A complexity analysis of functional interpretations [1]

Mircea-Dan HERNEST [a,2] and Ulrich KOHLENBACH [b,3]

[a] *Laboratoire D'Informatique (LIX), École Polytechnique, F – 91128 Palaiseau, FRANCE,* danher@lix.polytechnique.fr

[b] *Department of Mathematics, Darmstadt University of Technology, Schlossgartenstrasse 7, D – 64289 Darmstadt, GERMANY,* kohlenbach@mathematik.tu-darmstadt.de

Abstract

We give a quantitative analysis of Gödel's functional interpretation and its monotone variant. The two have been used for the extraction of programs and numerical bounds as well as for conservation results. They apply both to (semi-)intuitionistic as well as (combined with negative translation) classical proofs. The proofs may be formalized in systems ranging from weak base systems to arithmetic and analysis (and numerous fragments of these). We give upper bounds in basic proof data on the depth, size, maximal type degree and maximal type arity of the extracted terms as well as on the depth of the verifying proof. In all cases terms of size linear in the size of the proof at input can be extracted and the corresponding extraction algorithms have cubic worst-time complexity. The verifying proofs have depth linear in the depth of the proof at input and the maximal size of a formula of this proof.

Key words: Functional interpretation, proof complexity, functionals of finite type, proof mining, program extraction from (classical) proofs, software and systems verification, combinatorial logic, computational complexity, proof–carrying code.

[1] Shorter and revised version of the BRICS Technical Report RS-03-12, see [26].
[2] Supported equally by the *Danish National Research Foundation* via BRICS, University of Aarhus, DENMARK and the *Deutsche Forschungsgemeinschaft* via GKLI, University of Munich, GERMANY. Work carried out equally at BRICS and GKLI. The revision following the referees suggestions was financed by *Project LogiCal* – Pôle Commun de Recherche en Informatique du Plateau de Saclay, CNRS, École Polytechnique, INRIA and Université Paris-Sud – FRANCE.
[3] Partly supported by the *Danish Natural Science Research Council* with the grant no. 21-02-0474. Work carried out entirely at BRICS, University of Aarhus, DENMARK.

Contents

1 Introduction

This paper investigates the complexity of the extraction algorithms for effective data (such as programs and bounds) from proofs provided by Gödel's functional (*Dialectica*) interpretation and its monotone variant. The subject of extracting programs from proofs already has a long history. The techniques used can be roughly divided in two categories according to whether they are based on cut-elimination, normalization and related methods or on so-called proof interpretations. The latter typically make use of functionals of higher type. Prominent proof interpretations are realizability interpretations, particularly Kreisel's [38] modified realizability (see [54] for a survey) and Gödel's functional interpretation (first published in [22], see [3] for a survey). The no-counterexample interpretation (*n.c.i.*) due to Kreisel [36,37] is sometimes viewed as a simplification of the functional interpretation (it uses only types of degree ≤ 2). In fact *n.c.i.* is not a real alternative since it has a bad behavior with respect to the modus ponens rule `MP`. This is overcome only if `MP` is interpreted by functional interpretation (see [33]).

Cut-elimination, normalization and the related ε-substitution method globally rebuild the given proof thereby increasing its length in a potentially non-elementary recursive way. Hyper-exponential lower-bound examples were provided by Statman [52], Orevkov [41,42] and Pudlak [44] – see also [56] and the more recent [16–18]. In contrast, proof interpretations extract witnessing terms by recursion on the given proof tree which remains essentially unchanged in its structure. The latter techniques consequently enjoy full modularity: the global realizers of a proof can be computed from realizers of lemmas used in the proof. This suggests a radically lower complexity of the procedure and a radically smaller size of the extracted programs. Even though the latter would not be in normal form [4] they can be used substantially in many ways without having to normalize them. One merely exploits properties which can be established inductively over their structure with the use of logical relations (like, e.g., Howard's [27] notion of majorizability).

Both (modified) realizability and functional interpretations are applicable to a vast variety of formal systems and provide characterizations of their provably total programs. They had originally been applied to arithmetic in all finite types. They were subsequently adapted to various fragments thereof all the way down to weak systems of bounded arithmetic [8,32,43] or – more recently – the poly-time arithmetic of [47,48]. They were extended to analysis [13,39,51], type theories [20] and fragments of set theory [6]. Gödel's functional interpretation was recently adapted to yield an extraction of Herbrand terms from ordinary first-order predicate logic proofs [19].

Realizability and functional interpretations cannot be directly applied to classical systems. A canonical manner of interpreting classical proofs would be to first translate them to intuitionistic proofs via a so-called negative translation and to subsequently apply intuitionistic proof interpretations. However this fails for (modified) realizability since it extracts empty programs from negative formulas. The problem can be partly overcome by using an additional intermediate interpretation, the so-called Friedman-Dragalin A-translation [11,15] and its variants [9] [5]. Unlike realizability interpretations, functional inter-

[4] Normalization would bring back the aforementioned complexities.

[5] See, e.g., [5,40] for examples of program extractions using this approach. One drawback of this method is the limited modularity feature: only a restricted class of lemmas can be used to build the input proof. In contrast, the techniques based

pretations are sound for the so-called Markov principle and therefore feature extraction of programs from arbitrary proofs in fairly rich classical systems, like Peano arithmetic in all finite types $\mathtt{PA}^\omega$ (see also Section 5). Hence the need for an intermediate translation is avoided when using functional interpretations. Moreover, monotone functional interpretation can extract programs from proofs $\mathcal{T} \vdash \forall x^\rho \exists y^\tau Rec(x, y)$ in highly unconstructive systems $\mathcal{T}$ which contain, e.g., the binary König lemma. Here τ is an arbitrary finite type, ρ is a finite type of degree (aka level) at most 1 and $Rec(x, y)$ is a specification which must [6] be decidable if $\mathcal{T}$ is classical (we actually take it quantifier-free). This gives functional interpretations the ability of extracting programs and other effective data (such as numerical bounds) under certain conditions from ineffective proofs (*proof mining*). Proof mining based on the monotone functional interpretation has already produced important results in computational analysis and has helped to obtain new results in mathematical analysis (see [35]).

A natural question that arises is whether such applications which were obtained *by hand* could be automated or at least computer aided by implementing functional interpretations. In order to evaluate the feasibility of such a tool it is important to investigate the complexity aspects of functional interpretations. In the present paper we obtain upper bounds on the size of the terms which express the extracted programs. The interpretation algorithms only write down the extracted terms, proceeding by recursion on the structure of the input proof, see Section 3. It follows that their running time is proportional with the size of the extracted terms. Hence we obtain the time complexity of the extraction algorithms as a consequence of our quantitative analysis. Let n denote the size of the input proof $\mathcal{P}$ and m denote the maximal size of a formula of $\mathcal{P}$. Due to the modularity of functional interpretations, these algorithms feature an almost linear time complexity, namely $O(m^2 \cdot n)$ even for classical and analytical proofs. The *almost* refers to the fact that m is much smaller than n in most practical cases. In any case this time complexity is at most $O(n^3)$, a result previously obtained by Alexi in [1] for an ad-hoc program-extraction technique for intuitionistic proofs only. Since the design of Alexi's technique was driven by the optimal-time-overhead issue, *cubic* is probably the best worst-time-complexity one can expect from any program-extraction technique. We also give upper bounds on depths of the resulting verifying proofs – this is interesting for quantitative conservation results. In particular we obtain the feasibility of $\mathtt{WKL}$–elimination for Π^0_2–sentences over primitive recursive arithmetic [7] in all finite types by means of syntactic translations.

on the Dialectica interpretation feature full modularity: the input proofs may use arbitrary lemmas. See also [24] for applications of a form of recursive realizability.

[6] This restriction is generally unavoidable for classical proofs but is not necessary for intuitionistic proofs.

[7] This had been shown for a second–order fragment independently in [23] and [2], in the latter by means of a formalized forcing technique.

Our technique is immediately implementable and in addition provides a term extraction procedure from analytical proofs. A program-extraction module based on Gödel's functional interpretation was implemented by the first author in the proof-system `MINLOG` [49]. An experimental comparison between the performance of this and the existent refined A-translation [5] extraction module is reported in [25]. The newer module performs better in that case.

There exists a research line in extractive proof theory which is aimed at characterizing the classes of proofs from which programs belonging to certain complexity classes are extracted. Usually the *feasible* complexity classes are of interest, particularly poly-time, see e.g. [8,47]. The issue of characterizing the complexity of provably total function(al)s of a theory is completely separate from the present paper's topic. We are here concerned more with the performance of the extraction algorithm rather than with the one of the extracted programs.

The monotone variant of Gödel's functional interpretation was developed by the second author in [31]. It takes into account that most applications of functional interpretation in recent years both to concrete proofs in numerical analysis and to conservation results do not actually use terms which realize the Gödel functional interpretation but terms which majorize [8] (some) realizers. Monotone functional interpretation extracts majorizing terms which are simpler than the actual realizers produced by functional interpretation. This is due to the much simpler treatment of $\mathtt{CT}\wedge$, see Proposition 3.22 and the paragraph following Definition 4.16. Also the treatment of induction axioms is much simpler, see Section 5. Moreover, the upper bound on the depth of the verifying proof is better in the monotone case if the underlying logical system fairly supports monotone functional interpretation, see Remark 4.19.

1.1 Outline of the main results

We introduce the weak base system $\mathtt{EIL}^\omega$, a short for "(weakly extensional) extended intuitionistic equality logic in all finite types". $\mathtt{EIL}^\omega$ contains only the tools which are strictly necessary for carrying out the functional interpretation even for the most rudimentary intuitionistic systems. We present upper bounds for the following quantitative measures of realizing/majorizing terms t extracted from proofs $\mathcal{P}$ in both semi-intuitionistic [9] and classical systems based on $\mathtt{EIL}^\omega$ up to the analytical system $\mathtt{PA}^\omega + \mathtt{AC}_0 + \mathtt{WKL}$:

- the maximal degree (arity) of a subterm of t, denoted *mdg* (*mar*);
- the depth of t, denoted d (assuming a tree representation of terms);

[8] Majorization is understood in the sense of Howard [27] mentioned before.
[9] Here *semi-intuitionistic* means intuitionistic plus a version of Markov's principle $\mathtt{MK}$ and independence of premises for universal premises $\mathtt{IP}_\forall$, see Section 3.1 for details.

- the size of t, denoted S and defined as the number of all constants and variables used to build t .

We also give upper bounds on the depth [10] of the verifying proof and time overhead of the extraction algorithm, here denoted ∂_v and θ respectively. For the extraction procedure we consider both the usual [22] and the monotone variant [31] of functional interpretation. We first consider a binary-tree representation for terms, see also Footnote 35. Such a representation is more intuitive and therefore provides a better exposition of the bounds for mdg, mar. However it turns out that the same extracted terms have smaller size if represented in a more economic manner using pointers [11] , see Section 3.4 . Since their definition does not depend on the term representation, the bounds for mdg and mar still hold. From Section 3.4 on it is tacitly assumed that terms are represented in the economic manner. A representation for types becomes necessary only at the moment that we are interested in the space/time overhead of the extraction algorithm, see Section 3.5 . Let us denote by ∂ the depth and by S_i, S_c, S_m the size (in the sense of Definition 3.33) of $\mathcal{P}$ and for a formula A by

- vdg (var) the maximal degree (arity) of a variable occurring in A ;
- id (fd, ld) the implication (forall, logical) depth of A, namely the maximal number of $\rightarrow$ ($\forall$, all logical constants) on a path from root to leaves in the usual tree representation of A; by $fid :\equiv max\{fd\, , id\}$;
- qs the number of all quantifiers (including [12] $\vee$) of the universal closure of A ;
- ls the number of all $\forall, \exists, \wedge, \vee, \rightarrow, \bot, =$ and free variables of A .

We prove that (relative to our underlying deductive framework $\texttt{EIL}^\omega$)

- mdg and mar do not depend on ∂; the difference between mdg (mar) and the maximal degree (arity) of a variable occurring in an axiom of $\mathcal{P}$ is linear (quadratic) in the maximal complexity of an axiom of $\mathcal{P}$;
- d is linear in the maximal logical size ls of an axiom of $\mathcal{P}$ and ∂ ;
- S is linear in the size of $\mathcal{P}$ (here we use the economic representation of terms); also exponential in ∂ and in the logarithm of the maximal logical size ls of an axiom of $\mathcal{P}$ (in contrast to the former, this holds for both the economic and the binary-tree representation of terms) ;
- ∂_v is linear in ∂ and the maximal complexity of an axiom of $\mathcal{P}$.

More precisely, for **semi-intuitionistic** proofs $\mathcal{P}$ we have the following situation (below "FI" means "functional interpretation") :

[10] Proofs are represented as trees, see also the last paragraph of Section 1.2 .
[11] It would be possible to extract other terms which have the same smaller size also in the case of binary-tree representation for terms, but the bounds for mdg, mar would no longer hold in such a case – see also the remarks following Theorem 3.37 .
[12] We must count $\vee$ among the quantifiers because functional interpretation treats disjunction as an existential quantifier.

	usual FI	monotone FI
mdg	$O(1) + vdg_o + id_o$	$O(1) + vdg_o + id_o$
mar	$O(1) + var_o + qs_o \cdot id_o$	$O(1) + var_o + qs_o \cdot id_o$
d	$O(ld_1) + qs_o \cdot \partial$	$O(1) + qs_o \cdot \partial$
S	$O(S_i)$, $O(ls_1 \cdot qs_o^{\partial})$	$O(S_m)$, $O(qs_o^{\partial})$
∂_v	$O(ld_1 + \partial)$	$O(qs_o + \partial)$
θ	$O(qs_o \cdot ls_o \cdot S_m)$	$O(qs_o \cdot ls_o \cdot S_m)$

where vdg_o, var_o, id_o, qs_o, ls_o are maxima taken over all the axioms of $\mathcal{P}$ [13] of vdg, var, id, qs and ls respectively and ld_1, ls_1 are maxima of ld, ls taken over contractions $A \to A \wedge A$ of $\mathcal{P}$.

For **classical** proofs $\mathcal{P}$ a preprocessing double–negation translation must be employed, see Section 4.1. The above upper bounds must be adapted to take it into account. The situation changes as follows. There exists $k \in \mathbb{N}$ constant (independent of $\mathcal{P}$) such that (below "FI" means "functional interpretation") :

	usual FI	monotone FI
mdg	$vdg_o + O(fid_o)$	$vdg_o + O(fid_o)$
mar	$var_o + O(qs_o \cdot fid_o)$	$var_o + O(qs_o \cdot fid_o)$
d	$O(ls_o \cdot \partial)$	$O(qs_o \cdot \partial)$
S	$O(S_c)$, $O(ls_o \cdot qs_o^{k\cdot\partial})$	$O(S_m)$, $O(qs_o^{k\cdot\partial})$
∂_v	$O(ls_o + \partial)$	$O(qs_o + \partial)$
θ	$O(qs_o \cdot ls_o \cdot S_m)$	$O(qs_o \cdot ls_o \cdot S_m)$

where vdg_o, var_o, qs_o, ls_o are maxima taken over all the axioms of $\mathcal{P}$ of vdg, var, qs and ls respectively and fid_o is the maximum of fid over all the formulas of $\mathcal{P}$.

[13] In fact it is sufficient to consider only the axioms of the transformed proof $\mathcal{P}^{\mathrm{tr}}$, see Definition 3.8 . The same holds for the subsequent definitions as well, including the classical case.

Since they are not produced by functional interpretation, we normally do not count the terms t_1, t_2 which appear in prime formulas $t_1 = t_2$ of contractions $A \to A \wedge A$ and the quantifier axioms terms as part of the realizing terms. We rather consider them as "black boxes" and use their type and free variables information only (see Definition 3.10). From a programming perspective, they may be considered as subprograms residing in libraries and made accessible to the extracted program via references. The bounds for the usual functional interpretation actually hold also if we take into account the terms mentioned above provided that instead of ld, ls one uses wd, respectively ws, where

- wd is the whole depth of A, assuming a tree representation of A where tree representations of the terms occurring in A are linked from the corresponding leaves of the usual tree representation of A;
- ws is the whole size of A, i.e., the number of all logical constants of A plus the number of all occurrences of variables and constants in A.

For mdg and mar also the maximal degree, respectively arity of constants occurring in contraction and quantifier axioms terms must be taken into account. For more details see Remark 3.28.

1.2 Notational conventions

The symbols $:\equiv$ and $\equiv$ belong to the meta-level and mean *equal by definition to* and *is identical to* respectively. The symbol $=$ is used by abuse for equality in both meta-level and formal systems. For a set M we let $M^{\leq\omega} :\equiv \cup_{n\leq\omega} M^n$. The symbol $\mathbb{N}$ denotes the set of *natural numbers*. For a function $f : M' \mapsto \mathbb{N}$ and $M \subseteq M'$, M finite, we let $f(M) :\equiv max\{f(m) \,|\, m \in M\}$. An enumeration $\mathcal{S}_1, \ldots, \mathcal{S}_n$ denotes an ordered tuple abbreviated $\underline{\mathcal{S}}$. We denote by $\{\underline{\mathcal{S}}\}$ the set corresponding to $\underline{\mathcal{S}}$, by $|\underline{\mathcal{S}}|$ the length of $\underline{\mathcal{S}}$ and by $\underline{\mathcal{S}'}, \underline{\mathcal{S}''}$ the concatenation of $\underline{\mathcal{S}'}$ and $\underline{\mathcal{S}''}$. If $\{\underline{\mathcal{S}}\} \subseteq M'$ we abbreviate by $f(\underline{\mathcal{S}}) :\equiv f(\{\underline{\mathcal{S}}\})$. If p is a permutation of $\{1, \ldots, n\}$, $\underline{\mathcal{S}^p}$ abbreviates the tuple $\mathcal{S}_{p_1}, \ldots, \mathcal{S}_{p_n}$.

Let $k_0 \in \mathbb{N}$ be a sufficiently large constant ($k_0 \equiv 10$ suffices for our purposes) [14]. For a labeled tree $\triangle$ we denote by $\partial(\triangle)$ the depth of $\triangle$ plus k_0; by $\partial_L(\triangle)$ the L depth of $\triangle$ (here L is a meta–variable for labels), i.e., the maximal number of L labels on a path from root to leaves plus k_0; by $\mathrm{Lv}(\triangle)$ the set of labels of leaves of $\triangle$ and by $\mathrm{Vt}(\triangle)$ the set of labels of all vertices of $\triangle$.

A *(formal) proof* in some logical system is a tree whose vertices are labeled

[14] The meta–constant k_0 is only needed for technical reasons. It just helps to increase the readability of the numerous upper–bound expressions from the sequel. We consider that is clarifies the exposition when including k_0 unchanged in the various computations rather than combine (and therefore loose its trace) an actual constant. The indication $k_0 \equiv 10$ just gives a hint of the order of magnitude of k_0.

with formulas, such that the leaves are labeled with axioms and assumptions and any parent vertex is labeled with the result of the application of an instance of some rule to the labels of its sons. The edges which connect the parent vertex with its sons are labeled with the name of the corresponding rule. We denote by $L(\cdot)$ the *labeling function* on vertices and edges. We call a proof *complete* if all its leaves are labeled with axioms only. Notice that an *incomplete* proof is complete in the system extended with its assumptions as axioms. We will denote proofs by $\vdash$ or ——, possibly with bounds on the depth attached, such as $\vdash_n$ for a proof of depth at most n, $n \in \mathbb{N}$.

