Free-cut elimination in linear logic and an application to a feasible arithmetic

Baillot, Patrick; Das, Anupam

DOI:
10.4230/LIPIcs.CSL.2016.40

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Baillot, P & Das, A 2016, Free-cut elimination in linear logic and an application to a feasible arithmetic. in J-M Talbot & L Regnier (eds), *25th EACSL Annual Conference on Computer Science Logic (CSL 2016).*, 40, Leibniz International Proceedings in Informatics, LIPIcs, vol. 62, Schloss Dagstuhl, pp. 40:1-40:18, 25th EACSL Annual Conference on Computer Science Logic, CSL 2016 and the 30th Workshop on Computer Science Logic, Marseille, France, 29/08/16. https://doi.org/10.4230/LIPIcs.CSL.2016.40

Free-Cut Elimination in Linear Logic and an Application to a Feasible Arithmetic*

Patrick Baillot[1] and Anupam Das[2]

1 **Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France**
2 **Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France**

Abstract

We prove a general form of 'free-cut elimination' for first-order theories in linear logic, yielding normal forms of proofs where cuts are anchored to nonlogical steps. To demonstrate the usefulness of this result, we consider a version of arithmetic in linear logic, based on a previous axiomatisation by Bellantoni and Hofmann. We prove a witnessing theorem for a fragment of this arithmetic via the 'witness function method', showing that the provably convergent functions are precisely the polynomial-time functions. The programs extracted are implemented in the framework of 'safe' recursive functions, due to Bellantoni and Cook, where the ! modality of linear logic corresponds to normal inputs of a safe recursive program.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases proof theory, linear logic, bounded arithmetic, polynomial time computation, implicit computational complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.40

1 Introduction

Free-cut elimination[1] is a normalisation procedure on formal proofs in systems including nonlogical rules, e.g. the axioms and induction rules in arithmetic, introduced in [26]. It yields proofs in a form where, essentially, each cut step has at least one of its cut formulas principal for a nonlogical step. It is an important tool for proving witnessing theorems in first-order theories, and in particular it has been extensively used in *bounded arithmetic* for proving complexity bounds on representable functions, by way of the *witness function method* [9].

Linear logic [14] is a decomposition of both intuitionistic and classical logic, based on a careful analysis of duplication and erasure of formulas. It has been useful in proofs-as-programs correspondences, proof search [1] and logic programming [24]. By controlling structural rules with designated modalities, the *exponentials*, linear logic has allowed for a fine study of complexity bounds in the Curry-Howard interpretation, inducing variants with polynomial-time complexity [17] [16] [18].

In this work we explore how the finer granularity of linear logic can be used to control complexity in *first-order theories*, restricting the provably convergent functions rather than the typable terms as in the propositional setting. We believe this to be of general interest,

* This work was supported by by the ANR Project ELICA ANR-14-CE25-0005 and by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX- 0007) operated by the French National Research Agency (ANR).

[1] Also known as *anchored* or *directed* completeness, *partial* cut-elimination or *weak* cut-elimination in other works.

in particular to understand the effect of substructural restrictions on nonlogical rules, e.g. induction, in mathematical theories. Some related works exist, e.g. the naïve set theories of Girard and Terui [15] [27], but overall it seems that the first-order proof theory of linear logic is still rather undeveloped; in particular, to our knowledge, there seems to be no general form of free-cut elimination available in the literature (although special cases occur in [22] and [3]). Thus our first contribution, in Sect. 3, is to provide general sufficient conditions on nonlogical rules for a first-order linear logic system to admit free-cut elimination.

We illustrate the usefulness of this result by proving a witnessing theorem for an arithmetic in linear logic, showing that the provably convergent functions are precisely the polynomial-time computable functions (Sects. 6 and 7), henceforth denoted **FP**. Our starting point is an axiomatisation $\mathcal{A}_2^1$ from [7], based on a modal logic, already known to characterise **FP**. This approach, and that of [20] before, differs from the bounded arithmetic approach since it does not employ bounds on quantifiers, but rather restricts nonlogical rules by substructural features of the modality [7] or by *ramification* of formulas [21]. The proof technique employed in both cases is a realisability argument, for which [20] operates directly in intuitionistic logic, whereas [7] obtains a result for a classical logic via a double-negation translation, relying on a higher-type generalisation of *safe recursion* [6].

We show that Buss' witness function method can be employed to extract functions directly for classical systems similar to $\mathcal{A}_2^1$ based in linear logic, by taking advantage of free-cut elimination. The De Morgan normal form available in classical (linear) logic means that the functions we extract remain at ground type, based on the usual safe recursive programs of [6]. A similar proof method was used by Cantini in [11], who uses combinatory terms as the model of computation as opposed to the equational specifications in this work.[2]

Our result holds for an apparently weaker theory than $\mathcal{A}_2^1$, with induction restricted to positive existential formulas in a way similar to Leivant's RT_0 system in [21] (see also [23]), but the precise relationship between the two logical settings is unclear. We conclude in Sect. 8 with a survey of related work and some avenues for further applications of the free-cut elimination result.

A version of this article containing further proof details in appendices is available [4].

2 Preliminaries

We formulate linear logic without units with usual notation for the multiplicatives, additives and exponentials from [14]. We restrict negation to the atoms, so that formulae are always in De Morgan normal form, and we also consider rules for arbitrary weakening when working in affine settings.

[2] This turns out to be important due to the handling of right-contraction steps in the witnessing argument.

▶ **Definition 1.** The sequent calculus for (affine) linear logic is as follows:[3]

$$\bot\text{-}l\,\frac{}{p, p^\bot \vdash} \qquad id\,\frac{}{p \vdash p} \qquad \bot\text{-}r\,\frac{}{\vdash p, p^\bot} \qquad cut\,\frac{\Gamma \vdash \Delta, A \quad \Sigma, A \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}$$

$$\parr\text{-}l\,\frac{\Gamma, A \vdash \Delta \quad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \parr B \vdash \Delta, \Pi} \qquad \otimes\text{-}l\,\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta} \qquad \parr\text{-}r\,\frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \parr B} \qquad \otimes\text{-}r\,\frac{\Gamma \vdash \Delta, A \quad \Sigma \vdash \Pi, B}{\Gamma, \Sigma \vdash \Delta, \Pi, A \otimes B}$$

$$\oplus\text{-}l\,\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta} \qquad \&\text{-}l\,\frac{\Gamma, A_i \vdash \Delta}{\Gamma, A_1 \& A_2 \vdash \Delta} \qquad \oplus\text{-}r\,\frac{\Gamma \vdash \Delta, A_i}{\Gamma \vdash \Delta, A_1 \oplus A_2} \qquad \&\text{-}r\,\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \& B}$$

$$?\text{-}l\,\frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta} \qquad !\text{-}l\,\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad ?\text{-}r\,\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, ?A} \qquad !\text{-}r\,\frac{!\Gamma \vdash ?\Delta, A}{!\Gamma \vdash ?\Delta, !A}$$

$$wk\text{-}l\,\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad cntr\text{-}l\,\frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad wk\text{-}r\,\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A} \qquad cntr\text{-}r\,\frac{\Gamma \vdash \Delta, ?A, ?A}{\Gamma \vdash \Delta, ?A}$$

$$\exists\text{-}l\,\frac{\Gamma, A(a) \vdash \Delta}{\Gamma, \exists x.A(x) \vdash \Delta} \qquad \forall\text{-}l\,\frac{\Gamma, A(t) \vdash \Delta}{\Gamma, \forall x.A(x) \vdash \Delta} \qquad \exists\text{-}r\,\frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, \exists x.A(x)} \qquad \forall\text{-}r\,\frac{\Gamma \vdash \Delta, A(a)}{\Gamma \vdash \Delta, \forall x.A(x)}$$

where p is atomic, $i \in \{1, 2\}$, t is a term and the eigenvariable a does not occur free in Γ or Δ.

