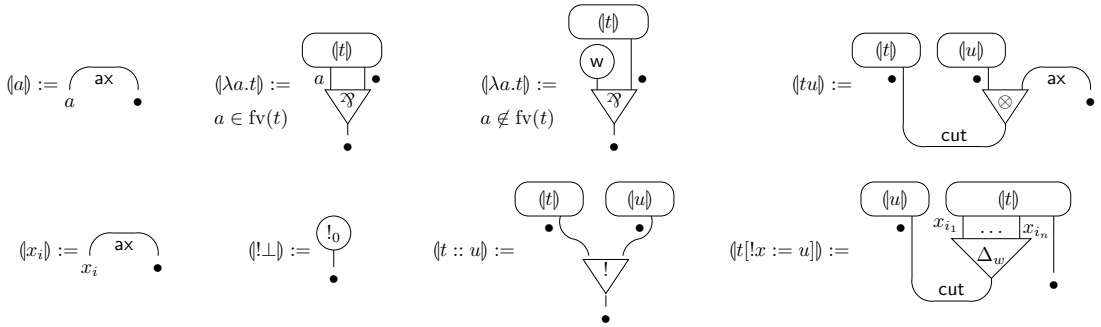
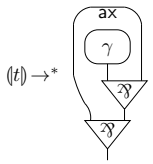
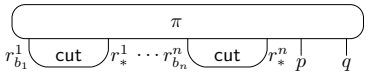


Figure 9. The encoding of $\ell\Lambda$ in proof nets.



PROOF. A special case of the standard simulation of term reduction by cut-elimination in proof nets. We do not obtain exactly the net $(|\underline{1}|)$ because we did not define cut-elimination for w cells, so, instead of being a w cell, γ may contain some garbage. □

Proof net satisfiability. Let PROOF NET SAT be the following problem: given a net π, $3n$ of its free ports $r_0^1, r_1^1, r_*^1, \ldots, r_0^n, r_1^n, r_*^n$ and 2 other of its free ports p, q, all pairwise distinct, are there $b_1, \ldots, b_n \in \{0,1\}$ such that



(we did not draw the free ports other than p, q and those concerned by the cuts) reduces by cut-elimination to a net in which p and q are the conclusions of the same ax cell? The complexity parameter for this problem is the size of π.

In the sequel, we write $\leq_m^{\mathsf{poly}}$ for polynomial-time many-one reducibility (also known as Karp reducibility).

LEMMA 17. HO CIRCUIT SAT $\leq_m^{\mathsf{poly}}$ PROOF NET SAT.

PROOF. A consequence of Lemma 16, observing that the distinction between $(|\underline{1}|)$ and $(|\underline{0}|)$ is just the presence/absence of an axiom connecting a certain pair of ports. □

Thanks to the absence of critical pairs, the operation of reducing in parallel all cuts in a net is well defined:

DEFINITION 8 (Parallel cut-elimination). *We write $\pi \Rightarrow \pi'$ when π' is obtained from π by simultaneously reducing all cuts, or $\pi' = \pi$ in case π is irreducible.*[5]

Note that there is exactly one π' such that $\pi \Rightarrow \pi'$. It is this deterministic operation which we will implement with a circuit.

Let P be a set of ports. For each $p, q, r \in P$ we consider Boolean variables associated with cells: $\mathrm{ax}(p,q;)$, $\mathrm{cut}(;p,q)$, $\mathrm{tens}(p;q,r)$, $\mathrm{par}(p;q,r)$, etc. (as the notation suggests, these are morally predicates, but technically they are variables). The semicolon separates conclusions and premises and is only mnemonic. A net π such that $\mathsf{P}_\pi \subseteq P$ will be represented as a valuation on the above variables: *e.g.* $\mathrm{ax}(p,q;) = 1$ iff in π there is an ax cell whose negative conclusion is p and positive conclusion is q.