2 The weak base system $\mathtt{EIL}^\omega$

In the following we introduce the system $\mathtt{EIL}^\omega$ [15] which forms in a sense a weak base system containing exactly the tools needed to carry out the functional interpretation. It extends intuitionistic logic in finite types with appropriate combinators[16], a cases operator $\mathcal{D}$ and some very basic arithmetic needed to define characteristic functionals for quantifier-free formulas. We also include C. Spector's quantifier-free rule of extensionality $\mathtt{ER_0}$, see Section 2.3. This allows an as extensional as possible treatment of higher type equality in the context of functional interpretation[17] – see also [34].

We first carry out a full quantitative analysis for the functional interpretation of an extension $\mathtt{EIL}^\omega_+ + \mathtt{AC} + \mathtt{IP}_\forall + \mathtt{MK}$ [18] of $\mathtt{EIL}^\omega$ into the quantifier-free fragment of $\mathtt{EIL}^\omega$. Due to the modularity of functional interpretation this analysis immediately relativizes to further extensions of $\mathtt{EIL}^\omega$ with certain axioms like, e.g., induction. Suppose that we consider an additional (closed) axiom A. Let us add to $\mathtt{EIL}^\omega$ new constants $\underline{c}$ of appropriate types and the axiom[19] $\forall \underline{y} A_{\mathtt{D}}(\underline{c}, \underline{y})$ expressing that $\underline{c}$ satisfies the functional interpretation of A. The quantitative analysis for the functional interpretation of $\mathtt{EIL}^\omega_+ + \mathtt{AC} + \mathtt{IP}_\forall + \mathtt{MK}$ immediately relativizes to this extension. Functional interpretation now provides realizing terms $\underline{t}[\underline{c}]$ built up out of the $\mathtt{EIL}^\omega$-material and $\underline{c}$. The complexity analysis for the extended theory is then completed by determining actual terms $\underline{s}$ which satisfy the functional interpretation $\exists \underline{x} \forall \underline{y} A_{\mathtt{D}}(\underline{x}, \underline{y})$ of A and the complexity of the verifying proof $\vdash \forall \underline{y} A_{\mathtt{D}}(\underline{s}, \underline{y})$.

[15] Acronym for "(weakly extensional) extended intuitionistic logic in all finite types".
[16] These allow the definition of λ-terms, see Definition 2.12.
[17] Most applications of functional interpretation have been based on such an extensional variant. For sentences containing only variables of type 0 or 1 the use of full extensionality is admissible since the elimination-of-extensionality procedure from [39] is applicable.
[18] $\mathtt{AC}$ is the Axiom of Choice, $\mathtt{IP}_\forall$ is Independence of Premises for universal premises. $\mathtt{MK}$ is a variant of Markov's principle, see Section 3.1. For $\mathtt{EIL}^\omega_+$ see Definition 3.10.
[19] Here $\exists \underline{x} \forall \underline{y} A_{\mathtt{D}}(\underline{x}, \underline{y})$ is the functional interpretation of A, see also Section 3.

There are two possible ways of handling λ-abstraction in a system like $\mathtt{EIL}^\omega$. We could treat λ-abstraction either as a primitive concept or as defined by combinators. The treatment via combinators provides a finer complexity analysis and reflects more faithfully the actual functional interpretation of a Hilbert-style axiomatization [20] of intuitionistic logic which we have – following Gödel's original formulation – used for $\mathtt{EIL}^\omega$. The combinators and projectors we use are more flexible than the usual Σ and Π first introduced by Schönfinkel in [46]. Our Σ provide in particular extensions of Schönfinkel's Σ to tuples (see Definition 2.4) and our Π are extensions of Schönfinkel's Π to tuples. This is natural since we use tuples of variables throughout our formulation of functional interpretation. The design of our Σ and Π is made according to the actual constructs required by functional interpretation while keeping the benefits of the usual Σ and Π. The latter allow one to avoid any notion of bound variables in terms and are the most convenient in connection with logical relations [21]. Our Σ and Π are in fact definable in terms of usual Σ and Π, though at the expense of a rather artificial increase in the length of the verifying proof. The upper bound on the size of the extracted terms would nevertheless still hold with such a definition, see Remark 3.30.

2.1 The type structure FT

The set FT of all *finite types* is inductively generated by the rules

(i) $o \in \mathtt{FT}$

(ii) If $\sigma, \tau \in \mathtt{FT}$ then $(\sigma\tau) \in \mathtt{FT}$.

Intuitively type o represents the set of natural numbers and $(\sigma\tau)$ represents the set of functions which map objects of type σ to objects of type τ. There are many alternative notations in the literature for $(\sigma\tau)$, like for example $\tau(\sigma)$, $(\sigma)\tau$, $(\sigma \to \tau)$. We make the convention that concatenation of types is right associative and consequently omit unnecessary parenthesis, writing $\delta\sigma\tau$ instead of $(\delta(\sigma\tau))$. It can immediately be verified by induction over FT that each $\sigma \in \mathtt{FT}$ has the form $\sigma_1 \ldots \sigma_n o$ with $n \geq 0$. We abbreviate by:

[20] In a natural deduction context, it might be more natural to treat λ-abstraction as a primitive concept. Natural deduction formulations of functional interpretation are provided by Diller-Nahm [10] (see also [45,53]) and Joergensen [28]. In the former all definitions by cases for the realizing terms of contractions are postponed to the end by collecting all candidates and making a single final global choice. In the latter choices are local and one has to apply a so-called "contraction lemma" for each of them, i.e., whenever more than one copy of an assumption gets cancelled. In any case, the analysis carried out in the present paper can immediately be adapted to a system with λ-abstraction included as primitive construct, see Remark 3.30.

[21] One example of a logical relation is Howard's majorizability which plays a key role in most applications of functional interpretation [3,31,35].

- $\underline{\sigma}$ the ordered tuple of types $\sigma_1, \ldots, \sigma_n$
- $\underline{\sigma}\tau$ the type $\sigma_1 \ldots \sigma_n \tau$.

Definition 2.1 For a type we define:

- the *arity* by $ar(o) :\equiv 0$ and $ar(\sigma\tau) :\equiv ar(\tau) + 1$;
- the *degree* by $dg(o) :\equiv 0$ and $dg(\sigma\tau) :\equiv max\{dg(\sigma) + 1\,,\, dg(\tau)\}$

and for a tuple of types we define

- the *arity* by $ar(\underline{\sigma}) :\equiv max\{ar(\sigma_1)\,,\, \ldots\,,\, ar(\sigma_n)\}$;
- the *degree* by $dg(\underline{\sigma}) :\equiv max\{dg(\sigma_1)\,,\, \ldots\,,\, dg(\sigma_n)\}$.

Then $dg(\underline{\sigma}\tau) = max\{dg(\underline{\sigma}) + 1\,,\, dg(\tau)\}$ and $ar(\underline{\sigma}\tau) = ar(\tau) + |\underline{\sigma}|$.

2.2 Intuitionistic Equality Logic over FT *(*$\texttt{IEL}^\omega$*)*

Our formalization of $\texttt{IEL}^\omega$ below is a slight modification of the axiomatic calculus for multisorted intuitionistic predicate logic used by Gödel in his original paper on functional interpretation [22]. The only differences are:

(1) The *syllogism* and *expansion* are formulated as axioms instead of rules. Gödel's formulation with rules was designed to ease the formulation of the soundness proof for the functional interpretation. Nevertheless for the quantitative analysis it is more convenient to use the axiom versions of

(a) the expansion rule $\dfrac{A \to B}{C \vee A \to C \vee B}$, since the formula C may introduce realizing terms of arbitrary complexities; also the formula complexity of the conclusion is higher than that of the premise;

(b) the syllogism rule $\dfrac{A \to B\,,\, B \to C}{A \to C}$, which would force us to consider the sum of quantitative measures of both premises when computing upper bounds for quantitative measures of the conclusion. We can immediately notice that the mere Modus Ponens $\dfrac{A\,,\, A \to B}{B}$ avoids such a situation, since the formula complexity of the premise $A \to B$ upper bounds that of the conclusion B .

(2) The quantifier rules and axioms are formulated with tuples of variables since we use tuples throughout the functional interpretation.

The language of $\texttt{IEL}^\omega[\mathcal{C}]$ contains, aside from the constants $\mathcal{C}$, the following:

- denumerably many *variables* which we denote by letters x, y, z, u, v, w, possibly capitalized or adorned with subscripts; $\underline{x} :\equiv x_1, \ldots, x_n$ denotes a tuple of variables; in the same context we use x as metavariable for an individual element of $\underline{x}$; each of the variables is associated a unique sort (mostly called type) which is an element of FT, such that there exist denumerably many variables for each sort; we possibly indicate the type of a variable by carrying it as a superscript, like x^σ and then we denote $\underline{x}^{\underline{\sigma}} :\equiv x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$;

- a binary predicate constant $=_o$ for equality between objects of type o ;
- logical constants $\perp$, $\wedge$, $\vee$, $\rightarrow$, $\forall x$ and $\exists x$ (for each variable x) .

Each of the constants in $\mathcal{C}$ is sorted as well, with the type possibly indicated as superscript. We often do not indicate $\mathcal{C}$ and write $\mathtt{IEL}^\omega$ when the set of constants is either clear from the context or not relevant. We use l as metavariable for both variables and constants.

The terms of $\mathtt{IEL}^\omega$ are sorted, with their types possibly indicated in superscripts and are inductively generated from variables and constants according to the rule that if $t^{\sigma\tau}$ and s^σ are terms then $(ts)^\tau$ is a term. Terms are denoted by letters s, t, r, possibly adorned with subscripts; tuples of terms are denoted like $\underline{t} :\equiv t_1, \ldots, t_n$; in the same context we use t as metavariable for an individual element of $\underline{t}$. We denote by $\mathcal{V}(t)$ the set of variables occurring in t and write $t[\underline{x}]$ to indicate that $\{\underline{x}\} \subseteq \mathcal{V}(t)$. If $\mathcal{V}(t) = \emptyset$ we say that t is a *closed* term. We make the convention that concatenation of terms is left associative and consequently omit unnecessary parenthesis, writing rst instead of $((rs)t)$. When writing down an expression it is always assumed that the terms are well-formed, i.e. the types are fitting. For t^σ we denote by $typ(t) :\equiv \sigma$ and by

- $ar(t) :\equiv ar(\sigma)$ the *arity* of t ;
- $dg(t) :\equiv dg(\sigma)$ the *degree* of t .

For a term we define

- the *depth* by $d(l) :\equiv 0$ and $d(ts) :\equiv max\{d(t)\,,\, d(s)\} + 1$;
- the *size* by $S(l) :\equiv 1$ and $S(ts) :\equiv S(t) + S(s)$.

The *subterm* relation is defined as the reflexive transitive closure of $\{(s, ts), (t, ts)\}$. We denote by $s \leq t$ the fact that s is a subterm of t . It is obvious that $\leq$ is a partial order relation. Let $mdg(t) :\equiv max\{dg(s) \mid s \leq t\}$ and $mar(t) :\equiv max\{ar(s) \mid s \leq t\}$. We notice that :

- $dg(t) \geq dg(ts)$, hence $mdg(r) = max_{l \leq r}\, dg(l)$
- $ar(t) \geq ar(ts)$, hence $mar(r) = max_{l \leq r}\, ar(l)$

We will abbreviate by $t(\underline{s}) :\equiv t\, s_1 \ldots s_m$ and $\underline{t}(\underline{s}) :\equiv t_1(\underline{s}), \ldots, t_n(\underline{s})$.

The formulas of $\mathtt{IEL}^\omega$ are inductively generated from *prime formulas* $s^o =_o t^o$ and $\perp$ according to the rule that if A and B are formulas then $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(\forall x A)$ and $(\exists x A)$ are formulas. Equivalence and negation of formulas are defined as $A \leftrightarrow B :\equiv ((A \rightarrow B) \wedge (B \rightarrow A))$ and respectively $\neg A :\equiv (A \rightarrow \perp)$. The expressions $\forall \underline{x}$, $\exists \underline{x}$ abbreviate $\forall x_1 \ldots \forall x_n$ and $\exists x_1 \ldots \exists x_n$

respectively. Equality between the terms s and t of type $\sigma = \sigma_1 \ldots \sigma_n o$ (with $1 \leq n$) is just an abbreviation for $\forall x_1^{\sigma 1} \ldots x_n^{\sigma_n}(s\, x_1 \ldots x_n =_o t\, x_1 \ldots x_n)$, where the variables $x_1, \ldots, x_n$ do not occur in s or t. Non–equality (or difference) between terms s and t is defined by $s \neq t :\equiv \neg(s = t)$. We abbreviate by $\underline{s} = \underline{t} \;:\equiv\; (s_1 = t_1), \ldots, (s_n = t_n)$ – hence a tuple of formulas.

We denote formulas by letters A, B, C, possibly adorned with subscripts or superscripts. In order to avoid unnecessary parenthesis we make the convention that $\forall x$, $\exists x$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ is the decreasing order of precedence and that $\rightarrow$ is right associative. We call a formula *quantifier-free* if it does not contain $\forall, \exists, \vee$. The subscript 0 always indicates a quantifier-free formula, such as A_0, B_0, C_0. We denote by $\mathcal{V}_\mathrm{f}(A)$, $\mathcal{V}_\mathrm{b}(A)$, $\mathcal{V}(A)$ the set of free, bound, respectively all variables occurring in A and write $A(\underline{x})$ to indicate that $\{\underline{x}\} \subseteq \mathcal{V}_\mathrm{f}(A)$. We denote by $\mathcal{C}(A)$ the set of constants occurring in A and by $vdg(A) :\equiv dg(\mathcal{V}(A))$, $var(A) :\equiv ar(\mathcal{V}(A))$, $cdg(A) :\equiv dg(\mathcal{C}(A))$ and $car(A) :\equiv ar(\mathcal{C}(A))$. We denote by $d_\mathcal{S}(\cdot)$ the $\mathcal{S}$*-depth* of a formula which is defined for $\mathcal{S} \subseteq \{\forall, \exists, \wedge, \vee, \rightarrow\}$ by

- $d_\mathcal{S}(s =_o t) :\equiv d_\mathcal{S}(\bot) :\equiv k_0$ (see Section 1.2 for the definition of k_0)
- For $Q \in \{\forall, \exists\}$, $d_\mathcal{S}(Qx\, A) :\equiv \begin{cases} d_\mathcal{S}(A) + 1\,, & \text{if } Q \in \mathcal{S} \\ d_\mathcal{S}(A) & \text{, if } Q \notin \mathcal{S} \end{cases}$
- For $\Box \in \{\wedge, \vee, \rightarrow\}$, $d_\mathcal{S}(A \Box B) :\equiv \begin{cases} max\{d_\mathcal{S}(A)\,,\, d_\mathcal{S}(B)\} + 1\,, & \text{if } \Box \in \mathcal{S} \\ max\{d_\mathcal{S}(A)\,,\, d_\mathcal{S}(B)\} & \text{, if } \Box \notin \mathcal{S} \end{cases}$

For a formula A we define the following:

- the *logical constants depth* by $ld(A) :\equiv d_{\forall,\exists,\wedge,\vee,\rightarrow}(A)$;
- the *whole depth* by $wd(A) :\equiv d'_{\forall,\exists,\wedge,\vee,\rightarrow}(A)$; here d' differs from d just in $d'_\mathcal{S}(s =_o t) :\equiv k_0 + max\{d(s)\,,\, d(t)\}$;
- the *implication depth* $id(A) :\equiv d^\mathrm{o}_\rightarrow(A)$ and the *forall depth* $fd(A) :\equiv d^\mathrm{o}_\forall(A)$; here d^o differs from d just in $d^\mathrm{o}_\mathcal{S}(A_0) :\equiv k_0$;
- the *forall/implication depth* by $fid(A) :\equiv max\{fd(A)\,,\, id(A)\}$;
- the *quantifier size*, denoted $qs(A)$, is the number of quantifiers (including $\vee$) occurring in A, when A is a closed formula and the quantifier size of its universal closure in the general case;
- the *logical constants size*, denoted $ls(A)$, is obtained by adding to $qs(A)$ the number of $\wedge, \rightarrow, \bot, =$ occurring in A;
- the *whole size*, denoted $ws(A)$, is obtained by adding to $ls(A)$ the number of all occurrences of variables and constants in A.

We present below the axioms and rules of $\mathtt{IEL}^\omega$:

Logical axioms

$\mathtt{CT}\vee$:	$A \vee A \to A$	(contraction)
$\mathtt{CT}\wedge$:	$A \to A \wedge A$	
$\mathtt{WK}\vee$:	$A \to A \vee B$	(weakening)
$\mathtt{WK}\wedge$:	$A \wedge B \to A$	
$\mathtt{PM}\vee$:	$A \vee B \to B \vee A$	(permutation)
$\mathtt{PM}\wedge$:	$A \wedge B \to B \wedge A$	
$\mathtt{SYL}$:	$(A \to B) \wedge (B \to C) \to (A \to C)$	(syllogism)
$\mathtt{EPN}$:	$(A \to B) \to (C \vee A \to C \vee B)$	(expansion)
$\mathtt{EFQ}$:	$\bot \to A$	(ex falso quodlibet)
$\mathtt{QA}\forall$:	$\forall \underline{z} A(\underline{z}) \to A(\underline{s})$	(quantifier axioms)
$\mathtt{QA}\exists$:	$A(\underline{s}) \to \exists \underline{z} A(\underline{z})$	

We denote by $\mathtt{QA} :\equiv \mathtt{QA}\forall + \mathtt{QA}\exists$. At $\mathtt{QA}$, s is free for z in A and the substitution is simultaneous.

For instances $B(\underline{s})$ of $\mathtt{QA}$ which involve the constants $\underline{s}$, we define the

- *term depth* of $B(\underline{s})$ by $td(B) :\equiv d(\underline{s})$;
- *term size* of $B(\underline{s})$ by $ts(B) :\equiv \Sigma_{s \in \underline{s}} S(s)$.

Logical rules

$\mathtt{MP}$:	$A\,,\, A \to B \vdash B$	(modus ponens)
$\mathtt{EXP}$:	$A \wedge B \to C \vdash A \to (B \to C)$	(exportation)
$\mathtt{IMP}$:	$A \to (B \to C) \vdash A \wedge B \to C$	(importation)
$\mathtt{QR}\forall$:	$B \to A \vdash B \to \forall \underline{z} A$	(quantifier rules)
$\mathtt{QR}\exists$:	$A \to B \vdash \exists \underline{z} A \to B$	

We denote by $\mathtt{QR} :\equiv \mathtt{QR}\forall + \mathtt{QR}\exists$. At $\mathtt{QR}$, z is not free in B .

Equality axioms

$\mathtt{REF}$:	$x =_o x$	(reflexivity)
$\mathtt{SYM}$:	$x =_o y \to y =_o x$	(symmetry)
$\mathtt{TRZ}$:	$x =_o y \wedge y =_o z \to x =_o z$	(transitivity)

Recall that $\vdash_n$, with $n \in \mathbb{N}$, denotes a deduction of length at most n.