We do not formally include a symbol for implication but we sometimes write $A \multimap B$ as shorthand for $A^\bot \parr B$, where $A^\bot$ is the De Morgan dual of A. We often omit brackets under associativity, and when writing long implications we assume the right-most bracketing.

We will use standard terminology to track formulae in proofs, as presented in e.g. [10]. In particular, each rule has a distinguished *principal formula*, e.g. $A \parr B$ in the rule $\parr$-l (and similarly for all rules for the binary connectives) and $?A$ in the rule *cntr*-r, and *active formulae*, e.g. A and B in $\parr$-l and so on. These induce the notions of (direct) descendants and ancestors in proofs, as in [10].

2.1 Theories and systems

A *language* is a set of nonlogical symbols (i.e. constants, functions, predicates) and a *theory* is a set of closed formulae over some language. We assume that all theories contain the axioms of equality:

$$\begin{array}{l} \forall x.x = x \quad , \quad \forall x, y.(x = y \multimap y = x) \quad , \quad \forall x, y, z.(x = y \multimap y = z \multimap x = z) \\ \forall \vec{x}, \vec{y}.(\vec{x} = \vec{y} \multimap f(\vec{x}) = f(\vec{y})) \quad , \quad \forall \vec{x}, \vec{y}.(\vec{x} = \vec{y} \multimap P(\vec{x}) \multimap P(\vec{y})) \end{array} \tag{1}$$

where $\vec{x} = \vec{y}$ is shorthand for $x_1 = y_1 \otimes \cdots \otimes x_n = y_n$.

We consider *systems* of 'nonlogical' rules extending Dfn. 1, which we write as follows,

$$init\,\frac{}{\vdash A} \qquad (R)\,\frac{\{!\Gamma, \Sigma_i \vdash \Delta_i, ?\Pi\}_{i \in I}}{!\Gamma, \Sigma' \vdash \Delta', ?\Pi}$$

where, in each rule (R), I is a finite possibly empty set (indicating the number of premises) and we assume the following conditions and terminology:

[3] We consider a two-sided system since it is more intuitive for certain nonlogical rules, e.g. induction, and also convenient for the witness function method we use in Sect. 7.

1. In (R) the formulas of Σ', Δ' are called *principal*, those of Σ_i, Δ_i are called *active*, and those of $!\Gamma, ?\Pi$ are called *context formulas*. In *init* A is called a principal formula.
2. Each rule (R) comes with a list $a_1, \ldots, a_k$ of eigenvariables such that each a_j appears in exactly one Σ_i, Δ_i (so in some active formulas of exactly one premise) and does not appear in Σ', Δ' or $!\Gamma, ?\Pi$.
3. A system $\mathcal{S}$ of rules must be closed under substitutions of free variables by terms (where these substitutions do not contain the eigenvariables a_j in their domain or codomain).
4. In (R) the sequent Σ' (resp. Δ') does not contain any formula of the shape $?B$ (resp. $!B$), and in *init* the formula A is not of the form $!B$.

Conditions 2 and 3 are standard requirements for nonlogical rules, independently of the logical setting, cf. [5]. Condition 2 reflects the intuitive idea that, in our nonlogical rules, we often need a notion of *bound* variables in the active formulas (typically for induction rules), for which we rely on eigenvariables. Condition 3 is needed for our proof system to admit elimination of cuts on quantified formulas. Condition 4 is peculiar to our linear logic setting in order to carry out certain proof-theoretic manipulations for the free-cut elimination argument in Sect. 3.

Observe that *init* rules can actually be seen as particular cases of (R) rules, with no premise, so in the following we will only consider (R) rules.

To each theory $\mathcal{T}$ we formally associate the system of *init* rules $\vdash A$ for each $A \in \mathcal{T}$.[4] A proof in such a system will be called a *$\mathcal{T}$-proof*, or just proof when there is no risk of confusion.

▶ Remark (Semantics). The models we consider are usual Henkin models, with linear connectives interpreted by their classical counterparts. Consequently, we do not have any completeness theorem for our theories, but we do have soundness.

2.2 Some basic proof-theoretic results

We briefly survey some well-known results for theories of linear logic.

A rule is *invertible* if each of its upper sequents is derivable from its lower sequent.

▶ **Proposition 2** (Invertible rules, folklore). *The rules $\otimes$-l, $\parr$-r, $\oplus$-l, $\&$-r, $\exists$-l, $\forall$-r are invertible.*

We will typically write c-inv to denote the inverse derivation for a logical symbol c.

We also rely on the following result, which is also folklore but appeared before in [2].

▶ **Theorem 3** (Deduction, folklore). *For any theory $\mathcal{T}$ and closed formula A, $\mathcal{T} \cup \{A\}$ proves B if and only if $\mathcal{T}$ proves $!A \multimap B$.*

Due to these results notice that, in place of the equality axioms, we can work in a quantifier-free system of rules:

▶ **Proposition 4** (Equality rules). (1) *is equivalent to the following system of rules,*

$$\frac{}{\vdash t = t} \quad \frac{}{s = t \vdash t = s} \quad \frac{}{r = s, s = t \vdash r = t} \quad \frac{}{\vec{s} = \vec{t} \vdash f(\vec{s}) = f(\vec{t})} \quad \frac{}{\vec{s} = \vec{t}, P(\vec{s}) \vdash P(\vec{t})}$$

where r, s, t range over terms.

[4] Notice that this naively satisfies condition 3 since theories consist of only closed formulae.

3 Free-cut elimination in linear logic

We first define which cut instances may remain in proofs after free-cut elimination.

Since our nonlogical rules may have many principal formulae on which cuts may be anchored, we need a slightly more general notion of principality.

▶ **Definition 5.** We define the notions of *hereditarily principal formula* and *anchored cut* in a $\mathcal{S}$-proof, for a system $\mathcal{S}$, by mutual induction as follows:

- A formula A in a sequent $\Gamma \vdash \Delta$ is *hereditarily principal* for a rule instance (S) if either (i) the sequent is in the conclusion of (S) and A is principal in it, or (ii) the sequent is in the conclusion of an anchored cut, the direct ancestor of A in the corresponding premise is hereditarily principal for the rule instance (S), and the rule (S) is nonlogical.
- A cut-step is an *anchored cut* if the two occurrences of its cut-formula A in each premise are hereditarily principal for nonlogical steps, or one is hereditarily principal for a nonlogical step and the other one is principal for a logical step.

A cut which is not anchored will also be called a *free-cut*.

As a consequence of this definition, an anchored cut on a formula A has the following properties:

- At least one of the two premises of the cut has above it a sub-branch of the proof which starts (top-down) with a nonlogical step (R) with A as one of its principal formulas, and then a sequence of anchored cuts in which A is part of the context.
- The other premise is either of the same form or is a logical step with principal formula A.

Due to condition 4 in Sect. 2, we have the following:

▶ **Lemma 6.** *A formula occurrence A on the LHS (resp. RHS) of a sequent and hereditarily principal for a nonlogical rule (R) cannot be of the form $A = ?A'$ (resp. $A = !A'$).*

Now we can state the main result of this section:

▶ **Theorem 7** (Free-cut elimination)**.** *Given a system $\mathcal{S}$, any $\mathcal{S}$-proof π can be transformed into a $\mathcal{S}$-proof π' with same end sequent and without any free-cut.*

The proof proceeds in a way similar to the classical proof of cut elimination for linear logic, but eliminating only free-cuts and verifying compatibility with our notion of nonlogical rule, in particular for the commutation cases.