LEMMA 18. *There exists a circuit ParRed_P of size $O(|P|^7)$ which, given in input π with $\mathsf{P}_\pi \subseteq P$, outputs the unique π' such that $\pi \Rightarrow \pi'$.*

[5] We say "irreducible" instead of cut-free because cut-elimination as defined in Fig. 7 does not eliminate cuts on w and $!_0$ cells.

$$\mathrm{reloc}_-(p',p) := \bigvee_s \mathrm{axcut}_-(p',s,p) \vee \left(p' = p \wedge \bigwedge_{s,s'} \neg \mathrm{axcut}_-(s',s,p) \right)$$

$$\mathrm{tens}'(p;q,r) := \bigwedge_{p',q',r'} \neg \mathrm{logcut}(p',q',r',p,q,r) \wedge \left(\bigvee_{p',q'} \mathrm{tens}(p';q',r) \wedge \mathrm{reloc}_+(p',p) \wedge \mathrm{reloc}_-(q',q) \right)$$

Figure 10. An example of output of the circuit ParRed_P.

PROOF. Let us give the idea. We want to know if π' contains a certain cell, *e.g.* we want to know whether in π' there is a $\otimes$ cell of conclusion p and premises q, r. This is tantamount to computing the output variable $\mathrm{tens}'(p;q,r)$ of ParRed_P (we add a "prime" to denote outputs). Now, there is such a cell c in π' only if, in π, there is no such cell involved in a $\otimes/\parr$ cut (which would disappear in π'); if such a necessary condition is met, then c must result from a $\otimes$ cell of π, because cut-elimination does not create $\otimes$ cells. The presence of a $\otimes/\parr$ cut in π may be computed by the constant-size circuit $\mathrm{logcut}(p,p',p'',q,q',q'')$ defined by

$$\mathrm{cut}(;p',p) \wedge \mathrm{par}(p';q',r') \wedge \mathrm{tens}(p;q,r).$$

Similarly, one may define circuits (this time of size $O(|P|)$) $\mathrm{axcut}_-(p_1,p_2,p_3)$ and $\mathrm{axcut}_+(p_1,p_2,p_3)$ detecting the presence of cuts of type ax_- and ax_+ at the given ports. To express the fact that axiom elimination rules "relocate" ports, we define the circuit $\mathrm{reloc}_-(p',p)$ as in Fig. 10 (s, s' range over P), which is of size $O(|P|^2)$. The meaning is that $\mathrm{reloc}_-(p',p) = 1$ iff either port p' will be relocated to port p because of the presence of an ax_- cut, or $p' = p$ is a port that will *not* be relocated by such a redex. A similar circuit, called $\mathrm{reloc}_+(p',p)$, may be given for ax_+ cuts. The circuit computing $\mathrm{tens}'(p;q,r)$ may then be defined as in Fig. 10, which reads: there is a $\otimes$ cell c of conclusion p and premises q, r in π' iff, in π, there is no such cell involved in a $\otimes/\parr$ cut and there is a similar cell which results in c, possibly after a relocation. The other outputs are computed following the same idea.

To compute the total size of ParRed_P, one may check that the worst case is given by the outputs of the form $\mathrm{tens}'(p;q,r)$ (and those of the form $\mathrm{par}'(p;q,r)$), of which there are $O(|P|^3)$, each computed by a circuit of size $O(|P|^4)$, which brings the size of ParRed_P to $O(|P|^7)$, as stated. □

LEMMA 19. PROOF NET SAT $\leq_m^{\mathsf{poly}}$ CIRCUIT SAT.