Remark 2.2 There exists $k \in \mathbb{N}$ constant such that for all σ :

Higher–order equality

$\mathtt{REF}[\sigma]$:	$\mathtt{IEL}^\omega \vdash_k x =_\sigma x$	(reflexivity)
$\mathtt{SYM}[\sigma]$:	$\mathtt{IEL}^\omega \vdash_k x =_\sigma y \to y =_\sigma x$	(symmetry)
$\mathtt{TRZ}[\sigma]$:	$\mathtt{IEL}^\omega \vdash_k x =_\sigma y \wedge y =_\sigma z \to x =_\sigma z$	(transitivity)

Given a set of rules (axioms are comprised as rules with empty premise) $\mathtt{Rl}$ whose formulas contain the constants $\mathcal{C}$, we denote by $\mathtt{IEL}^\omega[\mathtt{Rl}]$ the system $\mathtt{IEL}^\omega[\mathcal{C}]$ extended with the rules in $\mathtt{Rl}$. We sometimes abbreviate $\mathtt{IEL}^\omega[\mathtt{Rl}]$ with a different denotation (like $\mathtt{EIL}^\omega$ below) and then $(\mathtt{IEL}^\omega[\mathtt{Rl}])[\mathtt{Rl}'] :\equiv \mathtt{IEL}^\omega[\mathtt{Rl} \cup \mathtt{Rl}']$.

2.3 $\mathtt{EIL}^\omega$ *– Extended Intuitionistic Equality Logic over* $\mathtt{FT}$

Multisorted weakly extensional extended intuitionistic equality logic over $\mathtt{FT}$, which we denote by $\mathtt{EIL}^\omega$, is obtained by extending $\mathtt{IEL}^\omega$ with exactly the elements which are strictly necessary to carry out functional interpretation even for $\mathtt{IEL}^\omega$. The language of $\mathtt{EIL}^\omega$ contains the following constants:

- the *zero* constant $0 \equiv \mathtt{0}_o$ of type o and for each type $\rho \equiv \underline{\sigma}o$ the higher–order *zero* constant $\mathtt{0}_\rho$ which is defined by the axiom

  $\mathtt{Ax0}$: $\quad \mathtt{0}_\rho(\underline{z}^{\underline{\sigma}}) = 0$

  (hence for any type there exists at least one constant)
- the *successor* constant $\mathtt{S}$ of type oo which is defined by the axioms

  $\mathtt{AxS}$: $\quad \mathtt{S}\,x \neq 0 \quad$ and $\quad \mathtt{S}\,x = \mathtt{S}\,y \to x = y$
- the *boolean* constants $\nu, \mathtt{I}, \mathtt{E}$ all of type ooo which are defined by the axioms

  $\mathtt{Ax}\nu$: $\quad (x = 0 \wedge y = 0) \leftrightarrow \nu\,x\,y = 0$

  $\mathtt{AxI}$: $\quad (x = 0 \to y = 0) \leftrightarrow \mathtt{I}\,x\,y = 0$

  $\mathtt{AxE}$: $\quad x = y \leftrightarrow \mathtt{E}\,x\,y = 0$
- for each $n, i \in \mathbb{N}$ with $i \leq n$ and types $\underline{\sigma} \equiv \sigma_1, \ldots, \sigma_n$, the *decision* constant $\mathcal{D}_i^{\underline{\sigma}}$ of type $o\underline{\sigma}\,\underline{\sigma}\sigma_i$ which is defined by the axioms (below $|\underline{z}| = |\underline{z}'|$)

  $\mathtt{Ax}\mathcal{D}$: $\quad x = 0 \to \mathcal{D}_i^{\underline{\sigma}}(x, \underline{z}, \underline{z}') = z_i \quad$ and $\quad x \neq 0 \to \mathcal{D}_i^{\underline{\sigma}}(x, \underline{z}, \underline{z}') = z'_i$

- for each choice of the following
  - $n, m \in \mathbb{N}$ and $\underline{n} :\equiv n_0, n_1, \ldots, n_m \in \mathbb{N}$ and $\overline{n} :\equiv n^1, \ldots, n^m \in \mathbb{N}$ such that $n_0, n_1, \ldots, n_m \leq n$ and $n^1, \ldots, n^m \leq n$
  - permutations $\underline{p} :\equiv p_0, p_1, \ldots, p_m$ and $\overline{p} :\equiv p^1, \ldots, p^m$ of $\{1, \ldots, n\}$
  - types τ, $\underline{\sigma} \equiv \sigma_1, \ldots, \sigma_n$ and $\underline{\delta} \equiv \delta_1, \ldots, \delta_m$

  the *combinator* constant $\varSigma^{\underline{\sigma},\underline{\delta},\tau,m}_{\underline{p},\overline{p},\underline{n},\overline{n}}$ which is defined by the following axiom

  $\mathtt{Ax}\varSigma$: $\quad \varSigma^{\underline{\sigma},\underline{\delta},\tau,m}_{\underline{p},\overline{p},\underline{n},\overline{n}}(x, \underline{y}, \underline{z}) = x(\underline{z_0}, y_1(\underline{z}^1), \underline{z_1}, \ldots, y_m(\underline{z}^m), \underline{z_m})$

  The type of $\varSigma^{\underline{\sigma},\underline{\delta},\tau,m}_{\underline{p},\overline{p},\underline{n},\overline{n}}$ is $(\underline{\sigma_0}\, \delta_1\, \underline{\sigma_1}\, \ldots\, \delta_m\, \underline{\sigma_m}\, \tau)\, (\underline{\sigma}^1\, \delta_1) \ldots (\underline{\sigma}^m\, \delta_m)\, \underline{\sigma}\, \tau$, where we abbreviated by $\{\underline{\sigma_j} :\equiv \sigma_{(p_j)_1}, \ldots, \sigma_{(p_j)_{n_j}}\}_{j=0}^m$ and $\{\underline{\sigma}^j :\equiv \sigma_{(p^j)_1}, \ldots, \sigma_{(p^j)_{n^j}}\}_{j=1}^m$.
- for each $n \in \mathbb{N}$, permutation p of $\{1, \ldots, n\}$ and types $\tau, \underline{\sigma} \equiv \sigma_1, \ldots, \sigma_n$, the *permutation* constant $P^{\underline{\sigma},\tau}_{n,p}$ of type $(\underline{\sigma}\tau)\underline{\sigma}^p\tau$ which is defined by the axiom

  $\mathtt{Ax}P$: $\quad P^{\underline{\sigma},\tau}_{n,p}(x, \underline{z}^p) = x(\underline{z})$

  Recall from Section 1.2 that $\underline{\sigma}^p$ and $\underline{z}^p$ represent the p–permuted $\underline{\sigma}$ and $\underline{z}$.
- for each $n, i \in \mathbb{N}$, $i \leq n$ and types $\underline{\sigma} \equiv \sigma_1, \ldots, \sigma_n$, the *projector* constant $\varPi^{\underline{\sigma}}_i$ of type $\underline{\sigma}\sigma_i$ which is defined by the axiom

  $\mathtt{Ax}\varPi$: $\quad \varPi^{\underline{\sigma}}_i(\underline{z}) = z_i$

For simplicity we abbreviate by $1 :\equiv \mathtt{S}0$. The system $\mathtt{EIL}^\omega$ is finally obtained by adding the quantifier-free tertium non datur axiom

$\mathtt{TND_0}$: $\quad x = 0 \vee \neg(x = 0)$

and the quantifier-free extensionality rule

$$\mathtt{ER_0} : \quad \frac{A_0 \to s_1 = t_1\,, \;\ldots\;, A_0 \to s_n = t_n}{A_0 \to B_0(\underline{s}) \to B_0(\underline{t})}\;.$$

The formal proofs in the sequel will be in $\mathtt{EIL}^\omega$ if not otherwise indicated.

Remark 2.3 The constants P and $\varPi$ are definable in terms of $\varSigma$ and also $\mathtt{0}_{\underline{\sigma}o} = \varPi^{o,\underline{\sigma}}_1 0$. We nevertheless chose to define them separately since they play a particular rôle.

Definition 2.4 As particular cases of $\varSigma$ we distinguish the *tuple-Schönfinkel combinators* $\varSigma^{\underline{\sigma},(\delta),\tau,1}_{(\mathtt{1}_\mathtt{n},\mathtt{1}_\mathtt{n}),(\mathtt{1}_\mathtt{n}),(n,0),(n)}$ with defining axioms of shape

$$\varSigma^{\underline{\sigma},(\delta),\tau,1}_{(\mathtt{1}_\mathtt{n},\mathtt{1}_\mathtt{n}),(\mathtt{1}_\mathtt{n}),(n,0),(n)}(x, y, \underline{z}) = x(\underline{z}, y(\underline{z}))\;.$$

These are generalizations of the usual[22] Schönfinkel combinators $\varSigma$ to tuples and will be used in the λ-abstraction Definition 2.12. The usual *Schönfinkel combinators* $\varSigma$ are in fact particular cases of our $\varSigma$ of shape $\varSigma^{\underline{\sigma},(\delta),\tau,1}_{(\mathtt{1}_1,\mathtt{1}_1),(\mathtt{1}_1),(1,0),(1)}$ with defining axioms $\varSigma^{\underline{\sigma},(\delta),\tau,1}_{(\mathtt{1}_1,\mathtt{1}_1),(\mathtt{1}_1),(1,0),(1)}(x, y, z) = x(z, y(z))$. Also the usual *Schönfinkel projectors* $\varPi$ are particular cases of our $\varPi$ of shape $\varPi^{(\sigma_1,\sigma_2)}_1$ with defining axioms $\varPi^{(\sigma_1,\sigma_2)}_1(z_1, z_2) = z_1$.

[22] For the original definition of Schönfinkel's $\varSigma$ and $\varPi$ see [46]. See also the last paragraph before Section 2.1.

Remark 2.5 The quantifier-free tertium-non-datur $\mathtt{TND}_0$ becomes derivable in the presence of induction for propositional formulas. Moreover, in the presence of a modest amount of arithmetic, the constants $\mathcal{D}$, ν, $\mathtt{I}$ and $\mathtt{E}$ are definable and their axioms derivable. Therefore these axioms are in fact redundant in any concrete application of functional interpretations, e.g., to $\mathtt{HA}^\omega$ and fragments thereof. Examples of the latter are systems of bounded arithmetic like $\mathtt{IPV}^\omega$ of [8] and the poly-time arithmetic $\mathtt{LHA}$ of [47] [23] .

Remark 2.6 The extensionality axiom

$$\mathtt{EA}[\underline{\sigma}] : \qquad \underline{x}^{\underline{\sigma}} = \underline{y}^{\underline{\sigma}} \;\rightarrow\; f^{\underline{\sigma}o}\, \underline{x} =_o f^{\underline{\sigma}o}\, \underline{y} \qquad\qquad (\text{ let } \mathtt{EA} :\equiv \cup_{\underline{\sigma}} \mathtt{EA}[\underline{\sigma}])$$

is derivable in $\mathtt{EIL}^\omega$ for $\underline{\sigma} \equiv o, \ldots, o$, particularly using the rule $\mathtt{ER}_0$. Therefore $\mathtt{EIL}^\omega$ contains *all* equality axioms for type o. This no longer holds in general when $\underline{\sigma}$ contains higher types (follows from Section 3.5.10 of [55] and [27]). On the other hand, $\mathtt{ER}_0$ is derivable from $\mathtt{EA}$ in $\mathtt{EIL}^\omega \setminus \mathtt{ER}_0$, hence the rule is strictly weaker than the axiom, but only at higher types.

Remark 2.7 These hold in $\mathtt{EIL}^\omega$: $\vdash \bot \leftrightarrow 1 = 0$ and $\vdash x \neq 0 \leftrightarrow \mathtt{I}x1 = 0$.

Remark 2.8 There exists $k \in \mathbb{N}$ constant such that for all $\underline{s}$, $\underline{t}$, r, r_1, r_2, B_0, the following hold :

$$\underline{s} = \underline{t} \vdash_k \; B_0(\underline{s}) \rightarrow B_0(\underline{t}) \tag{1}$$

$$\underline{s} = \underline{t} \vdash_k \; r[\underline{s}] = r[\underline{t}]$$

$$r_1 = r_2, \underline{s} = \underline{t} \vdash_k \; r_1(\underline{s}) = r_2(\underline{t}). \tag{2}$$

Proposition 2.9 The following equalities hold:

$$\begin{aligned} dg(\Sigma^{\underline{\sigma},\underline{\delta},\tau,m}_{\underline{p},\overline{p},\underline{n},\overline{n}}) &= max\{dg(\underline{\sigma},\underline{\delta}) + 2\,,\, dg(\tau) + 1\} & dg(\Pi^{\underline{\sigma}}_i) &= dg(\underline{\sigma}) + 1 \\ dg(P^{\underline{\sigma},\tau}_{n,p}) &= max\{dg(\underline{\sigma}) + 2\,,\, dg(\tau) + 1\} & dg(\mathcal{D}^{\underline{\sigma}}_i) &= dg(\underline{\sigma}) + 1 \\ ar(\Sigma^{\underline{\sigma},\underline{\delta},\tau,m}_{\underline{p},\overline{p},\underline{n},\overline{n}}) &= ar(\tau) + |\underline{\sigma}| + |\underline{\delta}| + 1 & ar(\Pi^{\underline{\sigma}}_i) &= ar(\sigma_i) + |\underline{\sigma}| \\ ar(\mathcal{D}^{\underline{\sigma}}_i) &= ar(\sigma_i) + 2|\underline{\sigma}| + 1 & ar(P^{\underline{\sigma},\tau}_{n,p}) &= ar(\tau) + |\underline{\sigma}| + 1 \end{aligned}$$

In the proposition below we show how and at which cost in proof depth the quantifier-free formulas can be viewed as prime formulas.

Proposition 2.10 (Association of terms to quantifier-free formulas) There exists $k \in \mathbb{N}$ constant and an association of terms to quantifier-free formulas $A_0 \mapsto t_{A_0}$ such that for all A_0 ,

$$\vdash_{k \cdot ld(A_0)} \; A_0(\underline{a}) \leftrightarrow t_{A_0}[\underline{a}] = 0 \; . \tag{3}$$

Proof: The proof is by induction on the structure of A_0, making use of the boolean constants axioms. For prime formulas just take $t_{t_1 = t_2} :\equiv \mathtt{E}\, t_1 t_2$ and

[23] Even though $\mathtt{LHA}$ was designed in a modified realizability context, the outline of similar systems corresponding to functional interpretations is quite straightforward.

$t_\perp :\equiv 1$, then recursively define $t_{B_0 \wedge C_0} :\equiv \nu\, t_{B_0} t_{C_0}$ and $t_{B_0 \to C_0} :\equiv \mathtt{I}\, t_{B_0} t_{C_0}$. $\square$

Corollary 2.11 (TND and Stability for quantifier-free formulas) There exists $k \in \mathbb{N}$ constant such that for all quantifier–free A_0 ,

$$\vdash_{k \cdot ld(A_0)} A_0(\underline{a}) \vee \neg A_0(\underline{a}) \tag{4}$$

$$\vdash_{k \cdot ld(A_0)} \neg\neg A_0(\underline{a}) \to A_0(\underline{a}) \; . \tag{5}$$

Proof: The principle $\mathtt{STAB}_0 :\ \neg\neg x = 0 \to x = 0$ follows immediately with constant-depth proof from $\mathtt{TND}_0$. Both (4) and (5) follow immediately from $\mathtt{TND}_0$ and $\mathtt{STAB}_0$ respectively by (3) and (1) . $\square$

Definition 2.12 (λ-abstraction) To every term t^τ one associates a term $(\lambda \underline{x}^{\underline{\sigma}}.\, t)^{\underline{\sigma}\tau}$, with $\mathcal{V}(\lambda \underline{x}.\, t) = \mathcal{V}(t) - \{\underline{x}\}$, recursively defined as follows:

$$\begin{aligned} \lambda \underline{x}.\, x_i &:\equiv \Pi_i^{\underline{\sigma}} \\ \lambda \underline{x}.\, t &:\equiv \Pi_1^{(\tau,\underline{\sigma})} t \, , \ \text{ if } \{\underline{x}\} \cap \mathcal{V}(t) = \emptyset \\ \lambda \underline{x}.\, (t^{\delta\tau} s^{\delta}) &:\equiv \Sigma^{\underline{\sigma},(\delta),\tau,1}_{(\mathtt{1_n},\mathtt{1_n}),(\mathtt{1_n}),(n,0),(n)} (\lambda \underline{x}.\, t)(\lambda \underline{x}.\, s) \, , \ \text{ if } \{\underline{x}\} \cap \mathcal{V}(ts) \neq \emptyset \end{aligned} \tag{6}$$

Proposition 2.13 (β-reduction) There exists $k \in \mathbb{N}$ constant such that for all t and $\underline{r}$ the following holds: $\vdash_{k \cdot d(t)} (\, \lambda \underline{x}^{\underline{\sigma}}.\, t[\underline{x}] \,)\, \underline{r}^{\underline{\sigma}} =_\tau t[\underline{r}]$.

Proof: By straightforward induction on $d(t)$, using (2) when the induction step falls under (6) . $\square$

Proposition 2.14 The following inequalities hold:

$$\begin{aligned} & d(\lambda \underline{x}.\, t) \leq 2 \cdot d(t) && mdg(\lambda \underline{x}.\, t) \leq max\{dg(\underline{x}) + 1 \, , \, mdg(t)\} + 1 \\ & S(\lambda \underline{x}.\, t) \leq 3 \cdot S(t) && mar(\lambda \underline{x}.\, t) \leq max\{mar(t) + 1 \, , \, ar(\underline{x})\} + |\underline{x}| \end{aligned}$$

Proof: By structural induction on t , following Definition 2.12 . $\square$

Remark 2.15 In order to increase readability we will omit the adornments of Σ, P and Π from now on. We consider that this side information can be figured out from the context in a straightforward way. On the other hand its display would only complicate the exposition.

3 A quantitative analysis of functional interpretation

Gödel's functional (*Dialectica*) interpretation/translation was first introduced in [22] and is also presented in [39](4) and [55](3.5.1). It is a translation of proofs which includes a translation of formulas. Hence a given formula $A(\underline{a})$, with $\underline{a}$ all free variables of A, is interpreted to the associated formula $A^{\mathtt{D}} \equiv \exists \underline{x}\, \forall \underline{y}\, A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a})$ with $A_{\mathtt{D}}$ quantifier-free and $\underline{x}, \underline{y}$ tuples of variables of finite type such that $\underline{x}, \underline{y}, \underline{a}$ are all free variables of $A_{\mathtt{D}}$. We often omit to display the tuples of free variables wherever this creates no ambiguity. We will denote by $B(\underline{a'})^{\mathtt{D}} \equiv \exists \underline{u} \forall \underline{v}\, B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{a'})$ and $C(\underline{a''})^{\mathtt{D}} \equiv \exists \underline{g} \forall \underline{h}\, C_{\mathtt{D}}(\underline{g}; \underline{h}; \underline{a''})$. The Dialectica interpretation of formulas is then given by the following list of rules:

Definition 3.1 (Gödel's functional interpretation of formulas)

$$\begin{aligned}
A^{\mathtt{D}} &:\equiv (A_{\mathtt{D}} :\equiv A) \text{ for prime formulas } A \\
(A \wedge B)^{\mathtt{D}} &:\equiv \exists \underline{x}, \underline{u} \forall \underline{y}, \underline{v}\, [(A \wedge B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}) \wedge B_{\mathtt{D}}(\underline{u}; \underline{v})] \\
(\exists z A(\underline{a}, z))^{\mathtt{D}} &:\equiv \exists z, \underline{x} \forall \underline{y}\, [(\exists z A(\underline{a}, z))_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}, z)] \\
(\forall z A(\underline{a}, z))^{\mathtt{D}} &:\equiv \exists \underline{X} \forall z, \underline{y}\, (\forall z A(\underline{a}, z))_{\mathtt{D}} \qquad (7) \\
&\qquad (\forall z A(\underline{a}, z))_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{X}(z); \underline{y}; \underline{a}, z) \\
(A \to B)^{\mathtt{D}} &:\equiv \exists \underline{Y}, \underline{U} \forall \underline{x}, \underline{v}\, (A \to B)_{\mathtt{D}} \qquad (8) \\
&\qquad (A \to B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v}) \\
(A \vee B)^{\mathtt{D}} &:\equiv \exists z^o, \underline{x}, \underline{u} \forall \underline{y}, \underline{v}\, (A \vee B)_{\mathtt{D}} \\
&\qquad (A \vee B)_{\mathtt{D}} :\equiv (z = 0 \to A_{\mathtt{D}}(\underline{x}; \underline{y})) \wedge (Iz1 = 0 \to B_{\mathtt{D}}(\underline{u}; \underline{v}))
\end{aligned}$$

Remark 3.2 For quantifier–free formulas A, $A^{\mathtt{D}} = A_{\mathtt{D}} = A$. The types and lengths of $\underline{x}$ and $\underline{y}$ depend only on the logical structure of A. Notice that $\mathcal{V}_{\mathrm{f}}(A^{\mathtt{D}}) = \mathcal{V}_{\mathrm{f}}(A)$ and $\mathcal{V}_{\mathrm{b}}(A^{\mathtt{D}}) = \{\underline{x}, \underline{y}\}$. In the subsequent presentation, unless otherwise specified, $\underline{x}$ and $\underline{y}$ will refer to the $\underline{x}$ and $\underline{y}$ from $A^{\mathtt{D}}$. Similarly $\underline{u}$, $\underline{g}$ and $\underline{v}$, $\underline{h}$ are bound by default to $B^{\mathtt{D}}$ and $C^{\mathtt{D}}$ respectively.