First, observe that the only rules in which there is a condition on the context are the following ones: $(\forall\text{-}r)$, $(\exists\text{-}l)$, $(!\text{-}r)$, $(?\text{-}l)$, (R). These are thus the rules for which the commutation with cut steps are not straightforward. Commutations with logical rules other than $(!\text{-}r)$, $(?\text{-}l)$ are done in the standard way, as in pure linear logic:[5]

▶ **Lemma 8** (Standard commutations)**.** *Any logical rule distinct from $(!\text{-}r)$, $(?\text{-}l)$ can be commuted under a cut. If the logical rule is binary this may produce two cuts, each in a separate branch.*

For rules $(!\text{-}r)$, $(?\text{-}l)$, (R) we establish our second key lemma:

[5] Note that, for the $(\forall\text{-}r)$, $(\exists\text{-}l)$ rules, there might also be a global renaming of eigenvariables if necessary.

▶ **Lemma 9** (Key commutations). *A cut of the following form, where $?A$ is not principal for (R), can be commuted above the (R) step:*

$$\dfrac{\dfrac{\{!\Gamma, \Sigma_i \vdash \Delta_i, ?A, ?\Pi\}_{i\in I}}{!\Gamma, \Sigma' \vdash \Delta', ?A, ?\Pi}\,(R) \qquad ?A, !\Gamma' \vdash ?\Pi'}{!\Gamma', \Gamma, \Sigma' \vdash \Delta', ?A, ?\Pi, ?\Pi'}\,cut$$

Similarly if (R) is replaced with $(!\text{-}r)$, with $?A$ in its RHS context, and also for the symmetric situations: cut on the LHS of the conclusion of an (R) or a $(?\text{-}l)$ step on a (non-principal) formula $!A$, with a sequent $!\Gamma' \vdash ?\Pi', !A$.

Proof. The derivation is transformed as follows:

$$\dfrac{\dfrac{!\Gamma, \Sigma_i \vdash \Delta_i, ?A, ?\Pi \qquad ?A, !\Gamma' \vdash ?\Pi'}{\{!\Gamma', !\Gamma, \Sigma_i \vdash \Delta_i, ?\Pi, ?\Pi'\}_{i\in I}}\,cut}{!\Gamma', !\Gamma, \Sigma' \vdash \Delta', ?\Pi, ?\Pi'}\,(R)$$

Here if an eigenvariable in Σ_i, Δ_i happens to be free in $!\Gamma', ?\Pi'$ we rename it to avoid the collision, which is possible because by condition 2 on nonlogical rules these eigenvariables do not appear in Σ', Δ' or $!\Gamma, ?\Pi$. So the occurrence of (R) in this new subderivation is valid.

Similarly for the symmetric derivation with a cut on the LHS of the conclusion of an (R) on a formula $!A$. The analogous situations with rules $(!\text{-}r)$ and $(?\text{-}l)$ are handled in the same way, as usual in linear logic. ◀

Now we can prove the main free-cut elimination result:

Proof sketch of Thm. 7. Given a cut step c in a proof π, we call *degree* $\deg(c)$ the number of connectives and quantifiers of its cut-formula. Now the *degree* of π, $\deg(\pi)$, is the multiset of the degrees of its non-anchored cuts. We consider the usual Dershowitz-Manna ordering on multisets of natural numbers [12].[6] The proof proceeds by induction on $\deg(\pi)$. For a given degree we proceed with a sub-induction on the *height* $h(\pi)$ of the proof.

Consider a proof π of non-null degree. We want to show how to reduce it to a proof of strictly lower degree. Consider a top-most non-anchored cut c in π, i.e. such that there is no non-anchored cut above c. Let us call A the cut-formula, and (S_1) (resp. (S_2)) the rule above the left (resp. right) premise of c.

$$\dfrac{\dfrac{}{\Gamma \vdash \Delta, A}\,S_1 \qquad \dfrac{}{\Sigma, A \vdash \Pi}\,S_2}{\Gamma, \Sigma \vdash \Delta, \Pi}\,c\ cut$$

Intuitively we proceed as follows: if A is not hereditarily principal in one of its premises we try to commute c with the rule along its left premise (S_1), and if not possible then commute it with the rule along its right premise (S_2), by Lemmas 6, 8 and 9. If A is hereditarily principal in both premises we proceed with a cut-elimination step, as in standard linear logic. For this second step, the delicate part is the elimination of exponential cuts, for which we use a big-step reduction. This works because the contexts in the nonlogical rules (R) are marked with ! (resp. ?) on the LHS (resp. RHS). ◀

[6] Let $M, N : \mathbb{N} \to \mathbb{N}$ be two multisets of natural numbers. Then $M < N$ if $M \neq N$ and, whenever $M(x) > N(x)$ there is some $y > x$ such that $N(y) > M(y)$. When M and N are finite, i.e. have finite support, $<$ is well-founded.

4 A variant of arithmetic in linear logic

For the remainder of this article we will consider an implementation of arithmetic in the sequent calculus based on the theory $\mathcal{A}_2^1$ of Bellantoni and Hofmann in [7]. The axioms that we present are obtained from $\mathcal{A}_2^1$ by using linear logic connectives in place of their classical analogues, calibrating the use of additives or multiplicatives in order to be compatible with the completeness and witnessing arguments that we present in Sects. 6 and 7. We also make use of free variables and the structural delimiters of the sequent calculus to control the logical complexity of nonlogical rules.

We will work in the *affine* variant of linear logic, which validates weakening: $(A \otimes B) \multimap A$. There are many reasons for this; essentially it does not have much effect on complexity while also creating a more robust proof theory. For example it induces the equivalence: $!(A \otimes B) \equiv (!A \otimes !B)$.[7]

4.1 Axiomatisation and an equivalent rule system

We consider the language $\mathcal{L}$ consisting of the constant symbol ε, unary function symbols $\mathsf{s}_0, \mathsf{s}_1$ and the predicate symbol W, together with function symbols f, g, h etc. $\mathcal{L}$-structures are typically extensions of $\mathbb{W} = \{0,1\}^*$, in which $\varepsilon, \mathsf{s}_0, \mathsf{s}_1$ are intended to have their usual interpretations. The W predicate is intended to indicate those elements of the model that are binary words (in the same way as Peano's N predicate indicates those elements that are natural numbers).

As an abbreviation, we write $W(\vec{t})$ for $\bigotimes_{i=1}^{|\vec{t}|} W(t_i)$.

▶ Remark (Interpretation of natural numbers). Notice that the set $\mathbb{N}^+$ of positive integers is $\mathcal{L}$-isomorphic to $\mathbb{W}$ under the interpretation $\{\varepsilon \mapsto 1, \mathsf{s}_0(x) \mapsto 2x, \mathsf{s}_1(x) \mapsto 2x+1\}$, so we could equally consider what follows as theories over $\mathbb{N}^+$.

The 'basic' axioms are essentially the axioms of Robinson arithmetic (or Peano Arithmetic without induction) without axioms for addition and multiplication. Let us write $\forall x^W.A$ for $\forall x.(W(x) \multimap A)$ and $\exists x^W.A$ for $\exists x.(W(x) \otimes A)$. We use the abbreviations $\forall x^{!W}$ and $\exists x^{!W}$ similarly.