PROOF. Let π, r_0^i, r_1^i, r_*^i (with $1 \leq i \leq n$), p, q be an instance of PROOF NET SAT, with π of size s. We define an instance of CIRCUIT SAT as follows. First, we build the circuit

$$C_0 := \overbrace{\mathrm{ParRed}_{\mathsf{P}_\pi} \circ \cdots \circ \mathrm{ParRed}_{\mathsf{P}_\pi}}^{s}$$

(the outputs of the first copy are fed as inputs to the second, and so on). From this, we define a circuit C_1 by fixing the inputs of C_0 to the valuation corresponding to π, leaving unset only the inputs corresponding to $\mathrm{cut}(;r_b^i,r_*^i)$ ($1 \leq i \leq n$, $b \in \{0,1\}$) and we isolate the output corresponding to $\mathrm{ax}'(p,q;)$ (for instance by xoring all other outputs with themselves and $\vee$-ing the result with $\mathrm{ax}'(p,q;)$). Now, there obviously is a constant-size circuit which, on input x_i, sets $\mathrm{cut}(;r_1^i,r_*^i) = 1$ and $\mathrm{cut}(;r_0^i,r_*^i) = 0$ if $x_i = 1$ and dually if $x_i = 0$; we plug these n circuits to the relevant inputs of C_1, obtaining C. Such a C is obviously computable in time linear in its size, which is polynomial in s by Lemma 18. Furthermore, it is clear that C is satisfiable iff π is. □

All in all, we compile a higher order circuit of size s into a fan-in 2 circuit over $\{\neg, \wedge, \vee\}$ of size $O(s^8)$. By looking at (Terui 2004), one finds a compilation with a bound $O(s^4)$, which seems much better. However, Terui uses unbounded fan-in and a non-standard reachability gate. If we allow unbounded fan-in, we too get a circuit of size $O(s^4)$. The difference is in the depth, which is always linear in our case, whereas in Terui's case, using the reachability gate, it may be sublinear or even bounded.

6. Types and Perspectives

Intersection types. Consider the following types:

$$A, B ::= o \mid A \multimap B \mid \langle A_1, \ldots, A_n \rangle,$$

where o is an atomic type. We use $\overrightarrow{A}$ as a shorthand for the type $\langle A_1, \ldots, A_n \rangle$, and define $A_0 :: \overrightarrow{A} := \langle A_0, A_1, \ldots, A_n \rangle$. The *height* of a type is the height of its syntactic tree.

The type $\overrightarrow{A}$ may be seen as a $\otimes$ of arbitrary arity. It may also be considered as an associative, non-commutative and non-idempotent intersection type. In fact, the approximation relation of Fig. 5 is related to an intersection type system for $\mathrm{p}\Lambda$, presented in Fig. 11. Typing contexts are of the form $\Gamma;\Delta$ where Γ contains type declarations for exponential variables, of the form $x : \overrightarrow{A}$, and Δ contains type declarations for affine variables, of the form $a : A$ (note that A is arbitrary in the latter case, while it must be a sequence type in the former).

The type system itself must be read ignoring the annotations in gray. Conversely, if we only consider the annotations in gray, we obtain the simply-typed $\ell\Lambda$. More exactly: if all term annotations are erased, one obtains the natural logic for the types/formulas introduced above, a polyadic version of propositional intuitionistic multiplicative affine logic. Via Curry-Howard, the proofs of this logic correspond to terms of $\ell\Lambda$, those annotated in gray. However, the *same* type system may be read as an intersection type system for $\mathrm{p}\Lambda$. Given a derivation δ of $\Gamma;\Delta \vdash M : A$, one may consider the polyadic term t associated with δ and it turns out that $t \sqsubset M$. In fact, the approximation relation is *exactly* the type-erasure of the type system, justifying the use of the $\sqsubset$ symbol in Fig. 11, which we invite the reader to compare to Fig. 5. Furthermore, as we already observed, all of our continuity results (Lemma 3, Propositions 4 and 7) are reflected into subject expansion properties.

We call our types "intersection types" for $\mathrm{p}\Lambda$ because, although they are just simple types for $\ell\Lambda$, they are *not* the propositional formulas of parsimonious logic, the image of $\mathrm{p}\Lambda$ on the logical side of Curry-Howard. These were considered in (Mazza 2015; Mazza and Terui 2015) and are obtained by replacing the type $\overrightarrow{A}$ with $!A$ in the above grammar; the corresponding type system is quite different from that of Fig. 11. Instead, the cons rule (with the black annotation) looks a lot like an intersection rule. Also, connections between linearization of the λ-calculus and intersection types were already observed by (Kfoury 2000). Indeed, the approximation relation of Fig. 3 too stems from a non-idempotent intersection type system for the usual λ-calculus (with the same types).