Proposition 3.3 It can be easily proved by induction on the structure of the formula A (recall from Section 2.3 that qs also counts the free variables) that

$$qs(A^{\mathtt{D}}) = |\underline{x}, \underline{y}, \underline{a}| = qs(A) \qquad (9)$$

Lemma 3.4 The following hold (for k_0 see Section 1.2 and Footnote 14):

$$\begin{aligned}
dg(\mathcal{V}_{\mathrm{b}}(C^{\mathtt{D}})) &\leq vdg(C) + id(C) - k_0 + 1 \qquad (10) \\
ar(\mathcal{V}_{\mathrm{b}}(C^{\mathtt{D}})) &\leq var(C) + qs(C) \cdot (id(C) - k_0 + 1) \qquad (11)
\end{aligned}$$

Proof: The proof is by recursion on the structure of the formula C, following the Definition 3.1. We simply notice that

- $dg(\mathcal{V}_{\mathrm{b}}(\cdot^{D}))$ may increase only at (8), with the quantity 1; (7) forces us to start with $1 + vdg(C)$, since $dg(X) = max\{dg(x), dg(z) + 1\}$;
- $ar(\mathcal{V}_{\mathrm{b}}(\cdot^{D}))$ may increase with the quantity 1 at (7) and with at most $|\underline{x}, \underline{v}| \leq qs(A \to B) \leq qs(C)$ at (8), hence

$$ar(\mathcal{V}_{\mathrm{b}}(C^{\mathtt{D}})) \leq var(C) + fd(C) + qs(C) \cdot (id(C) - k_0) .$$

Definition 3.5 Let $\mathtt{Ax}$ be an arbitrary but fixed[24] set of axioms. For a set of closed terms Tm and a set Fm of formulas in the language of $\mathtt{EIL}^\omega[\mathtt{Ax}]$ we define

- the *prerealization* relation by $PR[\mathrm{Tm},\mathrm{Fm}] :\equiv \{(\underline{t}, A(\underline{a})) \subseteq \mathrm{Tm}^{\leq\omega} \times \mathrm{Fm} \mid |\underline{t}| = |\underline{x}|,\ \{\underline{a}\} = \mathcal{V}_f(A) \text{ and } \mathit{typ}(\underline{t}(\underline{a})) = \mathit{typ}(\underline{x})\}$. For $(\underline{t}, A(\underline{a})) \in PR[\mathrm{Tm},\mathrm{Fm}]$ we abbreviate by $\{\!|\, \underline{t}\,,\, A \,|\!\} :\equiv \forall \underline{y}\, A_{\mathtt{D}}(\underline{t}(\underline{a}); \underline{y}; \underline{a})$.
- the *realization* relation by

$$RR[\mathrm{Tm},\mathrm{Fm}] \ :\equiv\ \{\, (\underline{t}, A) \in PR[\mathrm{Tm},\mathrm{Fm}] \ \mid\ \mathtt{EIL}^\omega[\mathtt{Ax}] \vdash \{\!|\underline{t}, A|\!\}\,\}$$

- the set of *realizing tuple selections* $RTS[\mathrm{Fm},\mathrm{Tm}]$ as the set of those inverses to subsets of $RR[\mathrm{Tm},\mathrm{Fm}]$ which are *functions* from Fm to $\mathrm{Tm}^{\leq\omega}$.

We omit to display Tm when it denotes the set of all the closed terms of $\mathtt{EIL}^\omega[\mathtt{Ax}]$ or Fm when it denotes the set of all formulas in the language of $\mathtt{EIL}^\omega[\mathtt{Ax}]$. The set $\mathtt{Ax}$ will be determined by the context. Whenever $(\underline{t}, A) \in RR$ we denote this fact by $\underline{t}$ Dr A and say that

- $\underline{t}$ is a *realizing tuple* for $A^{\mathtt{D}}$;
- t is a *realizing term* for $A^{\mathtt{D}}$;
- $A^{\mathtt{D}}$ is *realized* by $\underline{t}$ or t .

We call

- *realizer* any realizing tuple or term ;
- *realizer-free* a formula A for which $|\underline{x}| = 0$, where $\underline{x}$ is from $A^{\mathtt{D}}$.

Definition 3.6 We say that a proof $\mathcal{P}$ is *realizer-free-normal* if all realizer-free formulas of $\mathcal{P}$ are located at the leaf level.

Remark 3.7 Let $\mathcal{P}$ be a realizer-free-normal proof. There exists no instance of $\mathtt{ER}_0$ in $\mathcal{P}$ since the conclusion is quantifier-free and consequently realizer-free. Realizer-free formulas of $\mathcal{P}$ may label only leaves of $\mathcal{P}$ which are left premises of $\mathtt{MP}$ instances. Indeed, if the conclusion in any of the rules $\mathtt{QR}$, $\mathtt{EXP}$, $\mathtt{IMP}$ is non-realizer-free then also the premise must be non-realizer-free. For the $\mathtt{MP}$ rule, if the conclusion is non-realizer-free then also the $A \to B$ premise must be non-realizer-free.

Definition 3.8 To any proof $\mathcal{P}$ in some extension of $\mathtt{EIL}^\omega$ we associate a realizer-free-normal proof $\mathcal{P}^{\mathrm{tr}}$ which is obtained from $\mathcal{P}$ by removing its maximal subtrees rooted at vertices labeled with realizer-free formulas, yet keeping these roots (which become assumptions in $\mathcal{P}^{\mathrm{tr}}$). There is a fairly simple algorithm which transforms $\mathcal{P}$ to $\mathcal{P}^{\mathrm{tr}}$ by recursion on proof structure.

Remark 3.9 The proofs we consider in the sequel are realizer-free-normal if not otherwise specified. See also Remark 3.32 .

[24] See also Definition 3.10 and especially Remark 3.11 .

Instances of the following three schemata are formulas whose correspondents under functional interpretation can be realized by very simple terms, basically projectors Π. This makes them the first to be considered for axiom extensions of $\mathtt{EIL}^\omega$ since their inclusion in proofs in the domain of functional interpretation causes no increase in complexity. Moreover the verifying proof is in $\mathtt{EIL}^\omega$ and has a constant bound on its depth. The first two are *logical axioms*, i.e., they are valid in classical logic. The third axiom is non–logical. The schemata are:

(1) A variant of Markov's principle (below A_0 is quantifier–free)

$\mathtt{MK}: \quad \neg\neg\,\exists\underline{x}\,A_0(\underline{x}) \to \exists\underline{x}\,\neg\neg\,A_0(\underline{x})$.

The usual[25] formulation of Markov's principle

$\mathtt{MK}': \quad \neg\neg\exists\underline{x}A_0(\underline{x}) \to \exists\underline{x}A_0(\underline{x})$

can be deduced from $\mathtt{MK}$ with a proof which makes use of (5) and therefore has depth upper bounded by $k \cdot ld(A_0)$ for some $k \in \mathbb{N}$ constant; on the other hand the proof of $\mathtt{MK}$ from $\mathtt{MK}'$ has constant depth.

(2) Independence of Premises for universal premises [below $\underline{y} \notin \mathcal{V}_f(\forall\underline{x}A_0(\underline{x}))$]

$\mathtt{IP}_\forall: \quad [\forall\underline{x}\,A_0(\underline{x}) \;\to\; \exists\underline{y}\,B(\underline{y})\,] \;\to\; \exists\underline{y}\,[\forall\underline{x}\,A_0(\underline{x}) \;\to\; B(\underline{y})\,]$.

(3) The Axiom of Choice

$\mathtt{AC}: \quad \forall\underline{x}\,\exists\underline{y}\,A(\underline{x},\underline{y}) \;\to\; \exists\underline{Y}\,\forall\underline{x}\,A(\underline{x},\underline{Y}(\underline{x}))$.

Another simple axiom extension of $\mathtt{EIL}^\omega$ is with realizer-free formulas since the quantitative analysis does not get affected in any way. There is a particular kind of such axiom extension which we consider in the sequel. Strictly speaking, the terms t_1, t_2 which appear in prime formulas $t_1 = t_2$ of contractions $A \to A \wedge A$ and terms $\underline{s}$ involved in quantifier axioms $A(\underline{s}) \equiv \forall\underline{z}B(\underline{z}) \to B(\underline{s})$ or $A(\underline{s}) \equiv B(\underline{s}) \to \exists\underline{z}B(\underline{z})$ are part of the realizing term (see Section 3.3). However we do not count them in the quantitative analysis, but rather introduce new constants $\widetilde{t_1}, \widetilde{t_2}, \widetilde{s}$ associated to terms t_1, t_2, s together with their defining axioms, such that any of the terms t_1, t_2, s contributes as much as a unit (plus the number of its free variables) of size to the realizing term. This is justified by the fact that we are only interested in the complexity of functional interpretation itself. The terms t_1, t_2, s are not created by functional interpretation – they are merely given as basic input data.

Definition 3.10 Let $\mathbb{A}\mathtt{x}$ be an arbitrary but fixed set of axioms and $\mathbb{T}\mathtt{h}_{\mathtt{rf}}$ an arbitrary but fixed set of realizer-free theorems of $\mathtt{EIL}^\omega[\mathbb{A}\mathtt{x}]$. We define below two extensions $\mathtt{EIL}^\omega_+$ and $\mathtt{EIL}^\omega_\mathtt{v}$ of $\mathtt{EIL}^\omega[\mathbb{A}\mathtt{x}]$. The system $\mathtt{EIL}^\omega_+$ is obtained by

[25] We prefer the variant $\mathtt{MK}$ because the verifying proof of its functional interpretation is much simpler than for $\mathtt{MK}'$. In the latter case the depth of the verifying proof is $k \cdot ld(A_0)$ for some $k \in \mathbb{N}$ constant.

simply adding $\mathbb{Th}_{\mathtt{rf}}$ to the set of axioms of $\mathtt{EIL}^\omega[\mathtt{Ax}]$. Let $\widetilde{\cdot}$ be a map which uniquely associates the $\widetilde{\cdot}$ *constants* $\widetilde{t}$ to terms $t[\underline{a}]$ of $\mathtt{EIL}^\omega[\mathtt{Ax}]$ such that

$$dg(\widetilde{t}) \;=\; max\{dg(\underline{a})+1\,,\, dg(t)\} \qquad \text{and} \qquad ar(\widetilde{t}) \;=\; |\underline{a}| + ar(t) \tag{12}$$

together with the defining axiom

$\mathtt{Ax}\widetilde{t}:\quad t[\underline{a}] \;=\; \widetilde{t}(\underline{a})$.

Let $\mathbb{Tm}$ be an arbitrary but fixed set of $\mathtt{EIL}^\omega[\mathtt{Ax}]$ terms. The system $\mathtt{EIL}^\omega_{\mathtt{v}}$ is obtained by extending $\mathtt{EIL}^\omega_{+}$ with the defining axioms $\mathtt{Ax}\widetilde{t}$ for the newly introduced constants $\widetilde{t}$ associated to terms $t \in \mathbb{Tm}$ by (12) .

Remark 3.11 All *arbitrary but fixed* items in the above definition will be implicitly given by their context if not explicitly described.

3.2 The treatment of $\mathtt{EIL}^\omega$ *rules*

Remark 3.12 Remember that the formal proofs below are by default in $\mathtt{EIL}^\omega$. See Section 1.2 for the definitions of $\mathbb{Vt}$, $\mathbb{Lv}$ and ∂. The definitions of qs, ls and the other quantitative measures of terms or formulas are given in Section 2.3. Recall that qs also counts the free variables of its argument. For the meaning of the relations *PR*, *RR* and *RTS* below see Definition 3.5 .

Lemma 3.13 The following hold for any proof $\mathcal{P}$:

$$qs(\mathbb{Vt}(\mathcal{P})) \;=\; qs(\mathbb{Lv}(\mathcal{P})) \qquad \text{and} \qquad ls(\mathbb{Vt}(\mathcal{P})) \;=\; ls(\mathbb{Lv}(\mathcal{P})) \tag{13}$$

$$\mathcal{V}(\mathbb{Vt}(\mathcal{P})) \;=\; \mathcal{V}(\mathbb{Lv}(\mathcal{P})) \qquad \text{and} \qquad \mathcal{C}(\mathbb{Vt}(\mathcal{P})) \;=\; \mathcal{C}(\mathbb{Lv}(\mathcal{P})) \tag{14}$$

Proof: The following (in)equalities are immediate :

$$qs(A \to \forall \underline{z} B(\underline{z})) = qs(A \to B(\underline{z})) \qquad qs(\exists \underline{z} A(\underline{z}) \to B) = qs(A(\underline{z}) \to B)$$

$$qs(A \wedge B \to C) = qs(A \to B \to C) \qquad\qquad qs(B) \leq qs(A \to B)$$

It follows by structural induction on $\mathcal{P}$ that $qs(A) \leq qs(\mathbb{Lv}(\mathcal{P}))$ for any formula $A \in \mathbb{Vt}(\mathcal{P})$ and then $qs(\mathbb{Vt}(\mathcal{P})) = qs(\mathbb{Lv}(\mathcal{P}))$ is immediate. The argument for ls is identical and (14) has a similar proof, with $\subseteq$ instead of $\leq$. $\Box$

Lemma 3.14 (MP) Let $\mathfrak{A}_{\mathtt{MP}}$ be the algorithm which produces $(\underline{t_4}, B(\underline{a'})) \in PR$ from the input $(\underline{t_1}, A(\underline{a}))$, $(\underline{t_2}, \underline{t_3}, (A \to B)(\underline{\widetilde{a}})) \in PR$, where $\{\underline{a_1}\} = \{\underline{a}\} - \{\underline{a'}\}$, $\{\underline{\widetilde{a}}\} = \{\underline{a}\} \cup \{\underline{a'}\}$ and t_4 is obtained from $t'_4 :\equiv \Sigma(t_3, \underline{t_1}, \underline{a_1}) = \lambda \underline{a'}.\, t_3(\underline{\widetilde{a}}, \underline{t_1}(\underline{a}))$ by replacing the variables $\underline{a_1}$ with constants $\mathtt{0}$ of corresponding types. There exists $k \in \mathbb{N}$ constant such that the following hold :

$$d(\underline{t_4}) \;\leq\; qs(A \to B) + d(\underline{t_1}, \underline{t_3}) \tag{15}$$

$$S(\underline{t_4}) \;\leq\; 1 + qs(A \to B) \cdot S(\underline{t_1}, \underline{t_3}) \tag{16}$$

$$dg(\underline{t_4}) \;\leq\; dg(\underline{t_3}) \qquad \text{and} \qquad ar(\underline{t_4}) \;\leq\; ar(\underline{t_3})$$

$$mdg(\underline{t_4}) \;\leq\; max\{mdg(\underline{t_1}, \underline{t_3})\,,\, dg(\underline{t_3})+1\}$$

$$\begin{aligned} mar(\underline{t_4}) \;&\leq\; max\{mar(\underline{t_1},\underline{t_3})\,,\, ar(\underline{t_3})+1\,,\, ar(\underline{a_1})\} \\ \{\!|\, \underline{t_1}\,,\, A \,|\!\} \quad , \quad \{\!|\, \underline{t_2},\underline{t_3}\,,\, A\to B \,|\!\} \;&\vdash_k\; \{\!|\, \underline{t_4}\,,\, B \,|\!\} \end{aligned}$$

Proof: There exists $k\in\mathbb{N}$ constant such that for all $(\underline{t_2},\underline{t_3},A\to B)\in PR$ and $(\underline{t_1},A)\in PR$ the following deductions hold:

$$\underline{y} :\equiv \underline{t_2}(\underline{\tilde{a}},\underline{t_1}(\underline{a}),\underline{v}) \quad \frac{\forall \underline{y}\, A_{\mathtt{D}}(\underline{t_1}(\underline{a});\underline{y})}{A_{\mathtt{D}}(\underline{t_1}(\underline{a});\underline{t_2}(\underline{\tilde{a}},\underline{t_1}(\underline{a}),\underline{v}))}\; k$$

$$\underline{x} :\equiv \underline{t_1}(\underline{a}) \quad \frac{\forall \underline{x},\underline{v}\,(A_{\mathtt{D}}(\underline{x};\underline{t_2}(\underline{\tilde{a}},\underline{x},\underline{v}))\to B_{\mathtt{D}}(\underline{t_3}(\underline{\tilde{a}},\underline{x});\underline{v}))}{A_{\mathtt{D}}(\underline{t_1}(\underline{a});\underline{t_2}(\underline{\tilde{a}},\underline{t_1}(\underline{a}),\underline{v}))\to B_{\mathtt{D}}(\underline{t_3}(\underline{\tilde{a}},\underline{t_1}(\underline{a}));\underline{v})}\; k$$

By using MP once we thus obtain that there exists $k\in\mathbb{N}$ constant such that for all $(\underline{t_1},A)$ and $(\underline{t_2},\underline{t_3},A\to B)$ members of *PR* the following deduction holds:

$$\frac{\forall \underline{y}\, A_{\mathtt{D}}(\underline{t_1}(\underline{a});\underline{y}) \qquad \forall \underline{x},\underline{v}\,(A_{\mathtt{D}}(\underline{x};\underline{t_2}(\underline{\tilde{a}},\underline{x},\underline{v}))\to B_{\mathtt{D}}(\underline{t_3}(\underline{\tilde{a}},\underline{x});\underline{v}))}{B_{\mathtt{D}}(\underline{t_3}(\underline{\tilde{a}},\underline{t_1}(\underline{a}));\underline{v})}\; k$$