▶ **Definition 10** (Basic axioms). The theory *BASIC* consists of the following axioms:

$W(\varepsilon)$	$\forall x^W.(\varepsilon \neq \mathsf{s}_0 x \otimes \varepsilon \neq \mathsf{s}_1 x)$	$\forall x^W.\mathsf{s}_0 x \neq \mathsf{s}_1 x$
$\forall x^W.W(\mathsf{s}_0 x)$	$\forall x^W, y^W.(\mathsf{s}_0 x = \mathsf{s}_0 y \multimap x = y)$	$\forall x^W.(x = \varepsilon \oplus \exists y^W.x = \mathsf{s}_0 y \oplus \exists y^W.x = \mathsf{s}_1 y)$
$\forall x^W.W(\mathsf{s}_1 x)$	$\forall x^W, y^W.(\mathsf{s}_1 x = \mathsf{s}_1 y \multimap x = y)$	$\forall x^W.(W(x) \otimes W(x))$

These axioms insist that, in any model, the set induced by $W(x)$ has the free algebra $\mathbb{W}$ as an initial segment. Importantly, there is also a form of contraction for the W predicate. We will consider theories over *BASIC* extended by induction schemata:

▶ **Definition 11** (Induction). The *(polynomial) induction* axiom schema, *PIND*, consists of the following axioms,

$$A(\varepsilon) \multimap !(\forall x^{!W}.(A(x) \multimap A(\mathsf{s}_0 x))) \multimap !(\forall x^{!W}.(A(x) \multimap A(\mathsf{s}_1 x))) \multimap \forall x^{!W}.A(x)$$

for each formula $A(x)$.

For a class Ξ of formulae, Ξ-*PIND* denotes the set of induction axioms when $A(x) \in \Xi$.

We write $I\Xi$ to denote the theory consisting of *BASIC* and Ξ-*PIND*.

[7] Notice that the right-left direction is already valid in usual linear logic, but the left-right direction requires weakening.

We use the terminology 'polynomial induction' to maintain consistency with the bounded arithmetic literature, e.g. in [9], where it is distinguished from induction on the *value* of a string (construed as a natural number). The two forms have different computational behaviour, specifically with regards to complexity, but we will restrict attention to *PIND* throughout this work, and thus may simply refer to it as 'induction'.

▶ **Proposition 12** (Equivalent rules). *BASIC is equivalent to the following set of rules,*

$$W_\varepsilon\ \frac{}{\vdash W(\varepsilon)} \qquad W_0\ \frac{}{W(t)\vdash W(\mathsf{s}_0 t)} \qquad \varepsilon_0\ \frac{}{W(t)\vdash \varepsilon \neq \mathsf{s}_0 t} \qquad \mathsf{s}_0\ \frac{}{W(s), W(t), \mathsf{s}_0 s = \mathsf{s}_0 t \vdash s = t}$$

$$inj\ \frac{}{W(t)\vdash \mathsf{s}_0 t \neq \mathsf{s}_1 t} \qquad W_1\ \frac{}{W(t)\vdash W(\mathsf{s}_1 t)} \qquad \varepsilon_1\ \frac{}{W(t)\vdash \varepsilon \neq \mathsf{s}_1 t} \qquad \mathsf{s}_1\ \frac{}{W(s), W(t), \mathsf{s}_1 s = \mathsf{s}_1 t \vdash s = t}$$

$$surj\ \frac{}{W(t)\vdash t = \varepsilon \oplus \exists y^W.t = \mathsf{s}_0 y \oplus \exists y^W.t = \mathsf{s}_1 y} \qquad W_{cntr}\ \frac{}{W(t)\vdash W(t)\otimes W(t)}$$

and PIND is equivalent to,

$$PIND\ \frac{!W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_0 a), ?\Delta \quad !W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_1 a), ?\Delta}{!W(t), !\Gamma, A(\varepsilon)\vdash A(t), ?\Delta} \tag{2}$$

where, in all cases, t varies over arbitrary terms and the eigenvariable a does not occur in the lower sequent of the PIND rule.

Note, in particular, that since this system of rules is closed under substitution of terms for free variables, free-cut elimination, Thm. 7, applies.

When converting from a *PIND* axiom instance to a rule instance (or vice-versa) the induction formula remains the same. For this reason when we consider theories that impose logical restrictions on induction we can use either interchangeably.

▶ Remark. Usually the induction axiom is also equivalent to a formulation with a designated premise for the base case:

$$\frac{!\Gamma\vdash A(\varepsilon) \quad !W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_0 a), ?\Delta \quad !W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_1 a), ?\Delta}{!W(t), !\Gamma\vdash A(t), ?\Delta} \tag{3}$$

However, this is not true in the linear logic setting since the proof that (3) simulates (2) above relies on contraction on the formula $A(\varepsilon)$, which is not in general available. Therefore (3) is somewhat weaker than (2), and is in fact equivalent to a version of the induction axiom with $!A(\varepsilon)$ in place of $A(\varepsilon)$. This distinction turns out to be crucial in Sect. 6, namely when proving the convergence of functions defined by predicative recursion on notation.

4.2 Provably convergent functions

As in the work of Bellantoni and Hofmann [7] and Leivant before [20], our model of computation is that of Herbrand-Gödel style *equational specifications*. These are expressive enough to define every partial recursive function, which is the reason why we also need the W predicate to have a meaningful notion of 'provably convergent function'.

▶ **Definition 13** (Equational specifications and convergence). An *equational specification* (ES) is a set of equations between terms. We say that an ES is *coherent* if the equality between any two distinct ground terms cannot be proved by equational logic.

The *convergence statement* $Conv(f, \mathcal{E})$ for an equational specification $\mathcal{E}$ and a function symbol f (that occurs in $\mathcal{E}$) is the following formula:

$$\bigotimes_{A\in\mathcal{E}} !\forall \vec{x}.A \ \multimap\ \forall \vec{x}^{!W}.W(f(\vec{x}))$$

The notion of coherence appeared in [20] and it is important to prevent a convergence statement from being a vacuous implication. In this work we will typically consider only coherent ESs, relying on the following result which is also essentially in [20]:

▶ **Proposition 14.** *The universal closure of a coherent ES $\mathcal{E}$ has a model satisfying BASIC + PIND.*

One issue is that a convergence statement contains universal quantifiers, which is problematic for the extraction of functions by the witness function method later on. We avoid this problem by appealing to the deduction theorem and further invertibility arguments:

Let us write $\overline{\mathcal{E}}$ for the closure of a specification $\mathcal{E}$ under substitution of terms for free variables.

▶ **Lemma 15.** *A system $\mathcal{S}$ proves $Conv(f, \mathcal{E})$ if and only if $\mathcal{S} \cup \overline{\mathcal{E}}$ proves $!W(\vec{a}) \vdash W(f(\vec{a}))$.*

Proof sketch. By deduction, Thm. 3, and invertibility arguments. ◀

Notice that the initial rules from $\overline{\mathcal{E}}$ are also closed under term substitution, and so compatible with free-cut elimination, and that $\overline{\mathcal{E}}$ and $W(\vec{a}) \vdash W(f(\vec{a}))$ are free of negation and universal quantifiers.

4.3 W-guarded quantifiers, rules and cut-reduction cases

We consider a quantifier hierarchy here analogous to the arithmetical hierarchy, where each class is closed under positive multiplicative operations. In the scope of this work we are only concerned with the first level:

▶ **Definition 16.** We define $\Sigma_0^{W^+}$ as the class of multiplicative formulae that are free of quantifiers where W occurs positively.[8] The class $\Sigma_1^{W^+}$ is the closure of $\Sigma_0^{W^+}$ by $\exists$, $\parr$ and $\otimes$.

For the remainder of this article we mainly work with the theory $I\Sigma_1^{W^+}$, i.e. *BASIC* + $\Sigma_1^{W^+}$*-PIND*.

It will be useful for us to work with proofs using the 'guarded' quantifiers $\forall x^W$ and $\exists x^W$ in place of their unguarded counterparts, in particular to carry out the argument in Sect. 7. Therefore we define the following rules, which are already derivable:

$$\frac{\Gamma, W(a) \vdash \Delta, A(a)}{\Gamma \vdash \Delta, \forall x^W.A(x)} \qquad \frac{\Gamma, A(t) \vdash \Delta}{\Gamma, W(t), \forall x^W A(x) \vdash \Delta} \qquad \frac{\Gamma, W(a), A(a) \vdash \Delta}{\Gamma, \exists x^W A(x) \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma, W(t) \vdash \Delta, \exists x^W.A(x)}$$

We now show that these rules are compatible with free-cut elimination.