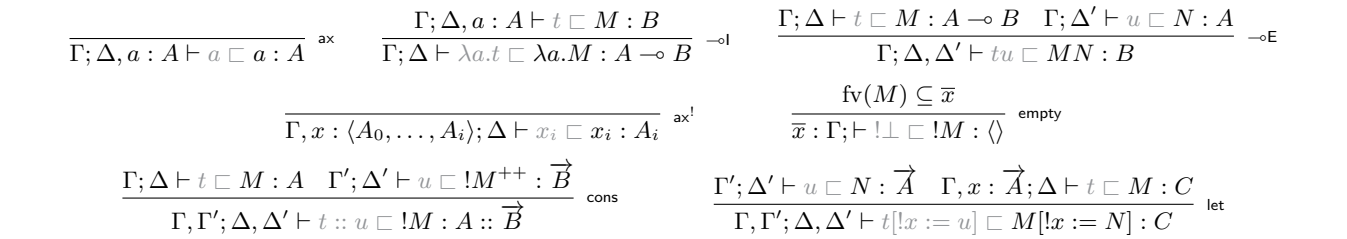


Figure 11. Simple types for $\ell\Lambda$ and intersection types for $\mathrm{p}\Lambda$, superposed.

Perspectives. What do we gain from this meeting of Church with Cook and Levin? In technical terms, the fine analysis of continuous approximations turns out to give much better bounds on the complexity of normalizing a simply-typed parsimonious term of size s and depth d: in (Mazza 2015), we had bounds of the form $O(s^{2^d})$, whereas our Lemma 8 here gives $O(s^{d^k})$, which is definitely an improvement.

In broader terms, the intriguing relationship between the affine approximations developed here and intersection types (originally observed by (Kfoury 2000)) offers a novel approach to the question of space cost models for functional languages. In other words, given a λ-term M, what do we count as space used by M? Or, if M decides a language L, what measure $f_M(n)$ can we take (where n is the input size) to infer that $L \in \mathsf{SPACE}(O(f_M(n)))$? To our knowledge, very few papers address this question (an example is (Spoonhower et al. 2008)) and none does it in an abstract, implementation-independent way. By contrast, work concerning abstract time cost models for the λ-calculus abounds, culminating in (Accattoli and Dal Lago 2014) (see the references therein).

The relationship between intersection types and complexity of λ-terms is well understood for time (de Carvalho 2009; Bernadet and Lengrand 2011; Terui 2012; De Benedetti and Ronchi Della Rocca 2013) but not for space. The key point is the use of the geometry of interaction (GoI), which is known to capture space-efficient computation, an idea originally due to (Schöpp 2007) and also (Dal Lago and Schöpp 2010). Without giving any detail, let us state a result which may be obtained by combining the GoI with reasoning similar to that of Theorem 14. We set

$$\overrightarrow{B}_{A_0,\dots,A_n} := \langle A_{n-1} \multimap A_n, \dots, A_0 \multimap A_1 \rangle,$$
$$\mathsf{Str}[A_0,\dots,A_n] := \overrightarrow{B}_{A_0,\dots,A_n} \multimap \overrightarrow{B}_{A_0,\dots,A_n} \multimap A_0 \multimap A_n.$$

What we obtain is somewhat analogous to (Borodin 1977):

THEOREM 20. *Let* $;\vdash t_n \sqsubset M : \mathsf{Str}[A_0^n,\dots,A_n^n] \multimap \mathsf{Bool}$*, and let* h_n *be the maximum height of* $A_0^n,\dots,A_n^n$*, for* $n \in \mathbb{N}$*. Then,* M *decides a language in* $\mathsf{SPACE}(O(h_n \log |t_n|))$.