By $\mathtt{Ax}\Sigma$ there exists $k\in\mathbb{N}$ constant such that for all $(\underline{t_1},A)$ and $(\underline{t_2},\underline{t_3},A\to B)$ members of *PR*, $\vdash_k t_3(\underline{\tilde{a}},\underline{t_1}(\underline{a}))=t_4'(\underline{a'})$ holds. Since $B_{\mathtt{D}}$ is quantifier–free, we obtain from (1) that there exists $k\in\mathbb{N}$ constant such that for all $(\underline{t_1},A)$ and $(\underline{t_2},\underline{t_3},A\to B)\in PR$ the deduction $B_{\mathtt{D}}(\underline{t_3}(\underline{\tilde{a}},\underline{t_1}(\underline{a}));\underline{v})\vdash_k B_{\mathtt{D}}(\underline{t_4'}(\underline{a'});\underline{v})$ holds. We conclude that there exists $k\in\mathbb{N}$ constant such that for all $(\underline{t_1},A)$ and $(\underline{t_2},\underline{t_3},A\to B)$ members of *PR*, the following deduction holds:

$$\frac{\forall \underline{y}\, A_{\mathtt{D}}(\underline{t_1}(\underline{a});\underline{y}) \qquad \forall \underline{x},\underline{v}\,(A_{\mathtt{D}}(\underline{x};\underline{t_2}(\underline{\tilde{a}},\underline{x},\underline{v}))\to B_{\mathtt{D}}(\underline{t_3}(\underline{\tilde{a}},\underline{x});\underline{v}))}{\forall \underline{v}\, B_{\mathtt{D}}(\underline{t_4'}(\underline{a'});\underline{v})}\; k$$

Since $|\underline{t_1},\underline{a_1}|+1\leq qs(A)+1\leq qs(A\to B)$ (for the second inequality here we also used that B is non-realizer-free), the inequalities (15) and (16) follow from

$$\begin{aligned} d(\underline{t_4}) \;&\leq\; |\underline{t_1},\underline{a_1}|+1+max\{d(\underline{t_1})\,,\,d(\underline{t_3})\} \\ S(\underline{t_4}) \;&\leq\; 1+(|\underline{t_1},\underline{a_1}|+1)\cdot max\{S(\underline{t_1})\,,\,S(\underline{t_3})\}\;. \end{aligned}$$

The remaining inequalities follow immediately from

$$\begin{aligned} dg(\underline{t_4}) \;&\leq\; dg(\underline{t_3}) \qquad dg(\underline{a_1}) \;\leq\; dg(\Sigma) \;=\; dg(t_3)+1 \\ ar(\underline{t_4}) \;&\leq\; ar(\underline{t_3}) \qquad ar(\Sigma) \;=\; ar(t_3)+1 \end{aligned}$$

which are proved using $\; t_3(\underline{\tilde{a}},\underline{t_1}(\underline{a})) \;=\; t_4'(\underline{a'}) \;=\; \Sigma(t_3,\underline{t_1},\underline{a_1},\underline{a'})\quad.\;\Box$

Lemma 3.15 ($\mathtt{QR}\forall,\mathtt{QR}\exists$) Let $\mathfrak{A}_{\mathtt{QR}\forall}$ be the algorithm that produces the output $(\underline{t_3},\underline{t_4},A(\underline{a})\to\forall\underline{z}\,B(\underline{a'},\underline{z}))\in PR$ from an input $(\underline{t_1},\underline{t_2},A(\underline{a})\to B(\underline{a'},\underline{z}))\in PR$, where $t_3:\equiv P\,t_1=\lambda\underline{\tilde{a}},\underline{x},\underline{z}.\,t_1(\underline{\tilde{a}},\underline{z},\underline{x})$ and $t_4:\equiv P\,t_2=\lambda\underline{\tilde{a}},\underline{x},\underline{z}.\,t_2(\underline{\tilde{a}},\underline{z},\underline{x})$ with $\{\underline{\tilde{a}}\}=\{\underline{a}\}\cup\{\underline{a'}\}$. There exists $k\in\mathbb{N}$ constant such that the following hold:

$$\begin{aligned} d(\underline{t_3},\underline{t_4}) \;&\leq\; d(\underline{t_1},\underline{t_2})+1 \quad\text{and}\quad dg(\underline{t_3},\underline{t_4}) \;=\; dg(\underline{t_1},\underline{t_2}) \\ S(\underline{t_3},\underline{t_4}) \;&\leq\; S(\underline{t_1},\underline{t_2})+1 \quad\text{and}\quad ar(\underline{t_3},\underline{t_4}) \;=\; ar(\underline{t_1},\underline{t_2}) \\ mdg(\underline{t_3},\underline{t_4}) \;&\leq\; max\{mdg(\underline{t_1},\underline{t_2})\,,\,dg(\underline{t_1},\underline{t_2})+1\} \\ mar(\underline{t_3},\underline{t_4}) \;&\leq\; max\{mdg(\underline{t_1},\underline{t_2})\,,\,ar(\underline{t_1},\underline{t_2})+1\} \\ \{\!|\, \underline{t_1},\underline{t_2}\,,\, A\to B(\underline{z}) \,|\!\} \;&\vdash_k\; \{\!|\, \underline{t_3},\underline{t_4}\,,\, A\to\forall\underline{z}\,B(\underline{z}) \,|\!\} \end{aligned}$$

A corresponding statement holds for $\mathtt{QR}\exists$ as well, with the same bounds.

Proof: By definition,

$$(A \to B(\underline{a'}, \underline{z}))^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{U}\, \forall \underline{x}, \underline{v}\, [A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v}; \underline{a'}, \underline{z})]$$

$$(A \to \forall \underline{z} B(\underline{a'}, \underline{z}))^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{U}\, \forall \underline{x}, \underline{z}, \underline{v}\, [A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{z}, \underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x}, \underline{z}); \underline{v}; \underline{a'}, \underline{z})] \, .$$

By $\mathtt{Ax}P$, there exists $k \in \mathbb{N}$ constant such that for all $(\underline{t_1}, \underline{t_2}, A \to B(\underline{z})) \in PR$,

$$\vdash_k t_3(\underline{\tilde{a}}, \underline{x}, \underline{z}, \underline{v}) = t_1(\underline{\tilde{a}}, \underline{z}, \underline{x}, \underline{v}) \qquad \text{and} \qquad \vdash_k t_4(\underline{\tilde{a}}, \underline{x}, \underline{z}) = t_2(\underline{\tilde{a}}, \underline{z}, \underline{x}) \, .$$

Since $A_{\mathtt{D}}(\underline{x}; \underline{y}) \to B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{a'}, \underline{z})$ is quantifier-free, by using (1) we obtain that there exists $k \in \mathbb{N}$ constant such that for all $(\underline{t_1}, \underline{t_2}, A \to B(\underline{z}))$ member of PR,

$$\frac{A_{\mathtt{D}}(\underline{x}; \underline{t_1}(\underline{\tilde{a}}, \underline{z}, \underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{t_2}(\underline{\tilde{a}}, \underline{z}, \underline{x}); \underline{v}; \underline{a'}, \underline{z})}{A_{\mathtt{D}}(\underline{x}; \underline{t_3}(\underline{\tilde{a}}, \underline{x}, \underline{z}, \underline{v})) \to B_{\mathtt{D}}(\underline{t_4}(\underline{\tilde{a}}, \underline{x}, \underline{z}); \underline{v}; \underline{a'}, \underline{z})}\, k \, .$$

Further, there exists $k \in \mathbb{N}$ constant such that for all $(\underline{t_1}, \underline{t_2}, A \to B(\underline{z})) \in PR$,

$$\frac{\forall \underline{x}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{t_1}(\underline{\tilde{a}}, \underline{z}, \underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{t_2}(\underline{\tilde{a}}, \underline{z}, \underline{x}); \underline{v}; \underline{a'}, \underline{z}))}{\forall \underline{x}, \underline{z}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{t_3}(\underline{\tilde{a}}, \underline{x}, \underline{z}, \underline{v})) \to B_{\mathtt{D}}(\underline{t_4}(\underline{\tilde{a}}, \underline{x}, \underline{z}); \underline{v}; \underline{a'}, \underline{z}))}\, k \, .$$

Obviously,

- $dg(t_3) = dg(t_1)$ and $dg(t_4) = dg(t_2)$, therefore $dg(\underline{t_3}, \underline{t_4}) = dg(\underline{t_1}, \underline{t_2})$
- $ar(t_3) = ar(t_1)$ and $ar(t_4) = ar(t_2)$, therefore $ar(\underline{t_3}, \underline{t_4}) = ar(\underline{t_1}, \underline{t_2})$

and the inequalities in the conclusion of this Lemma follow immediately. □

Lemma 3.16 ($\mathtt{EXP}, \mathtt{IMP}$) The following holds:

$$\{\!\!\{ \underline{t_1}, \underline{t_2}, \underline{t_3}\, ,\, A \to (B \to C) \}\!\!\} = \{\!\!\{ \underline{t_1}, \underline{t_2}, \underline{t_3}\, ,\, A \wedge B \to C \}\!\!\} \, .$$

Proof: By definition,

$$(A \wedge B \to C)^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{V}, \underline{G}\, \forall \underline{x}, \underline{u}, \underline{h}$$
$$[A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{u}, \underline{h})) \wedge B_{\mathtt{D}}(\underline{u}; \underline{V}(\underline{x}, \underline{u}, \underline{h})) \to C_{\mathtt{D}}(\underline{G}(\underline{x}, \underline{u}); \underline{h})]$$

$$(A \to B \to C)^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{V}, \underline{G}\, \forall \underline{x}, \underline{u}, \underline{h}$$
$$[A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{u}, \underline{h})) \to B_{\mathtt{D}}(\underline{u}; \underline{V}(\underline{x}, \underline{u}, \underline{h})) \to C_{\mathtt{D}}(\underline{G}(\underline{x}, \underline{u}); \underline{h})] \, .$$

Theorem 3.17 There exists $k \in \mathbb{N}$ constant and an algorithm $\mathfrak{A}$ which does the following. Let $\mathcal{P}$ be some proof of a formula A in $\mathtt{EIL}^{\omega}_{+}$ and $\underline{s}_{(\cdot)} \in RTS[\mathrm{Lv}(\mathcal{P})]$ a realizing tuple selection for the set of leaves of $\mathcal{P}$. Let $q_{\mathrm{o}} :\equiv max_{A \in \mathrm{Lv}(\mathcal{P})}\, q(\underline{s}_A)$ for $q \in \{d, S, dg, ar, mdg, mar\}$ and $q_{\mathrm{o}} :\equiv q(\mathrm{Lv}(\mathcal{P}))$ for $q \in \{qs, var\}$. Let[26] $\partial_{\mathtt{MP}} :\equiv \partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}} :\equiv \partial_{\mathtt{QR}}(\mathcal{P})$ and $\partial :\equiv \partial(\mathcal{P})$. Let $\partial_{\mathrm{o}} \in \mathbb{N}$ be a number such that for all $A \in \mathrm{Lv}(\mathcal{P})$, $\vdash_{\partial_{\mathrm{o}}} \{\!\!\{ \underline{s}_A, A \}\!\!\}$. When $\mathfrak{A}$ is presented with $\mathcal{P}$ and $\underline{s}_{(\cdot)}$ at input, it produces as output $(\underline{t}, A) \in RR$ and the following hold:

[26] See Section 1.2 for the meaning of $\partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}}(\mathcal{P})$ and $\partial(\mathcal{P})$. Notice that $\mathtt{QR}\forall$, $\mathtt{QR}\exists$ and $\mathtt{MP}$ label edges in our $\mathtt{EIL}^{\omega}$-proof-trees $\mathcal{P}$ and $\mathtt{QR}$ accumulates both $\mathtt{QR}\forall$ and $\mathtt{QR}\exists$ labels.

$$
\begin{aligned}
d(\underline{t}) &\leq d_{\text{o}} + \partial_{\texttt{QR}} + qs_{\text{o}} \cdot (\partial_{\texttt{MP}} - k_0) && (17)\\
S(\underline{t}) &\leq (S_{\text{o}} + \partial_{\texttt{QR}} - k_0 + 1) \cdot qs_{\text{o}}^{(\partial_{\texttt{MP}} - k_{\text{o}})} && (18)\\
dg(\underline{t}) \leq dg_{\text{o}} &\quad \text{and} \quad mdg(\underline{t}) \leq mdg_{\text{o}} + 1 && (19)\\
ar(\underline{t}) \leq ar_{\text{o}} &\quad \text{and} \quad mar(\underline{t}) \leq max\{var_{\text{o}}\,,\, mar_{\text{o}} + 1\} && (20)\\
\texttt{EIL}^{\omega}_{+} &\vdash_{\partial_{\text{o}} + k\,\partial} \{\!\!\{\, \underline{t}\,,\, A \,\}\!\!\} && (21)
\end{aligned}
$$

Proof: The algorithm proceeds by recursion on the structure of $\mathcal{P}$, using the algorithms in Lemmas 3.14 and 3.15 as subprocedures at the MP, respectively QR recursion steps; (21) follows immediately. We notice that dg and ar do not increase in the recursion, hence (19) and (20) are clear. Let $\underline{e} \equiv e_1 \ldots e_n$ denote paths from some leaf to the root of $\mathcal{P}$, i.e., $(e_i)_{i \in \overline{1,n}}$ denote edges such that e_1 is incident with a leaf and e_n is incident with the root of $\mathcal{P}$. Let $(d^{\underline{e}}_i\ ,\ S^{\underline{e}}_i)_{i \in \overline{0,n}}$ be a sequence of pairs of natural numbers defined by $(d^{\underline{e}}_0\ ,\ S^{\underline{e}}_0) :\equiv (d_{\text{o}}\ ,\ S_{\text{o}})$ and let

$$
(d^{\underline{e}}_i\ ,\ S^{\underline{e}}_i)\ :\equiv \begin{cases} (d^{\underline{e}}_{i-1} + qs_{\text{o}}\ ,\ qs_{\text{o}} \cdot S^{\underline{e}}_{i-1} + 1)\,, & \text{if } L(e_i) = \texttt{MP} \\ (d^{\underline{e}}_{i-1} + 1\ ,\ S^{\underline{e}}_{i-1} + 1) \quad , & \text{if } L(e_i) \in \texttt{QR} \\ (d^{\underline{e}}_{i-1}\ ,\ S^{\underline{e}}_{i-1}) \quad , & \text{otherwise} \end{cases} \cdot
$$

with $i \in \overline{1,n}$. Using (13) it follows that $max_{\underline{e}}\, d^{\underline{e}}_n$ and $max_{\underline{e}}\, S^{\underline{e}}_n$ are upper bounds on d, S respectively. Inequalities (17) and (18) follow now immediately[27]. $\Box$

Remark 3.18 Let us suppose that only unary (i.e., with $n = 1$) $\texttt{ER}_0$ is allowed in the verifying proof. The n-ary $\texttt{ER}_0$ can be obtained from unary $\texttt{ER}_0$ with a proof of depth proportional with n. It follows that we can upper bound the depths of proofs of lemmas used in verifying MP and QR with quantities proportional with qs_{o}. In consequence, (21) becomes $\texttt{EIL}^{\omega}_{+} \vdash_{\partial_{\text{o}} + k \cdot (qs_{\text{o}} + \partial)} \{\!\!\{\, \underline{t}\,,\, A \,\}\!\!\}$.

3.3 Bounds for realizing terms for $\texttt{EIL}^{\omega}_{+}$+AC+$\texttt{IP}_{\forall}$+MK *axioms*

Remark 3.19 Recall that the formal proofs below are by default in $\texttt{EIL}^{\omega}$.

Proposition 3.20 There exists $k \in \mathbb{N}$ constant such that for any instance A of CT$\vee$, WK$\vee$, WK$\wedge$, PM$\vee$, PM$\wedge$, SYL, EPN, EFQ, $\texttt{TND}_0$, MK, $\texttt{IP}_{\forall}$, AC, there exists a realizing tuple $\underline{t}$ for A^{D} such that:

$$
\begin{aligned}
d(\underline{t}) &\leq k && (22)\\
S(\underline{t}) &\leq k && (23)\\
mdg(\underline{t}) &\leq k + vdg(A) + id(A) && (24)\\
mar(\underline{t}) &\leq k + var(A) + qs(A) \cdot (id(A) - k_0 + 2) && (25)\\
\texttt{EIL}^{\omega} &\vdash_k \{\!\!\{\, \underline{t}\,,\, A \,\}\!\!\} && (26)
\end{aligned}
$$

[27] At (18) an intermediate upper bound is $(S_{\text{o}} + \partial_{\texttt{QR}} - k_0) \cdot qs_{\text{o}}^{(\partial_{\texttt{MP}} - k_{\text{o}})} + \sum_{i=0}^{(\partial_{\texttt{MP}} - k_0) - 1} qs_{\text{o}}^{\,i}$.