▶ **Proposition 17.** *Any cut between the principal formula of a quantifier rule above and the principal formula of a logical step is reducible.*

Proof. For a cut on $\forall x^W.A(x)$, the reduction is obtained by performing successively the two reduction steps for the $\forall$ and $\multimap$ connectives. The case of $\exists x^W A(x)$ is similar. ◀

▶ **Corollary 18** (Free-cut elimination for guarded quantifiers)**.** *Given a system $\mathcal{S}$, any $\mathcal{S}$-proof π using $\exists x^W$ and $\forall x^W$ rules can be transformed into free-cut free form.*

As a consequence of this Corollary observe that any $I\Sigma_1^{W^+}$-proof can be transformed into a proof which is free-cut free and whose formulas contain only $\exists x^W$ quantifiers.

[8] Since our proof system is in De Morgan normal form, this is equivalent to saying that there is no occurrence of $W^\perp$.

5 Bellantoni-Cook characterisation of polynomial-time functions

We recall the Bellantoni-Cook algebra BC of functions defined by *safe* (or *predicative*) recursion on notation [6]. These will be employed for proving both the completeness (all polynomial time functions are provably convergent) and the soundness result (all provably total functions are polynomial time) of $I\Sigma_1^{W^+}$. We consider function symbols f over the domain $\mathbb{W}$ with sorted arguments $(\vec{u};\vec{x})$, where the inputs $\vec{u}$ are called *normal* and $\vec{x}$ are called *safe*.

▶ **Definition 19** (BC programs). BC is the set of functions generated as follows:

1. The constant functions ε^k which takes k arguments and outputs $\varepsilon \in \mathbb{W}$.
2. The projection functions $\pi_k^{m,n}(x_1,\ldots,x_m;x_{m+1},\ldots,x_{m+n}) := x_k$ for $n,m \in \mathbb{W}$ and $1 \le k \le m+n$.
3. The successor functions $\mathsf{s}_i(;x) := xi$ for $i = 0,1$.
4. The predecessor function $p(;x) := \begin{cases} \varepsilon & \text{if } x = \varepsilon \\ x' & \text{if } x = x'i \end{cases}$.
5. The conditional function

$$C(;\varepsilon,y_\varepsilon,y_0,y_1) := y_\varepsilon \quad C(;x0,y_\varepsilon,y_0,y_1) := y_0 \quad C(;x1,y_\varepsilon,y_0,y_1) := y_1$$

6. Predicative recursion on notation (PRN). If g,h_0,h_1 are in BC then so is f defined by,

$$\begin{array}{rcl} f(0,\vec{v};\vec{x}) & := & g(\vec{v};\vec{x}) \\ f(\mathsf{s}_i u,\vec{v};\vec{x}) & := & h_i(u,\vec{v};\vec{x},f(u,\vec{v};\vec{x})) \end{array}$$

   for $i = 0,1$, so long as the expressions are well-formed.
7. Safe composition. If $g,\vec{h},\vec{h}'$ are in BC then so is f defined by,

$$f(\vec{u};\vec{x}) \quad := \quad g(\vec{h}(\vec{u};);\vec{h}'(\vec{u};\vec{x}))$$

   so long as the expression is well-formed.

We will implicitly identify a BC function with the equational specification it induces. The main property of BC programs is:

▶ **Theorem 20** ([6]). *The class of functions representable by BC programs is* **FP**.

Actually this property remains true if one replaces the PRN scheme by the following more general simultaneous PRN scheme [8]:

$(f^j)_{1\le j\le n}$ are defined by simultaneous PRN scheme from $(g^j)_{1\le j\le n}$, $(h_0^j,h_1^j)_{1\le j\le n}$ if for $1 \le j \le n$ we have:

$$\begin{array}{rcl} f^j(0,\vec{v};\vec{x}) & := & g^j(\vec{v};\vec{x}) \\ f^j(\mathsf{s}_i u,\vec{v};\vec{x}) & := & h_i^j(u,\vec{v};\vec{x},\vec{f}(u,\vec{v};\vec{x})) \end{array}$$

for $i = 0,1$, so long as the expressions are well-formed.

Consider a well-formed expression t built from function symbols and variables. We say that a variable y occurs *hereditarily safe* in t if, for every subexpression $f(\vec{r};\vec{s})$ of t, the terms in $\vec{r}$ do not contain y. For instance y occurs hereditarily safe in $f(u;y,g(v;y))$, but not in $f(g(v;y);x)$.

▶ **Proposition 21** (Properties of BC programs). *We have the following properties:*

1. *The identity function is in BC.*

2. *Let t be a well-formed expression built from BC functions and variables, denote its free variables as $\{u_1, \ldots, u_n, x_1, \ldots, x_k\}$, and assume for each $1 \leq i \leq k$, x_i is hereditarily safe in t. Then the function $f(u_1, \ldots, u_n; x_1, \ldots, x_k) := t$ is in BC.*
3. *If f is a BC function, then the function $g(\vec{u}, v; \vec{x})$ defined as $f(\vec{u}; v, \vec{x})$ is also a BC program.*

6 Convergence of Bellantoni-Cook programs in $I\Sigma_1^{W^+}$

In this section we show that $I\Sigma_0^{W^+}$, and so also $I\Sigma_1^{W^+}$, proves the convergence of any equational specification induced by a BC program, hence any function in **FP**. The underlying construction of the proof here is similar in spirit to those occurring in [11] and [20]. In fact, like in those works, only quantifier-free positive induction is required, but here we moreover must take care to respect additive and multiplicative behaviour of linear connectives.

We will assume the formulation of BC programs with regular PRN, not simultaneous PRN.

▶ **Theorem 22.** *If $\mathcal{E}$ is a BC program defining a function f, then $I\Sigma_0^{W^+}$ proves $Conv(f, \mathcal{E})$.*

Proof sketch. We appeal to Lemma 15 and show that $\overline{\mathcal{E}} \cup I\Sigma_0^{W^+}$ proves $\forall \vec{u}^{!W}.\forall \vec{x}^{W}.W(f(\vec{u};\vec{x}))$. We proceed by induction on the structure of a BC program for f, and sketch only the key cases here.

Suppose $f(u, \vec{v}; \vec{x})$ is defined by PRN from functions $g(\vec{v}; \vec{x}), h_i(u, \vec{v}; \vec{x}, y)$. From the inductive hypothesis for g, we construct the following proof,

$$\beta\,\dfrac{\alpha\,\dfrac{\vdash \forall \vec{v}^{!W}.\forall \vec{x}^{W}.W(g(\vec{v};\vec{x}))}{!W(\vec{v}), W(\vec{x}) \vdash W(g(\vec{v};\vec{x}))}}{!W(\vec{v}), W(\vec{x}) \vdash W(f(\varepsilon, \vec{v}; \vec{x}))} \tag{4}$$

where α is purely logical and β is obtained from $\overline{\mathcal{E}}$ and equality. We also construct the proofs,

$$\otimes\text{-}l\,\dfrac{cntr\text{-}l\,\dfrac{\otimes\text{-}r\,\dfrac{id\,\dfrac{}{W(\vec{x}) \vdash W(\vec{x})} \quad \beta\,\dfrac{\alpha\,\dfrac{\vdash \forall u^{!W}, \vec{v}^{!W}.\forall \vec{x}^{W}, y^{W}.W(h_i(u,\vec{v};\vec{x},y))}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(u,\vec{v};\vec{x})) \vdash W(h_i(u,\vec{v};\vec{x},f(u,\vec{v};\vec{x})))}}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(u,\vec{v};\vec{x})) \vdash W(f(\mathsf{s}_i u, \vec{v}; \vec{x}))}}{!W(a), !W(\vec{v}), W(\vec{x}), W(\vec{x}), W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i a, \vec{v}; \vec{x}))}}{!W(a), !W(\vec{v}), W(\vec{x}), W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i a, \vec{v}; \vec{x}))}}{!W(a), !W(\vec{v}), W(\vec{x}) \otimes W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i a, \vec{v}; \vec{x}))} \tag{5}$$