Theorem 20 may be used to factorize soundness results for implicit characterizations of complexity classes. For instance, retrospectively, the soundness part of (Mazza 2015) was about proving that type derivations of the simply-typed $\mathrm{p}\Lambda$ may be mapped to intersection types derivations of the form required by Theorem 20, of polynomial size and bounded height (in n), yielding logspace languages. Theorem 20 was also implicitly used in (Mazza and Terui 2015), yielding a refinement of the results of (Terui 2004): in particular, polysize bounded-depth higher-order circuits are now known to capture L/poly, whereas they were previously characterized in terms of a non-standard class.

Let us conclude by observing that continuity results analogous to Proposition 7 and its Corollary 9 also hold for the *linear substitution calculus* with linear head reduction (Accattoli 2012), which shows that our methodology is general and may be applied to calculi which were defined completely independently of the polyadic affine λ-calculus.

Acknowledgments

We acknowledge partial support of ANR grants COQUAS (ANR-12-JS02-006-01) and ELICA (ANR-14-CE25-0005).

References

B. Accattoli. An abstract factorization theorem for explicit substitutions. In *Proceedings of RTA*, pages 6–21, 2012.
B. Accattoli and U. Dal Lago. Beta reduction is invariant, indeed. In *Proceedings of CSL-LICS*, page 8, 2014.
S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
S. Bellantoni and S. A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.
A. Bernadet and S. Lengrand. Complexity of strongly normalising lambda-terms via non-idempotent intersection types. In *Proceedings of FOSSACS*, pages 88–107, 2011.
A. Borodin. On relating time and space to size and depth. *SIAM J. Comput.*, 6(4):733–744, 1977.
U. Dal Lago and U. Schöpp. Functional programming in sublinear space. In *Proceedings of ESOP*, pages 205–225, 2010.
E. De Benedetti and S. Ronchi Della Rocca. Bounding normalization time through intersection types. In *Proceedings of ITRS*, pages 48–57, 2013.
D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009.
J.-Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–102, 1987.
J.-Y. Girard. Light linear logic. *Inf. Comput.*, 143(2):175–204, 1998.
N. D. Jones. Logspace and ptime characterized by programming languages. *Theor. Comput. Sci.*, 228(1-2):151–174, 1999.
A. J. Kfoury. A linearization of the lambda-calculus and consequences. *J. Log. Comput.*, 10(3):411–436, 2000.
D. Leivant and J.-Y. Marion. Lambda calculus characterizations of polytime. *Fundam. Inform.*, 19(1/2), 1993.
H. G. Mairson. Linear lambda calculus and PTIME-completeness. *J. Funct. Program.*, 14(6):623–633, 2004.
D. Mazza. An infinitary affine lambda-calculus isomorphic to the full lambda-calculus. In *Proceedings of LICS*, pages 471–480, 2012.
D. Mazza. Non-uniform polytime computation in the infinitary affine lambda-calculus. In *Proceedings of ICALP, Part II*, pages 305–317, 2014.
D. Mazza. Simple parsimonious types and logarithmic space. In *Proceedings of CSL*, pages 24–40, 2015.
D. Mazza and K. Terui. Parsimonious types and non-uniform computation. In *Proceedings of ICALP, Part II*, pages 350–361, 2015.
C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
U. Schöpp. Stratified bounded affine logic for logarithmic space. In *Proceedings of LICS*, pages 411–420, 2007.
D. Spoonhower, G. E. Blelloch, R. Harper, and P. B. Gibbons. Space profiling for parallel functional programs. *J. Funct. Program.*, 20(5-6): 417–461, 2008.
K. Terui. Proof nets and boolean circuits. In *Proceedings of LICS*, pages 182–191, 2004.
K. Terui. Semantic evaluation, intersection types and complexity of simply typed lambda calculus. In *Proceedings of RTA*, pages 323–338, 2012.