Proof: We treat here $\mathtt{SYL}$ as an example, since it is the most complex among the above listed axioms. All remaining axioms are treated in the Appendix of [26]. We have

$$((A \to B) \wedge (B \to C) \to (A \to C))^{\mathtt{D}} \;\equiv\; \exists \underline{X}, \underline{V}, \underline{U'}, \underline{H}, \underline{Y'}, \underline{G'}\; \forall \underline{y}, \underline{u}, \underline{v'}, \underline{g}, \underline{x'}, \underline{h'}$$

$$\left(\begin{array}{c}
\left(\begin{array}{c}
A_{\mathtt{D}}(\underline{X}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'});\underline{y}(\underline{X}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'}),\underline{V}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'})))\\
\to\\
B_{\mathtt{D}}(\underline{u}(\underline{X}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'}));\underline{V}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'}))
\end{array}\right)\\
\bigwedge\\
\left(\begin{array}{c}
B_{\mathtt{D}}(\underline{U'}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'});\underline{v'}(\underline{U'}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'}),\underline{H}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'})))\\
\to\\
C_{\mathtt{D}}(\underline{g}(\underline{U'}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'}));\underline{H}(\underline{y},\underline{u},\underline{v'},\underline{g},\underline{x'},\underline{h'}))
\end{array}\right)\\
\longrightarrow\\
\left(\begin{array}{c}
A_{\mathtt{D}}(\underline{x'};\underline{Y'}(\underline{y},\underline{u},\underline{v'},\underline{g})(\underline{x'},\underline{h'}))\\
\to\\
C_{\mathtt{D}}(\underline{G'}(\underline{y},\underline{u},\underline{v'},\underline{g})(\underline{x'});\underline{h'})
\end{array}\right)
\end{array}\right)$$

from

$$((A \to B) \wedge (B \to C))^{\mathtt{D}} \;\equiv\; \exists \underline{Y}, \underline{U}, \underline{V'}, \underline{G}\; \forall \underline{x}, \underline{v}, \underline{u'}, \underline{h}$$
$$((A_{\mathtt{D}}(\underline{x};\underline{Y}(\underline{x},\underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x});\underline{v})) \wedge (B_{\mathtt{D}}(\underline{u'};\underline{V'}(\underline{u'},\underline{h})) \to C_{\mathtt{D}}(\underline{G}(\underline{u'});\underline{h})))$$

from

$$\begin{aligned}
(A \to B)^{\mathtt{D}} \;&\equiv\; \exists \underline{Y}, \underline{U}\, \forall \underline{x}, \underline{v}\; (A_{\mathtt{D}}(\underline{x};\underline{Y}(\underline{x},\underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x});\underline{v}))\\
(B \to C)^{\mathtt{D}} \;&\equiv\; \exists \underline{V'}, \underline{G}\, \forall \underline{u'}, \underline{h}\; (B_{\mathtt{D}}(\underline{u'};\underline{V'}(\underline{u'},\underline{h})) \to C_{\mathtt{D}}(\underline{G}(\underline{u'});\underline{h}))\\
(A \to C)^{\mathtt{D}} \;&\equiv\; \exists \underline{Y'}, \underline{G'}\, \forall \underline{x'}, \underline{h'}\; (A_{\mathtt{D}}(\underline{x'};\underline{Y'}(\underline{x'},\underline{h'})) \to C_{\mathtt{D}}(\underline{G'}(\underline{x'});\underline{h'}))
\end{aligned}$$

and we can take (below $\{\underline{a}\} = \mathcal{V}_{\mathrm{f}}((A \to B) \wedge (B \to C) \to (A \to C))$)

$$\begin{aligned}
t_X \;&:\equiv\; \mathit{\Pi} = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{v'}, \underline{g}, \underline{x'}, \underline{h'}.\, x'\\
t_H \;&:\equiv\; \mathit{\Pi} = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{v'}, \underline{g}, \underline{x'}, \underline{h'}.\, h'\\
t_{U'} \;&:\equiv\; P\,\mathit{\Sigma} = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{v'}, \underline{g}, \underline{x'}, \underline{h'}.\, u(\underline{x'})\\
t_V \;&:\equiv\; P\,\mathit{\Sigma} = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{v'}, \underline{g}, \underline{x'}.\, v'(\underline{u}(\underline{x'}))\\
t_{Y'} \;&:\equiv\; P\,(\mathit{\Sigma}\,\mathit{\Sigma}) = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{v'}, \underline{g}, \underline{x'}, \underline{h'}.\, y(\underline{x'}, \underline{v'}(\underline{u}(\underline{x'}), \underline{h'}))\\
t_{G'} \;&:\equiv\; P\,\mathit{\Sigma} = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{v'}, \underline{g}, \underline{x'}.\, g(\underline{u}(\underline{x'}))
\end{aligned}$$

The proofs of (22) and (23) are immediate, for (24) and (25) we use the results in Proposition 2.9 plus (10), respectively (9, 11) and for (26) we use $\mathtt{Ax}\mathit{\Sigma}$, $\mathtt{Ax}P$, $\mathtt{Ax}\mathit{\Pi}$, $\mathtt{Ax}\mathcal{D}$ and (1) . □

Proposition 2.10 gives us an algorithm for associating terms $t_{A_{\mathtt{D}}}$ to formulas A such that $\vdash A_{\mathtt{D}} \leftrightarrow t_{A_{\mathtt{D}}} = 0$. Since $\mathcal{V}(t_{A_{\mathtt{D}}}) = \mathcal{V}_{\mathrm{f}}(A_{\mathtt{D}})$ these $t_{A_{\mathtt{D}}}$ are generally not closed, whereas we want to produce closed realizing terms for contractions $A \to A \wedge A$. We could certainly close these $t_{A_{\mathtt{D}}}$ via the λ-abstraction algorithm of Definition 2.12 . However this would force us to count in our complexity analysis the full size of the quantifier axioms terms and of the terms t_1, t_2 which appear in prime formulas $t_1 = t_2$ of contractions $A \to A \wedge A$. This is exactly what we want to avoid, remember the comment at the end of Section 1.1 , Definition 3.5 and its preceding comment. We therefore give an association of closed terms to all $\mathtt{EIL}^{\omega}_{+}$ formulas such that $\tilde{\cdot}$ *constants* are used instead of the original building terms.

Proposition 3.21 (Association of *closed* terms to *all* $\mathtt{EIL}^{\omega}_{+}$ formulas) There exists $k \in \mathbb{N}$ constant and an association of terms to $\mathtt{EIL}^{\omega}_{+}$ formulas $A \mapsto t^{\mathtt{D}}_{A}$ such that for all A (with $\{\underline{a}\} = \mathcal{V}_{\mathrm{f}}(A)$)

$$\begin{aligned} d(t^{\mathtt{D}}_{A}) &\le k \cdot ld(A) & (27)\\ S(t^{\mathtt{D}}_{A}) &\le k \cdot ls(A) & (28)\\ mdg(t^{\mathtt{D}}_{A}) &\le k + vdg(A) + id(A) & (29)\\ mar(t^{\mathtt{D}}_{A}) &\le k + var(A) + qs(A) \cdot (id(A) - k_0 + 2) & (30)\end{aligned}$$

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \vdash_{k \cdot ld(A)} A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) \;\leftrightarrow\; t^{\mathtt{D}}_{A}(\underline{x}, \underline{y}, \underline{a}) = 0 \ . \tag{31}$$

The $\tilde{\cdot}$ constants in (31) are only those corresponding to terms occurring in A .

Proof: Induction on the structure of A. For prime formulas just take $t^{\mathtt{D}}_{\perp} :\equiv 1$ and (below $\{\underline{a_1}\} = \mathcal{V}(t_1)$, $\{\underline{a_2}\} = \mathcal{V}(t_2)$ and $\{\underline{a}\} = \mathcal{V}_{\mathrm{f}}(t_1 = t_2)$)

$$t^{\mathtt{D}}_{t_1 = t_2} :\equiv \Sigma\, \mathtt{E}\, \widetilde{t_1}\, \widetilde{t_2} = \lambda \underline{a}.\, \mathtt{E}\, \widetilde{t_1}(\underline{a_1})\, \widetilde{t_2}(\underline{a_2})$$

and otherwise define (below $\{\underline{\tilde{a}}\} = \{\underline{a}\} \cup \{\underline{a'}\}$)

$$\begin{aligned} t^{\mathtt{D}}_{A \wedge B} &:\equiv \Sigma\, \nu\, t^{\mathtt{D}}_{A}\, t^{\mathtt{D}}_{B} = \lambda \underline{x}, \underline{u}, \underline{y}, \underline{v}, \underline{\tilde{a}}.\, \nu\, t^{\mathtt{D}}_{A}(\underline{x}, \underline{y}, \underline{a})\, t^{\mathtt{D}}_{B}(\underline{u}, \underline{v}, \underline{a'})\\ t^{\mathtt{D}}_{\exists z A(\underline{a}, z)} &:\equiv P\, t^{\mathtt{D}}_{A(\underline{a}, z)} = \lambda z, \underline{x}, \underline{y}, \underline{a}.\, t^{\mathtt{D}}_{A(\underline{a}, z)}(\underline{x}, \underline{y}, \underline{a}, z)\\ t^{\mathtt{D}}_{\forall z A(\underline{a}, z)} &:\equiv \Sigma\, t^{\mathtt{D}}_{A(\underline{a}, z)} = \lambda \underline{X}, z, \underline{y}, \underline{a}.\, t^{\mathtt{D}}_{A(\underline{a}, z)}(\underline{X}(z), \underline{y}, \underline{a}, z)\\ t^{\mathtt{D}}_{A \to B} &:\equiv \Sigma\, \Sigma\, \mathtt{I}\, t^{\mathtt{D}}_{A}\, t^{\mathtt{D}}_{B} = \lambda \underline{Y}, \underline{U}, \underline{x}, \underline{v}, \underline{\tilde{a}}.\, \mathtt{I}\, t^{\mathtt{D}}_{A}(\underline{x}, \underline{Y}(\underline{x}, \underline{v}), \underline{a})\, t^{\mathtt{D}}_{B}(\underline{U}(\underline{x}), \underline{v}, \underline{a'})\\ t^{\mathtt{D}}_{A \vee B} &:\equiv \Sigma\, \Sigma\, \nu\, \mathtt{I}\, t^{\mathtt{D}}_{A}\, t^{\mathtt{D}}_{B}\, \mathtt{I}\, 1 =\\ &\qquad = \lambda z, \underline{x}, \underline{u}, \underline{y}, \underline{v}, \underline{\tilde{a}}.\, \nu\, (\mathtt{I}\, z\, t^{\mathtt{D}}_{A}(\underline{x}, \underline{y}, \underline{a}))\, (\mathtt{I}\, (\mathtt{I}\, z\, 1)\, t^{\mathtt{D}}_{B}(\underline{u}, \underline{v}, \underline{a'}))\end{aligned}$$

The inequalities (27) and (28) are immediate, (29) and (30) follow from (10), respectively (9, 11) and (31) follows using the axioms $\mathtt{Ax}\Sigma$, $\mathtt{AxI}$, $\mathtt{Ax}\nu$, $\mathtt{AxE}$. $\square$

Proposition 3.22 There exists $k \in \mathbb{N}$ constant such that for any instance A of $\mathtt{CT}\wedge$ there exists a realizing tuple $\underline{t}$ for $A^{\mathtt{D}}$ such that

$$\begin{aligned} d(\underline{t}) &\le k \cdot ld(A)\\ S(\underline{t}) &\le k \cdot ls(A)\\ mdg(\underline{t}) &\le k + vdg(A) + id(A)\\ mar(\underline{t}) &\le k + var(A) + qs(A) \cdot (id(A) - k_0 + 3)\end{aligned}$$

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \vdash_{k \cdot ld(A)} \{\!\!\{\, \underline{t}\,,\, A \,\}\!\!\} \ . \tag{32}$$

The $\tilde{\cdot}$ constants in (32) are only those corresponding to terms occurring in A .

Proof: We have

$$(A \equiv B \to B \wedge B)^{\mathtt{D}} \;\equiv\; \exists \underline{Y}, \underline{X'}, \underline{X''}\, \forall \underline{x}, \underline{y'}, \underline{y''}$$
$$[\, B_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{y'}, \underline{y''})) \;\to\; B_{\mathtt{D}}(\underline{X'}(\underline{x}); \underline{y'}) \;\wedge\; B_{\mathtt{D}}(\underline{X''}(\underline{x}); \underline{y''}) \,]$$

and we can take (here $\{\underline{a}\} = \mathcal{V}_{\mathsf{f}}(A) = \mathcal{V}_{\mathsf{f}}(B)$)

$$t_{X'} \;:\equiv\; t_{X''} :\equiv \varPi = \lambda \underline{a}, \underline{x}.\, x$$

$$t_Y \;:\equiv\; \varSigma\, \mathcal{D}\, t_B^{\mathtt{D}} = \lambda \underline{a}, \underline{x}, \underline{y'}, \underline{y''}.\, \mathcal{D}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'})$$

We first prove (32). By $\mathtt{Ax}\mathcal{D}$, there exists $k \in \mathbb{N}$ constant such that for all B ,

$$\begin{cases} \vdash_k \; t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}) \;=\; 0 \;\to\; \mathcal{D}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'}) = y'' \\ \vdash_k \; \mathtt{I}\, t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a})\, 1 \;=0 \;\to \mathcal{D}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'}) = y' \end{cases}$$

and by using $\mathtt{ER_0}$, there exists $k \in \mathbb{N}$ constant such that for all B ,

$$\begin{cases} \vdash_k \; t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}) = 0 \;\to\; B_{\mathtt{D}}(\underline{x}; \underline{\mathcal{D}}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'})) \;\to\; B_{\mathtt{D}}(\underline{x}; \underline{y''}) \\ \vdash_k \; \mathtt{I}\, t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a})\, 1 = 0 \;\to\; B_{\mathtt{D}}(\underline{x}; \underline{\mathcal{D}}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'})) \;\to\; B_{\mathtt{D}}(\underline{x}; \underline{y'}) \;. \end{cases}$$

By $\mathtt{TND_0}$ and $\mathtt{AxI}$, there exists $k \in \mathbb{N}$ constant such that for all B ,

$$\vdash_k \; t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}) = 0 \;\vee\; \mathtt{I}\, t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a})\, 1 \;=\; 0 \;.$$

From (31) , there exists $k \in \mathbb{N}$ constant such that for all B ,

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \;\vdash_{k \cdot ld(B)}\; B_{\mathtt{D}}(\underline{x}; \underline{y'}) \;\leftrightarrow\; t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}) = 0 \;,$$

hence there exists $k \in \mathbb{N}$ constant such that for all B ,

$$\begin{cases} \mathtt{EIL}^{\omega}_{\mathtt{v}} \;\vdash_{k \cdot ld(B)}\; B_{\mathtt{D}}(\underline{x}; \underline{y'}) \to B_{\mathtt{D}}(\underline{x}; \underline{\mathcal{D}}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'})) \to B_{\mathtt{D}}(\underline{x}; \underline{y''}) \\ \mathtt{EIL}^{\omega}_{\mathtt{v}} \;\vdash_{k \cdot ld(B)}\; \neg B_{\mathtt{D}}(\underline{x}; \underline{y'}) \to B_{\mathtt{D}}(\underline{x}; \underline{\mathcal{D}}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'})) \to B_{\mathtt{D}}(\underline{x}; \underline{y'}) \;. \end{cases}$$

Since there exists $k \in \mathbb{N}$ constant such that for all A, B and C ,

$$A \vee \neg A \,,\; A \to B \to C \,,\; \neg A \to B \to A \;\vdash_k\; B \to A \wedge C \;,$$

we finally obtain that there exists $k \in \mathbb{N}$ constant such that for all B ,

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \;\vdash_{k \cdot ld(B)}\; B_{\mathtt{D}}(\underline{x}; \underline{\mathcal{D}}(t_{\mathtt{B}}^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'})) \to B_{\mathtt{D}}(\underline{x}; \underline{y'}) \wedge B_{\mathtt{D}}(\underline{x}; \underline{y''}) \;.$$

Since there exists $k \in \mathbb{N}$ constant such that for all B ,

$$\vdash_k \; t_Y(\underline{a}, \underline{x}, \underline{y'}, \underline{y''}) = \mathcal{D}(t_B^{\mathtt{D}}(\underline{x}, \underline{y'}, \underline{a}), \underline{y''}, \underline{y'})$$

$$\vdash_k \; t_{X'}(\underline{a}, \underline{x}) = x$$

$$\vdash_k \; t_{X''}(\underline{a}, \underline{x}) = x \,,$$

we obtain from (1) that there exists $k \in \mathbb{N}$ constant such that

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \;\vdash_{k \cdot ld(B)}\; B_{\mathtt{D}}(\underline{x}; \underline{t_Y}(\underline{a}, \underline{x}, \underline{y'}, \underline{y''})) \to B_{\mathtt{D}}(\underline{t_{X'}}(\underline{a}, \underline{x}); \underline{y''}) \wedge B_{\mathtt{D}}(\underline{t_{X''}}(\underline{a}, \underline{x}); \underline{y'}) \,.$$

This gives (32) . The other inequalities follow directly from Proposition 3.21 .

Proposition 3.23 There exists $k \in \mathbb{N}$ constant such that for every instance $A(\underline{s})$ of $\mathtt{QA}\forall$ or $\mathtt{QA}\exists$ there exists a realizing tuple $\underline{t}$ for $A^{\mathtt{D}}$ such that :

$$\begin{aligned} d(\underline{t}) \;&\leq\; k + fd(A), \text{ when } A(\underline{s}) \in \mathtt{QA}\forall \text{ and} && (33)\\ d(\underline{t}) \;&\leq\; k, \text{ when } A(\underline{s}) \in \mathtt{QA}\exists \\ S(\underline{t}) \;&\leq\; k + fd(A), \text{ when } A(\underline{s}) \in \mathtt{QA}\forall \text{ and} && (34)\\ S(\underline{t}) \;&\leq\; k, \text{ when } A(\underline{s}) \in \mathtt{QA}\exists \\ mdg(\underline{t}) \;&\leq\; k + vdg(A) + id(A) && (35)\\ mar(\underline{t}) \;&\leq\; k + var(A) + qs(A)\cdot(id(A) - k_0 + 2) && (36)\\ \mathtt{EIL}^{\omega}_{\mathtt{v}} \;&\vdash_k\; \{\!\!|\, \underline{t}\,,\, A \,|\!\!\} && (37)\end{aligned}$$

The $\widetilde{\cdot}$ constants in (37) are only those corresponding to terms occurring in A.

Proof: Let $A(\underline{s}) \equiv \forall \underline{z} B(\underline{z}, \underline{a''}) \to B(\underline{s}, \underline{a''})$ be an instance of $\mathtt{QA}\forall$, s free for z in B. Let $\underline{a'} :\equiv \mathcal{V}(\underline{s})$ and $\underline{a} :\equiv \underline{a'}, \underline{a''} = \mathcal{V}_{\mathsf{f}}(A(\underline{s}))$. Also $\underline{s} \equiv s_1, \ldots, s_n$ and let $\underline{a_i} :\equiv \mathcal{V}(s_i)$ for $i \in \overline{1,n}$. We have that $(\forall \underline{z} B(\underline{z}) \to B(\underline{s}))^{\mathtt{D}}$ is given by

$$\exists \underline{Z}, \underline{Y}, \underline{X}\; \forall \underline{x}, \underline{y}\; [\; B_{\mathtt{D}}(\underline{x}(\underline{Z}(\underline{x}, \underline{y})); \underline{Y}(\underline{x}, \underline{y}); \underline{Z}(\underline{x}, \underline{y})) \;\to\; B_{\mathtt{D}}(\underline{X}(\underline{x}); \underline{y}; \underline{s})\;]$$

and we can take (recall from Definition 3.10 that $\widetilde{s_i}(\underline{a_i}) = s_i$)

$$\begin{aligned} t_{Z_i} \;&:\equiv\; \Sigma'\, \widetilde{s_i} = [\lambda u_i, \underline{a}, \underline{x}, \underline{y}.\, u_i(\underline{a_i})]\widetilde{s_i} = \lambda \underline{a}, \underline{x}, \underline{y}.\, \widetilde{s_i}(\underline{a_i})\\ t_Y \;&:\equiv\; \Pi = \lambda \underline{a}, \underline{x}, \underline{y}.\, y\\ t_X \;&:\equiv\; P\, \Sigma\, \widetilde{\underline{s}} = [\lambda \underline{u}, \underline{a}, \underline{x}.\, x(u_1(\underline{a_1}), \ldots, u_n(\underline{a_n}))]\, \widetilde{\underline{s}}\\ &\qquad\qquad = \lambda \underline{a}, \underline{x}.\, x(\widetilde{s_1}(\underline{a_1}), \ldots, \widetilde{s_n}(\underline{a_n}))\end{aligned}$$

From Proposition 2.9, $typ(u_i) = typ(\widetilde{s_i})$ and (12) it immediately follows that

$$\begin{aligned} dg(\Sigma') \;&\leq\; 2 + max\{dg(\underline{a}, \underline{x}, \underline{y})\,,\, dg(s_i)\}\\ ar(\Sigma') \;&\leq\; 1 + |\underline{a}, \underline{x}, \underline{y}| + ar(s_i)\\ dg(\Pi) \;&\leq\; 1 + dg(\underline{a}, \underline{x}, \underline{y})\\ ar(\Pi) \;&\leq\; |\underline{a}, \underline{x}, \underline{y}| + ar(y)\\ dg(P) \;&\leq\; 1 + dg(\Sigma) \;\leq\; 3 + max\{dg(\underline{a}, \underline{x})\,,\, dg(\underline{s})\}\\ ar(P) \;&\leq\; 1 + ar(\Sigma) \;\leq\; 2 + |\underline{a}, \underline{x}| + |\underline{s}| + ar(x)\end{aligned}$$

and (35), (36) now follow immediately from (10), respectively (9, 11), also using that $|\underline{z}| = |\underline{s}|$ and $typ(z_i) = typ(s_i)$. The inequalities (33) and (34) are immediate from $|\underline{s}| \leq fd(A)$. The proof of (37) uses the fact (which follows from (1)) that there exists $k \in \mathbb{N}$ constant such that for all $A(\underline{s})$,

$$\vdash_k\; B_{\mathtt{D}}(\, \underline{x}(\widetilde{s_1}(\underline{a_1}) ... \widetilde{s_n}(\underline{a_n}))\,; \underline{y};\; \widetilde{s_1}(\underline{a_1}) ... \widetilde{s_n}(\underline{a_n})\,) \to B_{\mathtt{D}}(\, \underline{x}(\widetilde{s_1}(\underline{a_1}) ... \widetilde{s_n}(\underline{a_n}))\,; \underline{y};\; \underline{s}\,)\,.$$