from the inductive hypotheses for h_i, where α and β are similar to before. Finally we compose these proofs as follows:

$$\forall\text{-}r\,\dfrac{cntr\text{-}l\,\dfrac{cut\,\dfrac{!W(\vec{v}), W(\vec{x}) \vdash W(f(\varepsilon,\vec{v};\vec{x})) \qquad wk\,\dfrac{\otimes\text{-}inv\,\dfrac{PIND\,\dfrac{\{!W(a), !W(\vec{v}), W(\vec{x})W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i u, \vec{v}; \vec{x}))\}_i}{!W(u), !W(\vec{v}), W(\vec{x}) \otimes W(f(\varepsilon,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(\varepsilon,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(\varepsilon,\vec{v};\vec{x})) \vdash W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), !W(\vec{v}), W(\vec{x}), W(\vec{x}) \vdash W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), W(\vec{x}) \vdash W(f(u,\vec{v};\vec{x}))}}{\vdash \forall u^{!W}, \vec{v}^{!W}.\forall \vec{x}^{W}.W(f(u,\vec{v};\vec{x}))}$$

for $i = 0, 1$, where the steps $\otimes$-inv are obtained from invertibility of $\otimes$-l.

Safe compositions are essentially handled by many cut steps, using α and β like derivations again and, crucially, left-contractions on both $!W$ and W formulae.[9] The initial functions are routine. ◀

7 Witness function method

We now prove the converse to the last section: any provably convergent function in $I\Sigma_1^{W^+}$ is polynomial-time computable,

using the witness function method (WFM) [9].

The WFM differs from realisability and Dialectica style witnessing arguments mainly since it does not require functionals at higher type. Instead a translation is conducted directly from a proof in De Morgan normal form, i.e. with negation pushed to the atoms, relying on classical logic.

The combination of De Morgan normalisation and free-cut elimination plays a similar role to the double-negation translation, and this is even more evident in our setting where the transformation of a classical proof to free-cut free form can be seen to be a partial 'constructivisation' of the proof. As well as eliminating the (nonconstructive) occurrences of the $\forall$-right rule, as usual for the WFM, the linear logic refinement of the logical connectives means that right-contraction steps are also eliminated. This is important due to the fact that we are in a setting where programs are equational specifications, not formulae (as in bounded arithmetic [9]) or combinatory terms (as in applicative theories [11]), so we cannot in general decide atomic formulae.

7.1 The translation

We will associate to each (free-cut free) proof of a convergence statement in $I\Sigma_1^{W^+}$ a function on $\mathbb{W}$ defined by a BC program. In the next section we will show that this function satisfies the equational specification of the convergence statement.

▶ **Definition 23** (Typing). To each $(\forall, ?)$-free W^+-formula A we associate a sorted tuple of variables $\mathtt{t}(A)$, intended to range over $\mathbb{W}$, as follows:

$$\begin{array}{llll} \mathtt{t}(W(t)) := (;x^{W(t)}) & \mathtt{t}(s \neq t) := (;x^{s\neq t}) & \mathtt{t}(A \star B) := (\vec{u},\vec{v};\vec{x},\vec{y}) \\ \mathtt{t}(s = t) := (;x^{s=t}) & \mathtt{t}(!A) := (\vec{u},\vec{x};) & \mathtt{t}(\exists x^W.A) := (\vec{u};\vec{x},y) \end{array}$$

where $\mathtt{t}(A) = (\vec{u};\vec{x})$, $\mathtt{t}(B) = (\vec{v};\vec{y})$ and $\star \in \{\parr, \otimes, \oplus, \&\}$.

For a sorted tuple $(u_1, \ldots u_m; x_1, \ldots, x_n)$ we write $|(\vec{u};\vec{x})|$ to denote its length, i.e. $m+n$. This sorted tuple corresponds to input variables, normal and safe respectively.

Let us fix a particular (coherent) equational specification $\mathcal{E}(f)$. Rather than directly considering $I\Sigma_1^{W^+}$-proofs of $Conv(f,\mathcal{E})$, we will consider a $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$ sequent proof of $!W(\vec{x}) \vdash W(f(\vec{x}))$, under Lemma 15. Free-cut elimination crucially yields strong regularity properties for proofs:

▶ **Proposition 24** (Freedom). *A free-cut free $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$ sequent proof of $!W(\vec{x}) \vdash W(f(\vec{x}))$ is:*

1. *Free of any negative occurrences of W.*
2. *Free of any $\forall$ symbols.*

[9] In the latter case, strictly speaking, we mean cuts against W_{cntr}.

3. *Free of any ? symbols.*
4. *Free of any $\oplus$ or & steps.*[10]

For this reason we can assume that $\mathtt{t}$ is well-defined for all formulae occurring in a free-cut free proof of convergence, and so we can proceed with the translation from proofs to BC programs.

▶ **Definition 25** (Translation). We give a translation from a free-cut free $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$ proof π, satisfying properties 1, 2, 3, 4 of Prop. 24 above, of a sequent $\Gamma \vdash \Delta$ to BC programs for a tuple of functions $\vec{f}^{\pi}$ with arguments $\mathtt{t}\,(\bigotimes \Gamma)$ such that $|\vec{f}^{\pi}| = |\mathtt{t}\,(\parr \Delta)\,|$.

The translation is by induction on the structure of π, so we proceed by inspection of its final step.

If π is an instance of the initial rules $W_\varepsilon, \varepsilon^0, \varepsilon^1, \mathsf{s}_0, \mathsf{s}_1$ or *inj* then $\vec{f}^{\pi}$ is simply the constant function ε (possibly with dummy inputs as required by $\mathtt{t}$). If π is an $\overline{\mathcal{E}}$ or $=$ initial step it is also translated simply to ε. The initial steps W_0, W_1, *surj* and W_{cntr} are translated to $\mathsf{s}_0(;x), \mathsf{s}_1(;x), (\varepsilon, p(;x), p(;x))$ and $(id(;x), id(;x))$ respectively. Finally, suppose π is a logical initial step. If π is an instance of *id*, i.e. $p \vdash p$, then we translate it to *id*. Notice that, if π is an instance of $\bot$-l (i.e. $p, p^\bot \vdash$) or $\bot$-r (i.e. $\vdash p, p^\bot$) then p must be an equality $s = t$ for some terms s, t, since p must be atomic and, by assumption, W does not occur negatively. Therefore π is translated to tuples of ε as appropriate.

Now we consider the inductive cases. If π ends with a $\otimes$-r or $\parr$-l step then we just rearrange the tuple of functions obtained from the inductive hypothesis. If π consists of a subproof π' ending with a $\otimes$-l or $\parr$-r-step, then $\vec{f}^{\pi}$ is exactly $\vec{f}^{\pi'}$. By assumption, there are no $\oplus$, &, ? or $\forall$ steps occurring in π, and if π consists of a subproof π' followed by a $\exists$-l step then $\vec{f}^{\pi}$ is exactly the same as $\vec{f}^{\pi'}$, under possible reordering of the tuple.

Suppose π consists of a subproof π' followed by a $\exists$-r step,

$$\exists\text{-}r\ \frac{\Gamma \vdash \Delta, A(t)}{\Gamma, W(t) \vdash \Delta, \exists x^W.A(x)}$$

so by the inductive hypothesis we have functions $\vec{f}^{\Delta}, \vec{f}^{A(t)}$ with arguments $(\vec{u};\vec{x}) = \mathtt{t}(\bigotimes \Gamma)$. We define $\vec{f}^{\pi}(\vec{u};\vec{x},y)$ as $\Big(\vec{f}^{\Delta}(\vec{u};\vec{x}), id(;y), \vec{f}^{A(t)}(\vec{u};\vec{x})\Big)$.