Let $A(\underline{s}) \equiv B(\underline{s}, \underline{a''}) \to \exists \underline{z} B(\underline{z}, \underline{a''})$ be an instance of $\mathtt{QA}\exists$, s free for z in B. The tuples $\underline{a'}$, $\underline{a}$ and $\underline{a_i}$ below are defined like in the $\mathtt{QA}\forall$ case above. We have

$$(B(\underline{s}) \to \exists \underline{z} B(\underline{z}))^{\mathtt{D}} \;\equiv\; \exists \underline{Y}, \underline{Z}, \underline{X}\, \forall \underline{x}, \underline{y}\, [\, B_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{y}); \underline{s}) \;\to\; B_{\mathtt{D}}(\underline{X}(\underline{x}); \underline{y}; \underline{Z}(\underline{x}))\,]$$

and we can take (recall from Definition 3.10 that $\widetilde{s_i}(\underline{a_i}) = s_i$)

$$\begin{aligned} t_Y &:\equiv \Pi = \lambda \underline{a}, \underline{x}, \underline{y}.\, y \\ t_{Z_i} &:\equiv \Sigma\, \tilde{s}_i = (\lambda u_i, \underline{a}, \underline{x}.\, u_i(\underline{a_i}))\tilde{s}_i = \lambda \underline{a}, \underline{x}.\, \tilde{s}_i(\underline{a_i}) \\ t_X &:\equiv \Pi = \lambda \underline{a}, \underline{x}.\, x \end{aligned}$$

The inequalities (33) and (34) are trivial, (35), (36) follow with an argument similar to the one in the QA∀ case. For (37) we use the fact (which follows from (1)) that there exists $k \in \mathbb{N}$ constant such that for all $A(\underline{s})$,

$$\vdash_k\ B_{\mathtt{D}}(\,\underline{x}\,;\,\underline{y}\,;\,s_1,\ldots,s_n\,) \to B_{\mathtt{D}}(\,\underline{x}\,;\,\underline{y}\,;\widetilde{s_1}(\,\underline{a_1}),\ldots,\widetilde{s_n}(\underline{a_n})\,)\ .$$

Notation 3.24 We will denote by $qs_0(\mathcal{P}) :\equiv max\{2\,,\, qs(\mathrm{Lv}(\mathcal{P}))\}$ and

$$\begin{aligned} vdg_0(\mathcal{P}) &:\equiv vdg(\mathrm{Lv}(\mathcal{P})) & var_0(\mathcal{P}) &:\equiv var(\mathrm{Lv}(\mathcal{P})) \\ fd_1(\mathcal{P}) &:\equiv fd(\mathtt{QA}\forall \cap \mathrm{Lv}(\mathcal{P})) & id_0(\mathcal{P}) &:\equiv id(\mathrm{Lv}(\mathcal{P})) \\ ld_1(\mathcal{P}) &:\equiv ld(\mathtt{CT}\wedge \cap \mathrm{Lv}(\mathcal{P})) & ls_1(\mathcal{P}) &:\equiv ls(\mathtt{CT}\wedge \cap \mathrm{Lv}(\mathcal{P})) \\ fid_0(\mathcal{P}) &:\equiv fid(\mathrm{Vt}(\mathcal{P})) & ls_0(\mathcal{P}) &:\equiv ls(\mathrm{Lv}(\mathcal{P})) \end{aligned}$$

We will omit $\mathcal{P}$ when this will be clear from the context.

Theorem 3.25 There exists $k \in \mathbb{N}$ constant such that for any proof $\mathcal{P}$ in $\mathtt{EIL}^{\omega}_{+}$+AC+IP$_{\forall}$+MK and any non-realizer-free $A \in \mathrm{Lv}(\mathcal{P})$ there exists $\underline{t}_A$ such that $\underline{t}_A$ Dr A and the following hold:

- if A is not an instance of (CT∧, QA∀) then

$$\left.\begin{aligned} d(\underline{t}_A) &\leq k \\ S(\underline{t}_A) &\leq k \\ mdg(\underline{t}_A) &\leq k + vdg_0 + id_0 \\ mar(\underline{t}_A) &\leq k + var_0 + qs_0 \cdot id_0 \\ \mathtt{EIL}^{\omega}_{\mathtt{v}} &\vdash_k \{\!|\, \underline{t}_A\,,\, A \,|\!\} \end{aligned}\right\} \tag{38}$$

- if A is an instance of CT∧, (38) holds except that $\mathtt{EIL}^{\omega}_{\mathtt{v}} \vdash_{k\cdot ld_1} \{\!|\, \underline{t}_A\,,\, A \,|\!\}$, $d(\underline{t}_A) \leq k \cdot ld_1$ and $S(\underline{t}_A) \leq k \cdot ls_1$;
- if A is an instance of QA∀, (38) holds except that $d(\underline{t}_A) \leq k + fd_1$ and $S(\underline{t}_A) \leq k + fd_1$.

The $\tilde{\cdot}$ constants of $\mathtt{EIL}^{\omega}_{\mathtt{v}}$ above[28] are only those required by the terms $\underline{t}_A$ and hence are limited to those corresponding to terms occurring in A.

Proof: Follows immediately from Propositions 3.20, 3.22, 3.23 and $k_0 \geq 10$.

Theorem 3.26 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of a formula A in $\mathtt{EIL}^{\omega}_{+}$+AC+IP$_{\forall}$+MK it produces as output $\underline{t}$ such that $\underline{t}$ Dr A and, with the notations 3.24 and abbreviations[29] $\partial_{\mathtt{MP}} :\equiv \partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}} :\equiv \partial_{\mathtt{QR}}(\mathcal{P})$ and $\partial :\equiv \partial(\mathcal{P})$, the following hold:

[28] See also Definition 3.10 and Remark 3.11.

[29] See Footnote 26 for the meaning of $\partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}}(\mathcal{P})$ and $\partial(\mathcal{P})$.

$$
\begin{aligned}
d(\underline{t}) \;&\leq\; k \cdot ld_{1} + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot \partial_{\mathtt{MP}} && (39)\\
S(\underline{t}) \;&\leq\; (k \cdot ls_{1} + \partial_{\mathtt{QR}}) \cdot {qs_{\mathrm{o}}}^{\partial_{\mathtt{MP}}} && (40)\\
mdg(\underline{t}) \;&\leq\; k + vdg_{\mathrm{o}} + id_{\mathrm{o}} && (41)\\
mar(\underline{t}) \;&\leq\; k + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}} && (42)\\
\mathtt{EIL}^{\omega}_{\mathtt{V}} \;&\vdash_{k\cdot(ld_{1}+\partial)} \;\{\!|\, \underline{t}\,,\, A \,|\!\} && (43)
\end{aligned}
$$

The $\widetilde{\cdot}$ constants of $\mathtt{EIL}^{\omega}_{\mathtt{V}}$ in (43) are among those corresponding to terms occurring in the leaves of $\mathcal{P}$.

Proof: Just a synthesis of the results in Theorems 3.17 and 3.25. For (39) and (40) we use that $fd_{1} \leq qs_{\mathrm{o}}$ and $k_0 \geq 10$, hence

$$max\{k \cdot ld_{1}\,,\, k + fd_{1}\} + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot (\partial_{\mathtt{MP}} - k_0) \;\leq k \;\cdot ld_{1} + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot \partial_{\mathtt{MP}}$$

$$(max\{k \cdot ls_{1}\,,\, k + fd_{1}\} + qs_{\mathrm{o}} + \partial_{\mathtt{QR}} - k_0 + 1) \cdot {qs_{\mathrm{o}}}^{(\partial_{\mathtt{MP}} - k_{\mathrm{o}})} \;\leq\; (k \cdot ls_{1} + \partial_{\mathtt{QR}}) \cdot {qs_{\mathrm{o}}}^{\partial_{\mathtt{MP}}}$$

Notation 3.27 We will denote by

$$
\begin{aligned}
wd_{1}(\mathcal{P}) \;&:\equiv\; max\{wd(\mathtt{CT}\wedge \cap \mathrm{Lv}(\mathcal{P}))\,,\, td(\mathtt{QA} \cap \mathrm{Lv}(\mathcal{P}))\}\\
ws_{1}(\mathcal{P}) \;&:\equiv\; max\{ws(\mathtt{CT}\wedge \cap \mathrm{Lv}(\mathcal{P}))\,,\, ts(\mathtt{QA} \cap \mathrm{Lv}(\mathcal{P}))\}\\
cdg_{1}(\mathcal{P}) \;&:\equiv\; cdg((\mathtt{CT}\wedge \cup \mathtt{QA}) \cap \mathrm{Lv}(\mathcal{P}))\\
car_{1}(\mathcal{P}) \;&:\equiv\; car((\mathtt{CT}\wedge \cup \mathtt{QA}) \cap \mathrm{Lv}(\mathcal{P}))
\end{aligned}
$$

We will omit $\mathcal{P}$ when this will be clear from the context.

Remark 3.28 Theorem 3.26 holds also when the terms t_1, t_2 which build prime formulas $t_1 = t_2$ of contractions $\mathtt{CT}\wedge$ and the quantifier axioms terms $\underline{s}$ are counted as components of the global realizer (instead of just taking the associated constants $\widetilde{t_1}, \widetilde{t_2}, \widetilde{s}$). We only need to use wd_{1}, ws_{1} instead of ld_{1}, ls_{1} and (41), (42) must be replaced with

$$
\begin{aligned}
mdg(\underline{t}) \;&\leq\; max\{k + vdg_{\mathrm{o}} + id_{\mathrm{o}}\,,\, cdg_{1}\}\\
mar(\underline{t}) \;&\leq\; max\{k + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}}\,,\, car_{1}\}
\end{aligned}
$$

Corollary 3.29 There exists $k' \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of a formula $A \equiv \forall \underline{x}\, \exists \underline{y}\; B(\underline{x}, \underline{y})$ with $\{\underline{x}, \underline{y}\} = \mathcal{V}_{\mathrm{f}}(B)$ in $\mathtt{EIL}^{\omega}_{+}$+$\mathtt{AC}$+$\mathtt{IP}_{\forall}$+$\mathtt{MK}$ it produces as output $\underline{t_Y}$ such that

$$\mathtt{EIL}^{\omega}_{\mathtt{V}}\text{+}\mathtt{AC}\text{+}\mathtt{IP}_{\forall}\text{+}\mathtt{MK} \;\vdash_{k'\cdot max\{ld_{1}+\partial\,,\, ld(B)\}}\; \forall \underline{x}\; B(\underline{x}, \underline{t_Y}(\underline{x}))$$

Proof: In this case we have $A^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{U}\, \forall \underline{x}, \underline{v}\; B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v}; \underline{x}, \underline{Y}(\underline{x}))$, hence by Theorem 3.26 we get $\mathtt{EIL}^{\omega}_{\mathtt{V}} \vdash_{k\cdot(ld_{1}+\partial)} \forall \underline{v}\; B_{\mathtt{D}}(\underline{t_U}(\underline{x}); \underline{v}; \underline{x}, \underline{t_Y}(\underline{x}))$ and further

$$\mathtt{EIL}^{\omega}_{\mathtt{V}} \;\vdash_{k\cdot(ld_{1}+\partial)}\; \exists \underline{u}\, \forall \underline{v}\; B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{x}, \underline{t_Y}(\underline{x})) \quad [\,\equiv\, B^{\mathtt{D}}(\underline{x}, \underline{t_Y}(\underline{x}))\,] \tag{44}$$

It can be easily proved by induction on $ld(B)$ that there exists $k'' \in \mathbb{N}$ such that for all formulas B ,

$$\mathtt{EIL}^{\omega}+\mathtt{AC}+\mathtt{IP}_{\forall}+\mathtt{MK}\ \vdash_{k''\cdot\, ld(B)}\ B \leftrightarrow B^{\mathrm{D}} \tag{45}$$

The conclusion now follows immediately by combining (44) and (45) . $\square$

Remark 3.30 If λ-abstraction were treated as primitive and Σ, P, Π were defined in terms of it then (40) would still hold. Indeed, let Σ be defined by $\lambda x, \underline{y}, \underline{z}.\, x(\underline{z_0}, y_1(\underline{z}^1), \underline{z_1}, \ldots, y_m(\underline{z}^m), \underline{z_m})$. We would have $S(\Sigma) \leq 2 \cdot |x, \underline{y}, \underline{z}|^2$ and on the other hand $|x, \underline{y}, \underline{z}| \leq qs_o$ for all Σ which appear in $\underline{t}$. Similarly (40) would still hold if only Schönfinkel Σ and Π were allowed[30] . This follows from the λ-abstraction Definition 2.12 . There exists $k \in \mathbb{N}$ constant such that at most $k \cdot |x, \underline{y}, \underline{z}|^2 \leq k \cdot {qs_o}^2$ tuple-Schönfinkel Σ and Π are needed to simulate our Σ and any of these tuple-Schönfinkel Σ and Π can be defined[31] in terms of at most $k \cdot |x, \underline{y}, \underline{z}| \leq k \cdot qs_o$ usual Schönfinkel Σ and Π .

Remark 3.31 If we allowed only unary (see Remark 3.18) $\mathtt{ER_0}$ in the verifying proof then (43) would become $\mathtt{EIL}^{\omega}_{\mathtt{v}}\ \vdash_{k\cdot(ld_1+qs_o+\partial)}\ \{\!|\, \underline{t}\,,\, A \,|\!\}$.

Remark 3.32 The algorithm of Theorem 3.26 can be applied to complete proofs $\mathcal{P}$ in $\mathtt{EIL}^{\omega}_{+}+\mathtt{AC}+\mathtt{IP}_{\forall}+\mathtt{MK}$ after a preprocessing phase to $\mathcal{P}^{\mathrm{tr}}$ via the procedure of Definition 3.8 . Since $\mathtt{IEL}^{\omega} \vdash A \leftrightarrow A^{\mathrm{D}}$ for any realizer-free assumption A produced by the realizer-free-elimination procedure, the verifying proof can use the same assumptions as $\mathcal{P}^{\mathrm{tr}}$. A complete verifying proof in $\mathtt{EIL}^{\omega}_{\mathtt{v}}$ can be produced by (re)including the parts of $\mathcal{P}$ which were eliminated in the preprocessing phase.

3.4 Better bounds on the size of extracted terms

Smaller terms can be extracted if we use a simplification provided by the definitional equation of Σ . The size of the extracted terms becomes linear in the size of the proof at input. Nevertheless the use of extra Σ's brings an increase in type complexity. This can be avoided by using a more economical representation of the realizing tuples by means of pointers to parts which are shared by all members of a tuple. In such a setting all inequalities of Theorem 3.26 remain valid. The simplification is based on the observation that all terms t_4 produced by $\mathtt{MP}$ (see Lemma 3.14) contain a common part. Namely $\underline{t_1}, \underline{\mathtt{0}}$, which is somehow redundant to count for all t_4 in $\underline{t_4}$ - and this is what we have done so far. We give below a small example. Consider the following proof of C from A, $A \to B$ and $B \to C$: $\{\{A\,,\, A \to B\} \vdash B\,,\, B \to C\} \vdash C$. Let $\underline{t_1}$ Dr A,

[30] See Definition 2.4 for the notions of "tuple-Schönfinkel" and "Schönfinkel" combinators Σ . Also for "Schönfinkel" projectors Π .

[31] For Σ the proof is by induction on $|\underline{z}|$ of Definition 2.4 . We have $\Sigma\, x\, y\, z\, \underline{z'} = x\, z\, \underline{z'}\, (y\, z\, \underline{z'}) = \Sigma'\, (xz)\, (yz)\, \underline{z'}$ hence $\Sigma = \lambda x, y, z.\, \Sigma'\, (xz)\, (yz)$. For Π we can use the iterated λ-abstraction $\lambda z_1.\, (\ldots \lambda z_n.\, z_i)$.

$(\underline{t_2}, \underline{t_3})$ Dr $(A \to B)$ and $(\underline{t_5}, \underline{t_6})$ Dr $(B \to C)$. The algorithm in Lemma 3.14 first produces $\underline{t_4}$ Dr B defined as $t_4 \equiv \Sigma(t_3, \underline{t_1}, \underline{\mathtt{0}})$ and then produces the realizing tuple for C, namely $\underline{t_7}$ Dr C defined as

$$\begin{aligned} t_7 &\equiv \Sigma(t_6, \underline{t_4}, \underline{\mathtt{0}'}) \\ &\equiv \Sigma(t_6, \Sigma(t_3^1, \underline{t_1}, \underline{\mathtt{0}}), \ldots, \Sigma(t_3^{|\underline{t_3}|}, \underline{t_1}, \underline{\mathtt{0}}), \underline{\mathtt{0}'}) \end{aligned}$$

We immediately notice that the tuple $\underline{t_1}, \underline{\mathtt{0}}$ is common to all terms $t_4 \in \underline{t_4}$ and is multiply included in t_7. We describe below how it is possible to extract realizing terms such that the common parts which were previously multiply included are now counted only once for all the terms of a tuple.

Definition 3.33 For a proof $\mathcal{P}$ we define three *size* measures, denoted $S_i(\mathcal{P})$, $S_c(\mathcal{P})$ and $S_m(\mathcal{P})$, which are to be used in the semi-**i**ntuitionistic (i.e., what we have already described), the **c**lassical and in the **m**onotone case respectively (the last two cases will be treated in Section 4 below). The measure $S_m(\mathcal{P})$ will be used also for the time upper bounds (see Section 3.5) in all cases. All three size measures are obtained by adding the following to the sum of $qs(A \to B)$ for all MP-right-premises $A \to B$ plus the sum of $qs(C)$ for all QR-conclusions $qs(C)$ (below A are non-realizer-free leaves):

$S_i(\mathcal{P})$**:** the sum of $qs(A)$ for non-CT$\wedge$ A plus the sum of $ls(A)$ for CT$\wedge$ A;

$S_c(\mathcal{P})$**:** the sum of $ls(A)$ for all non-realizer-free leaves A;

$S_m(\mathcal{P})$**:** the sum of $qs(A)$ for all non-realizer-free leaves A.

It is immediate that $S_m(\mathcal{P}) \le S_i(\mathcal{P}) \le S_c(\mathcal{P})$, which reflects the fact that the monotone functional interpretation gives a simpler treatment of contraction than Gödel's functional interpretation and that the pre-processing negative translation brings an increase in complexity for Gödel's functional interpretation (but not for the monotone functional interpretation – see Theorem 4.20).