If π consists of a subproof π' followed by a !-r step then $\vec{f}^{\pi}$ is exactly the same as $\vec{f}^{\pi'}$. If π ends with a !-l step then we just appeal to Prop. 21 to turn a safe input into a normal input.

Since there are no ? symbols in π, we can assume also that there are no *cntr*-r steps in π.[11]

If π ends with a *cntr*-l step then we duplicate some normal inputs of the functions obtained by the inductive hypothesis.

If π ends with a *cut* step whose cut-formula is free of modalities, then it can be directly translated to a safe composition of functions obtained by the inductive hypothesis, by relying on Prop. 21. Otherwise, the cut-formula must be of the form $!W(t)$ since it must directly descend from the left-hand side of an induction step, by free-cut freeness. Since the cut is

[10] Because of the *surj* rule, the proof may still contain $\oplus$ symbols, but these must be direct ancestors of some cut-step by free-cut freeness.

[11] Again, this is crucially important, since we cannot test the equality between arbitrary terms in the presence of nonlogical function symbols.

anchored, we can also assume that the cut formula is principal on the other side, i.e. π ends as follows:

$$\dfrac{\dfrac{!\Gamma \vdash W(t)}{!\Gamma \vdash !W(t)}\,{\scriptstyle !\text{-}r} \quad !W(t), \Sigma \vdash \Delta}{!\Gamma, \Sigma \vdash \Delta}\,{\scriptstyle cut}$$

where we assume there are no side-formulae on the right of the end-sequent of the left subproof for the same reason as *cntr-r*: π does not contain any occurrences of ?. By the inductive hypothesis we have functions $g(\vec{u};)$ and $\vec{h}(v, \vec{w}; \vec{x})$ where $\vec{u}$, v and $(\vec{w}; \vec{x})$ correspond to $!\Gamma$, $!W(t)$ and Σ respectively. We construct the functions $\vec{f^{\pi}}$ as follows:

$$\vec{f^{\pi}}(\vec{u}, \vec{w}; \vec{x}) \quad := \quad \vec{h}(g(\vec{u};), \vec{w}; \vec{x})$$

Notice, again, that all safe inputs on the left occur hereditarily safe on the right, and so these expressions are definable in BC by Prop. 21.

If π ends with a *wk-r* step then we just add a tuple of constant functions $\vec{\varepsilon}$ of appropriate length as dummy functions. If π ends with a *wk-l* step then we just add dummy inputs of appropriate length.

Finally, suppose π ends with a *PIND* step. Since there are no occurrences of ? in π we can again assume that there are no side formulae on the right of any induction step. Thus π ends as follows:

$$PIND\,\dfrac{!W(a), !\Gamma, A(a) \vdash A(\mathsf{s}_0 a) \quad !W(a), !\Gamma, A(a) \vdash A(\mathsf{s}_1 a)}{!W(t), !\Gamma, A(\varepsilon) \vdash A(t)}$$

By the inductive hypothesis we have functions $\vec{g}^0(u, \vec{v}; \vec{x})$ and $\vec{g}^1(u, \vec{v}; \vec{x})$ with u, $\vec{v}$ and $\vec{x}$ corresponding to $!W(a)$, $!\Gamma$ and $A(a)$ respectively. We define $\vec{f^{\pi}}$ by simultaneous predicative recursion on notation as follows:

$$\begin{array}{rcl} \vec{f^{\pi}}(\varepsilon, \vec{v}; \vec{x}) & := & \vec{x} \\ \vec{f^{\pi}}(\mathsf{s}_i u, \vec{v}; \vec{x}) & := & \vec{g}^i(u, \vec{v}; \vec{f^{\pi}}(u, \vec{v}; \vec{x})) \end{array}$$

The induction step above is the reason why we enrich the usual BC framework with a simultaneous version of PRN.

7.2 Witness predicate and extensional equivalence of functions

Now that we have seen how to extract BC functions from proofs, we show that these functions satisfy the appropriate semantic properties, namely the equational program $\mathcal{E}$ we started with. For this we turn to a quantifier-free *classical* theory, in a similar fashion to PV for S^1_2 in [9] or system T in Gödel's Dialectica interpretation. This is adequate since we only care about extensional properties of extracted functions at this stage.

We could equally use a realisability approach, as done in e.g. [11] and other works in applicative theories: since the formulae we deal with are essentially positive there is not much difference between the two approaches. Indeed here the witness predicate plays the same role as the realisability predicate in other works.

Let *IQF* be the classical theory over the language $\{\varepsilon, \mathsf{s}_0, \mathsf{s}_1, (f^k_i)\}$ obtained from the axioms $\varepsilon, \mathsf{s}_0, \mathsf{s}_1$, *inj*, *surj* and *PIND* by dropping all relativisations to W (or $!W$), replacing all linear connectives by their classical counterparts, and restricting induction to only quantifier-free formulae.

▶ **Definition 26** (Witness predicate). For formulae A, B of $I\Sigma_1^{W^+}$ satisfying properties 1, 2, 3 of Prop. 24, we define the following 'witness' predicate as a quantifier-free formula of IQF:

$$\begin{array}{rcl} Wit_{W(t)}(a) & := & a = t \\ Wit_{s=t}(a) & := & s = t \\ Wit_{s\neq t}(a) & := & s \neq t \\ Wit_{!A}(\vec{a}^A) & := & Wit_A(\vec{a}^A) \end{array} \qquad \begin{array}{rcl} Wit_{A\bullet B}(\vec{a}^A, \vec{a}^B) & := & Wit_A(\vec{a}^A) \vee Wit_B(\vec{a}^B) \\ Wit_{A\circ B}(\vec{a}^A, \vec{a}^B) & := & Wit_A(\vec{a}^A) \wedge Wit_B(\vec{a}^B) \\ Wit_{\exists x^W A}(a, \vec{a}^A) & := & Wit_A(\vec{a}^A, a) \end{array}$$

where $\bullet \in \{\parr, \oplus\}$, $\circ \in \{\otimes, \&\}$, $|\vec{a}^A| = |\mathtt{t}(A)|$ and $|\vec{a}^B| = |\mathtt{t}(B)|$.

Notice that, unlike in the bounded arithmetic setting where the W predicate is redundant (since variables are tacitly assumed to range over W), we do not parametrise the witness predicate by an assignment to the free variables of a formula. Instead these dependencies are taken care of by the explicit occurrences of the W predicate in $I\Sigma_1^{W^+}$.

▶ **Lemma 27.** *Let π be a free-cut free proof in $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$, satisfying properties 1, 2, 3, 4 of Prop. 24, of a sequent $\Gamma \vdash \Delta$. Let $\mathcal{E}^\pi$ denote the BC program for $\vec{f}^\pi$.*[12] *Then IQF proves:*

$$\left(\forall\mathcal{E} \wedge \forall\mathcal{E}^\pi \wedge Wit_{\bigotimes\Gamma}(\vec{a})\right) \to Wit_{\parr\Delta}(\vec{f}^\pi(\vec{a}))$$

where $\forall\mathcal{E}$ and $\forall\mathcal{E}^\pi$ denote the universal closures of $\mathcal{E}$ and $\mathcal{E}^\pi$ respectively.

Proof sketch. By structural induction on π, again, following the definition of $\vec{f}^\pi$.[13] ◀

Finally, we arrive at our main result, providing a converse to Thm. 22.

▶ **Theorem 28.** *For any coherent equational specification $\mathcal{E}$, if $I\Sigma_1^{W^+}$ proves $Conv(f, \mathcal{E})$ then there is a polynomial-time function g on $\mathbb{W}$ (of same arity as f) satisfying $\mathcal{E}[g/f]$.*

Proof sketch. Follows from Lemmas 15 and 27, Dfn. 26 and coherence of $\mathcal{E}$, cf. 14. ◀

8 Conclusions

As mentioned in the introduction, our motivation for this work is to commence a proof-theoretic study of first-order theories in linear logic, in particular from the point of view of complexity. To this end we proved a general form of 'free-cut elimination' that generalises forms occurring in prior works, e.g. [22]. We adapted an arithmetic of Bellantoni and Hofmann in [7] to the linear logic setting, and used the free-cut elimination result, via the witness function method [9], to prove that a fragment of this arithmetic characterises **FP**.

From the point of view of constructing theories for complexity classes, the choice of linear logic and witness function method satisfies two particular desiderata:

1. Complexity is controlled by 'implicit' means, not explicit bounds.
2. Extraction of programs relies on functions of only ground type.

From this point of view, a relevant related work is that of Cantini [11], based on an *applicative theory*, which we recently became aware of. The main difference here is the choice of model of computation: Cantini uses terms of combinatory logic, whereas we use equational specifications (ESs). As we have mentioned, this choice necessitates a different

[12] We assume that the function symbols occurring in $\mathcal{E}^\pi$ are distinct from those occurring in $\mathcal{E}$.

[13] Notice that, since we are in a classical theory, the proof of the above lemma can be carried out in an arbitrary model, by the completeness theorem, greatly easing the exposition.

proof-theoretic treatment, in particular since equality between terms is not decidable in the ES framework, hindering any constructive interpretation of the right-contraction rule. This is why Bellantoni and Hofmann require a double-negation translation into intuitionistic logic and the use of functionals at higher types, and why Leivant disregards classical logic altogether in [20]. Notice that this is precisely why our use of linear logic is important: the control of ? occurrences in a proof allows us to sidestep this problem. At the same time we are able to remain in a classical linear setting. We do not think that either model of computation is better, but rather that it is interesting to observe how such a choice affects the proof-theoretic considerations at hand.

Most works on the relationships between linear logic and complexity fit in the approach of the proofs-as-programs correspondence and study variants of linear logic with weak exponential modalities [17] [15] [18]. However, Terui considers a naïve set theory [27] that also characterises **FP** and is based on *light linear logic*.[14] His approach relies on functionals at higher type, using the propositional fragment of the logic to type the extracted functionals. Another work using linear logic to characterize complexity classes by using convergence proofs is [19] but it is tailored for second-order logic. The status of first-order theories is more developed for other substructural logics, for instance *relevant logic* [13], although we do not know of any works connecting such logics to computational complexity.

Concerning the relationship between linear logic and safe recursion, we note that an embedding of a variant of safe recursion into light linear logic has been carried studied in [25], but this is in the setting of functional computation and is quite different from the present approach. Observe, however, that a difficulty in this setting was the nonlinear treatment of safe arguments which here we manage by including in our theory an explicit contraction axiom for W.

We have already mentioned the work of Bellantoni and Hofmann [7], which was somewhat the inspiration behind this work. Our logical setting is very similar to theirs, under a certain identification of symbols, but there is a curious disconnect in our use of the additive disjunction for the *surj* axiom. They rely on just one variant of disjunction. As we said, they rely on a double-negation translation and thus functionals at higher-type.

In further work we would like to apply the free-cut elimination theorem to theories based on other models of computation, namely the formula model employed in bounded arithmetic [9]. We believe that the witness function method could be used at a much finer level in this setting,[15] and extensions of the theory for other complexity classes, e.g. the polynomial hierarchy, might be easier to obtain.

The problem of right-contraction seems to also present in work by Leivant [20], which uses equational specifications, where the restriction to positive comprehension to characterise polynomial-time only goes through in an intuitionistic setting. It would be interesting to see if a linear logic refinement could reproduce that result in the classical setting, as we did here.

[14] He also presents a cut-elimination result but, interestingly, it is entirely complementary to that which we present here: he obtains full cut-elimination since he works in a system without full exponentials and with Comprehension as the only nonlogical rule. Since the latter admits a cut-reduction step, the former ensures that cut-elimination eventually terminates by a *height-based* argument, contrary to our argument that analyses the *degrees* of cut-formulae.

[15] One reason for this is that atomic formulae are decidable, and so we can have more freedom with the modalities in induction steps.

References

1 Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Log. Comput.*, 2(3):297–347, 1992. doi:10.1093/logcom/2.3.297.

2 Arnon Avron. The semantics and proof theory of linear logic. *Theor. Comput. Sci.*, 57:161–184, 1988.

3 David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Log.*, 13(1):2, 2012.

4 Patrick Baillot and Anupam Das. Free-cut elimination in linear logic and an application to a feasible arithmetic. Preprint, 2016. URL: https://hal.archives-ouvertes.fr/hal-01316754.

5 Arnold Beckmann and Samuel R. Buss. Corrected upper bounds for free-cut elimination. *Theor. Comput. Sci.*, 412(39):5433–5445, 2011.

6 Stephen Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.

7 Stephen Bellantoni and Martin Hofmann. A new "feasible" arithmetic. *J. Symb. Log.*, 67(1):104–116, 2002.

8 Stephen J. Bellantoni. *Predicative Recursion and Computational Complexity*. PhD thesis, University of Toronto, 1992.

9 Samuel R Buss. *Bounded arithmetic*, volume 86. Bibliopolis, 1986.

10 Samuel R Buss. An introduction to proof theory. *Handbook of proof theory*, 137:1–78, 1998.

11 Andrea Cantini. Polytime, combinatory logic and positive safe induction. *Arch. Math. Log.*, 41(2):169–189, 2002.

12 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, August 1979. doi:10.1145/359138.359142.

13 Harvey Friedman and Robert K. Meyer. Whither relevant arithmetic? *J. Symb. Log.*, 57(3):824–831, 1992. doi:10.2307/2275433.

14 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.

15 Jean-Yves Girard. Light linear logic. In *Logical and Computational Complexity. Selected Papers. LCC'94.*, pages 145–176, 1994. doi:10.1007/3-540-60178-3_83.

16 Jean-Yves Girard. Light linear logic. *Inf. Comput.*, 143(2):175–204, 1998.

17 Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992.

18 Yves Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004.

19 Marc Lasson. Controlling program extraction in light logics. In *Typed Lambda Calculi and Applications – 10th International Conference, TLCA 2011, Novi Sad, Serbia, June 1-3, 2011. Proceedings*, volume 6690 of *Lecture Notes in Computer Science*, pages 123–137. Springer, 2011.

20 Daniel Leivant. A foundational delineation of poly-time. *Inf. Comput.*, 110(2):391–420, 1994.

21 Daniel Leivant. Intrinsic theories and computational complexity. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC'94, Indianapolis, Indiana, USA, 13-16 October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 1995.

22 Patrick Lincoln, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Ann. Pure Appl. Logic*, 56(1-3):239–311, 1992.

23 Jean-Yves Marion. Actual arithmetic and feasibility. In *Proceedings of Computer Science Logic (CSL 2001)*, volume 2142 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 2001.

24 Dale Miller. Overview of linear logic programming. In Thomas Ehrhard, editor, *Linear Logic in Computer Science*, pages 316–119. Cambridge University Press, 2004.

25 Andrzej S. Murawski and C.-H. Luke Ong. On an interpretation of safe recursion in light affine logic. *Theor. Comput. Sci.*, 318(1-2):197–223, 2004.

26 G. Takeuti. *Proof Theory*. North-Holland, Amsterdam, 1987. and ed.

27 Kazushige Terui. Light affine set theory: A naive set theory of polynomial time. *Studia Logica*, 77(1):9–40, 2004.