Definition 3.34 For the tuples $\underline{t} \equiv t_1, \ldots, t_n$ extracted by the algorithm of Theorem 3.26 we define a *size* measure, denoted $Sz'(\underline{t})$ in the following way. There exists $m \ge 0$ and a tuple $\underline{t'}$ such that each $t_i \in \underline{t}$ is either of shape $t_i \equiv P_1^i(\ldots P_m^i(t^i))$ or of shape $t_i \equiv P_1^i(\ldots P_m^i(t^i(\underline{t'})))$ where $\{P_j^i\}_{j=1}^m$ and t^i are *characteristic* to t_i and $\underline{t'}$ is *common* to all t_i in the corresponding subset of $\underline{t}$. It is possible that $m = 0$ and/or the aforementioned subset is $\emptyset$. We define

$$Sz'(\underline{t}) :\equiv m \cdot |\underline{t}| + \Sigma_{t' \in \underline{t'}} S(t') + \Sigma_{i=1}^n S(t^i) .$$

Lemma 3.35 There exists $k \in \mathbb{N}$ constant s.t. for every term $P_1(P_2\, x)$ with P_1 and P_2 permutations there exists a permutation P_3 s.t. $\vdash_k P_1(P_2\, x) = P_3\, x$.

Proof: By AxP for P_1 we obtain $(P_1(P_2\, x))(\underline{z^p}) = P_2(x, \underline{z})$. We can now apply Ax$P$ for P_2 and we distinguish two cases:

- $\underline{z} \equiv \underline{u^{p'}}, \underline{v}$ and $P_2(x, \underline{u^{p'}}) = x(\underline{u})$ hence $P_2(x, \underline{z}) = x(\underline{u}, \underline{v}) \equiv x(\underline{z^{p''}})$ and the last term is equal to $P_3(x, \underline{z^p})$ via a definitional equation for P_3.
- $\underline{z}, \underline{y} \equiv \underline{u^{p'}}$ and $P_2(x, \underline{u^{p'}}) = x(\underline{u})$ hence $(P_1(P_2\, x))(\underline{z^p}, \underline{y}) = x(\underline{u})$ and the last term is equal to $P_3(x, \underline{z^p}, \underline{y})$ via a definitional equation for P_3.

Lemma 3.36 There exists $k \in \mathbb{N}$ constant such that for any term $P_1(\ldots(P_m\,x))$ with $P_1,\ldots,P_m$ permutations there exists a permutation P_0 such that $\vdash_{k\cdot m} P_1(\ldots(P_m\,x)) = P_0\,x$.

Proof: Repeated applications of Lemma 3.35 and transitivity of equality. □

Theorem 3.37 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of a formula A in $\mathtt{EIL}^\omega_+ + \mathtt{AC} + \mathtt{IP}_\forall + \mathtt{MK}$ it produces as output $\underline{t}$ such that $\underline{t}$ Dr A with (43) and (below $\#_{\mathtt{MP}}$ denotes the number of $\mathtt{MP}$ instances in $\mathcal{P}$)

$$Sz'(\underline{t}) \leq k \cdot S_i(\mathcal{P}) \tag{46}$$

$$3\,\#_{\mathtt{MP}} \leq d(\underline{t}) \tag{47}$$

$$\partial_{\mathtt{MP}} - k_0 \leq mdg(\underline{t}) \tag{48}$$

$$3\,\#_{\mathtt{MP}} \leq mar(\underline{t}) \tag{49}$$

Proof: The proof of (46) is by structural induction on $\mathcal{P}$. For axioms A we use the same realizing terms as before. When A is not an instance of $\mathtt{CT}\wedge$ or $\mathtt{QA}\forall$, (46) follows from $|\underline{t}| \leq qs(A)$. If $A \equiv B \to B \wedge B$ then we notice that $t_B^{\mathtt{D}}$ of Proposition 3.22 is common to all realizing t_Y, hence using (9) and (28),

$$Sz'(\underline{t_{X'}}, \underline{t_{X''}}, \underline{t_Y}) \leq k' \cdot |\underline{Y}, \underline{X'}, \underline{X''}| + S(t_B^{\mathtt{D}}) \leq k' \cdot qs(A) + k'' \cdot ls(A) \leq k \cdot ls(A)$$

If $A \equiv \forall \underline{z} B(\underline{z}, \underline{a''}) \to B(\underline{s}, \underline{a''})$ then the tuple $\underline{\widetilde{s}}$ of Proposition 3.23 is common to all realizing t_X, hence $Sz'(\underline{t_Z}, \underline{t_Y}, \underline{t_X}) \leq k' \cdot |\underline{Z}, \underline{Y}, \underline{X}| + |\underline{\widetilde{s}}| \leq k \cdot qs(A)$. There is nothing to prove for instances of $\mathtt{EXP}$ and $\mathtt{IMP}$, see Lemma 3.16. For $\mathtt{QR}$ instances the proof is trivial using Lemma 3.15. For instances of $\mathtt{MP}$ we use Lemma 3.14 and further improve the result by applying a number of Σ definitional equations. The algorithm in Lemma 3.14 is presented with the tuples $\underline{t_3}$ and $\underline{t_1}$, represented [32] as

$$\left.\begin{aligned} t_3 &\equiv P'_1(\ldots P'_{m'}(t_0(\underline{t'}))) = P'(t_0(\underline{t'})) \\ t_1^i &\equiv P_1^i(\ldots P_m^i(t^i(\underline{t}))) \;\; = P_i(t^i(\underline{t})) \end{aligned}\right\} \text{ Using Lemma 3.36}$$

and it produces (we assumed without loss of generality that $1 \leq m', m$)

$$\begin{aligned} t_4 &\equiv \Sigma_1(P'(t_0(\underline{t'})), P_1(t^1(\underline{t})), \ldots, P_n(t^n(\underline{t})), \underline{0}) = \\ &= \Sigma_2(\Sigma_1, P', P_1, \ldots, P_n, t_0(\underline{t'}), t^1(\underline{t}), \ldots, t^n(\underline{t}), \underline{0}) = \\ &= \Sigma_3(\Sigma_2, t_0, t^1, \ldots, t^n, \underline{t'}, \underline{t}, P', P_1, \ldots, P_n, \Sigma_1, \underline{0}) = \\ &= P(\Sigma_3, \Sigma_2, P', t_0, t^1, \ldots, t^n, \underline{t'}, \underline{t}, P_1, \ldots, P_n, \Sigma_1, \underline{0}) \end{aligned}$$

[32] In the case when $\underline{t_3}$ or $\underline{t_1}$ comes from a (sub)proof which involved $\mathtt{CT}\wedge$ or $\mathtt{QA}\forall$ and no $\mathtt{MP}$ then we have an exception in the sense that only a part of the terms in the tuple share a common tuple, see also Definition 3.34. The reason should be obvious from the above treatment of $\mathtt{CT}\wedge$ and $\mathtt{QA}\forall$. The final shape of the term t_4 in (50) below is nevertheless not affected by this technical exception. After an $\mathtt{MP}$ all terms of the realizing tuple share a common tuple.

hence we can actually take

$$t_4 \equiv (P\,\Sigma_3\,\Sigma_2\,P'\,t_0)(t^1,\dots,t^n,\underline{t'},\underline{t},P_1,\dots,P_n,\Sigma_1,\underline{\mathtt{0}}) \tag{50}$$

where $t^1,\dots,t^n,\underline{t'},\underline{t},P_1,\dots,P_n,\Sigma_1,\underline{\mathtt{0}}$ is common to all $t_4 \in \underline{t_4}$. Hence

$$\begin{aligned} Sz'(\underline{t_4}) &\le |P,\Sigma_3,\Sigma_2| \cdot |\underline{t_4}| + |\Sigma_1,\underline{\mathtt{0}}| + Sz'(\underline{t_3}) + Sz'(\underline{t_1}) \\ &\le 3 \cdot qs(A \to B) + Sz'(\underline{t_3}) + Sz'(\underline{t_1})\,, \end{aligned}$$

where for the last inequality we used that $|\underline{t_4}| + max\{1\,,\,|\underline{\mathtt{0}}|\} \le qs(A \to B)$.

In order to prove the remaining inequalities it is useful to denote by $\mathtt{cp}(\underline{t})$ the common tuple in the canonical representation of the tuple $\underline{t}$ (i.e., $\underline{t'}$) . We have $|\mathtt{cp}(\underline{t_1})| + |\mathtt{cp}(\underline{t_3})| + 1 \le |\mathtt{cp}(\underline{t_4})|$ because at least the constant Σ_1 appears new at each MP application. It follows that for the final extracted tuple $\underline{t}$ we have $\#_{\mathtt{MP}} \le |\mathtt{cp}(\underline{t})|$. Now (47) and (49) are immediate because $|\mathtt{cp}(\underline{t})| \le d(\underline{t})$ and $|\mathtt{cp}(\underline{t})| \le mar(\underline{t})$. Also (48) is immediate once we notice that $dg(\mathtt{cp}(\underline{t}))$ increases by at least 1 at each MP application; this is due to the fact that t^i enters $\mathtt{cp}(\underline{t_4})$ and $dg(t^i) \ge dg(\underline{t}) + 1$.

The proof that (43) still holds is by straightforward computations. □

We notice that the price to pay for having smaller realizing terms is an increase in type complexity. This is unavoidable with the actual representation of terms. The maximal degree of the realizing term increases by at least 1 at each MP application. This is due to the fact that subterms from the private part, which have degree greater by at least 1 than the maximal degree of a subterm from the common part now enter the new common part. We can avoid the increase in type complexity only by modifying the term representation such that the terms in the common part are multiply pointed from each member of the realizing tuple. In this way Σ_3 is no longer needed for feeding the common part to each member of the realizing tuple. The increase in degree was due exactly to these Σ_3's. We can now state the following theorem, where Sz is defined in the new pointer setting similarly to Sz', i.e., by counting common parts only once.

Theorem 3.38 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of a formula A in $\mathtt{EIL}^\omega_+ + \mathtt{AC} + \mathtt{IP}_\forall + \mathtt{MK}$ it produces as output $\underline{t}$ such that $\underline{t}$ Dr A , $Sz(\underline{t}) \le k \cdot S_i(\mathcal{P})$ and the inequalities of Theorem 3.26 all hold.

Remark 3.39 The following inequalities are immediate:

$$\begin{aligned} S(t) &\le Sz(\underline{t}) \\ S_i(\mathcal{P}) &\le (ls_1(\mathcal{P}) + qs_0(\mathcal{P})) \cdot 2^{\partial(\mathcal{P})} \;\le\; 3 \cdot ls_1(\mathcal{P}) \cdot qs_0(\mathcal{P})^{\partial(\mathcal{P})}\;. \end{aligned}$$

They just express the fact that the new bounds on size are indeed better.

Remark 3.40 We will tacitly assume in the sequel that terms are represented with pointers in the manner described above.

3.5 Space and time complexity of the functional interpretation algorithm

In a real-world implementation of the algorithm of Theorem 3.26 we ought to count also the size of types associated to the $\mathtt{EIL}^\omega$-constants as part of the size of the realizing terms. This *real-size* of the extracted terms actually gives also the time complexity of the algorithm[33] since what this does is only writing down the extracted terms.

In order to compute the real-size we need to decide upon some representation of types. It turns out that the most efficient is to use *dags* [34]. We choose dags instead of normal binary trees[35] because dags allow the reuse of existent types via pointers. Hence given the input proof $\mathcal{P}$ we start with the types of all variables and constants which appear in $\mathcal{P}$ and build the types of constants which are produced by functional interpretation. We therefore need to count for the real-size only the number of new type-nodes which are created in order to represent the type of a newly created constant c. By straightforward computations it follows that there exists $k' \in \mathbb{N}$ constant such that for any formula C, the number of new type-nodes required by $\mathcal{V}_b(C^{\mathtt{D}})$ is at most $k' \cdot qs(C)^2 \cdot ls(C)$. Hence the number of new type-nodes required by the new variables created in the interpretation of the leaves, right $\mathtt{MP}$–premises and $\mathtt{QR}$–conclusions of $\mathcal{P}$ is at most $k' \cdot qs_o \cdot ls_o \cdot S_m(\mathcal{P})$. Then we can immediately see that whenever a new constant c of type $\underline{\sigma}\tau$ is created by the algorithm of Theorem 3.26, the types $\underline{\sigma}, \tau$ are immediately available from the existent terms or variables created by the functional interpretation. There exists $k'' \in \mathbb{N}$ constant such that for any such new constant c created at a leaf C, instance of $\mathtt{MP}$–right–premise C or instance of $\mathtt{QR}$–conclusion C of $\mathcal{P}$, at most $k'' \cdot |\underline{\sigma}|^2 \leq k'' \cdot qs(C)^2$ new type-nodes are necessary to represent the type of c. Hence overall we have at most $k'' \cdot qs_o \cdot ls_o \cdot S_m(\mathcal{P})$ newly created type-nodes in this category. We can now state the following theorem.

Theorem 3.41 There exists $k \in \mathbb{N}$ constant such that the time overhead of the algorithm in Theorem 3.38 is upper bounded by $k \cdot qs_o \cdot ls_o \cdot S_m(\mathcal{P})$.

[33] The space complexity follows immediately by the principle that the space overhead of an algorithm is always less than its time overhead.
[34] Here "dag" is the usual acronym for "directed acyclic graph".
[35] The representation with binary trees is in fact equivalent to the usual parenthesized-strings representation.

4 Immediate extensions of the quantitative analysis

4.1 Treatment of classical $\mathtt{EIL}^\omega$*. The system* $\mathtt{ECL}^\omega_+ + \mathtt{AC}_0$

So far we have considered only semi-intuitionistic systems. We describe in the sequel how our complexity analysis can easily be adapted to classical logic (and theories) as well by applying it to the image of the classical system under a suitable negative translation. The so-called *negative* or *double–negation* translations have all in common the fact that the image of a formula is (intuitionistically equivalent to) a negative formula [36]. Negative translations were initially produced by Gödel [21], Gentzen, Kolmogorov, Glivenko. We use below a variant due to Kuroda of Gödel's translation which we further adapt in order to handle blocks of universal quantifiers.

Definition 4.1 (Kuroda's N-translation) To a formula A one associates $A^{\mathtt{N}} \equiv \neg\neg A^*$, where A^* is defined by structural induction on A as follows:

- $A^* \;:\equiv\; A$, if A is a prime formula
- $(A \Box B)^* \;:\equiv\; A^* \Box B^*$, where $\Box \in \{\wedge, \vee, \to\}$
- $(\exists x A(x))^* \;:\equiv\; \exists x (A(x))^*$
- $(\forall \underline{x} A(\underline{x}))^* \;:\equiv\; \forall \underline{x} \neg\neg (A(\underline{x}))^*$, where $A(\underline{x}) \not\equiv \forall y B(y, \underline{x})$

Remark 4.2 $A^{\mathtt{N}}$ is realizer-free iff A is realizer-free.

N-translation followed by functional interpretation gives a proof interpretation for theories based on classical logic [37]. Remark 4.2 implies that given a (complete) proof $\mathcal{P}$ in some classical system the following are equivalent:

- carry out the composed [38] interpretation to $\mathcal{P}^{\mathrm{tr}}$;
- first do the N-translation of $\mathcal{P}$, then apply the realizer-free-elimination algorithm of Definition 3.8 and finally carry out the functional interpretation of $(\mathcal{P}^{\mathtt{N}})^{\mathrm{tr}}$.

The former approach is obviously more efficient: one does not carry out the N-translation of parts which subsequently get eliminated.

[36] By definition, a formula is called *negative*, respectively *existential-free* if it is built up from negated prime, respectively prime formulas by means of $\bot$, $\wedge$, $\to$ and $\forall$ only. In our system negative formulas are trivially existential-free. On the other hand, $\mathtt{EIL}^\omega \vdash s =_o t \leftrightarrow \neg\neg(s =_o t)$ for any prime formula $s =_o t$ and hence also every existential-free formula is equivalent to a negative formula.
[37] Details of the use of negative translation in combination with functional interpretation may be found, e.g., in [3,29,39].
[38] In fact parts which are produced by N-translation also need to be transformed.

Let $\mathtt{ECL}^{\omega}, \mathtt{ECL}^{\omega}_{+}$ be the classical versions[39] of $\mathtt{EIL}^{\omega}, \mathtt{EIL}^{\omega}_{+}$ respectively, obtained by replacing $\mathtt{TND}_0$ with the full tertium-non-datur schema $A \vee \neg A$. Let

$$\mathtt{AC}_0 : \quad \forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x}, \underline{y}) \;\rightarrow\; \exists \underline{Y}\, \forall \underline{x}\, A_0(\underline{x}, \underline{Y}(\underline{x}))$$

be the quantifier-free axiom-of-choice (with $\underline{x}$ and $\underline{y}$ of arbitrary types).

Remark 4.3 The proof–size measure S_c is introduced in Definition 3.33.

Proposition 4.4 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of a formula A in $\mathtt{ECL}^{\omega}_{+}+\mathtt{AC}_0$ it produces as output a proof $\mathcal{P}^{\mathtt{N}}$ of $A^{\mathtt{N}}$ in $\mathtt{EIL}^{\omega}_{+}+\mathtt{AC}_0+\mathtt{MK}$ and the following hold:

(1) $\partial(\mathcal{P}^{\mathtt{N}}) \;\leq\; k \cdot \partial(\mathcal{P})$ and $S_i(\mathcal{P}^{\mathtt{N}}) \;\leq\; k \cdot S_c(\mathcal{P})$;

(2) $$\begin{cases} qs_0(\mathcal{P}^{\mathtt{N}}) \;\leq\; qs(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;\leq\; k \cdot qs(\mathrm{Vt}(\mathcal{P})) \overset{(13)}{=} k \cdot qs_0(\mathcal{P}) \\ ld_1(\mathcal{P}^{\mathtt{N}}) \;\leq\; ls_1(\mathcal{P}^{\mathtt{N}}) \;\leq\; ls(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;\leq\; k \cdot ls(\mathrm{Vt}(\mathcal{P})) \overset{(13)}{=} k \cdot ls_0(\mathcal{P}) \end{cases}$$

(3) $id_0(\mathcal{P}^{\mathtt{N}}) \leq k \cdot fid_0(\mathcal{P})$; we must use $fid_0(\mathcal{P})$ because in the N-translation a $\forall$ brings two $\neg$, hence in fact two $\rightarrow$ due to our treatment of negation;

(4) no new variable or constant appears in $\mathcal{P}^{\mathtt{N}}$, hence (using (14))

$$\begin{aligned} vdg(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;&\leq\; vdg(\mathrm{Vt}(\mathcal{P})) \;=\; vdg(\mathrm{Lv}(\mathcal{P})) \\ var(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;&\leq\; var(\mathrm{Vt}(\mathcal{P})) \;=\; var(\mathrm{Lv}(\mathcal{P})) \\ cdg(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;&\leq\; cdg(\mathrm{Vt}(\mathcal{P})) \;=\; cdg(\mathrm{Lv}(\mathcal{P})) \\ car(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;&\leq\; car(\mathrm{Vt}(\mathcal{P})) \;=\; car(\mathrm{Lv}(\mathcal{P})) \end{aligned}$$

Proof: The algorithm proceeds by recursion on the structure of $\mathcal{P}$, see [29] for details. The proof of its correctness makes use of the following schemata of intuitionistic logic:

$$\neg\neg(A \rightarrow B) \leftrightarrow (A \rightarrow \neg\neg B) \leftrightarrow (\neg\neg A \rightarrow \neg\neg B) \tag{51}$$

$$\neg\neg\forall \underline{x} \neg\neg A(\underline{x}) \leftrightarrow \forall \underline{x} \neg\neg A(\underline{x}) \tag{52}$$

$$A \rightarrow \neg\neg A \tag{53}$$

These schemata have proofs in which the axiom instances and intermediate formulas have size (depth) at most linear in the size (depth) of the formula to be proved. We only need to further notice that there exists $k' \in \mathbb{N}$ constant such that the following hold:

- the N-translation of any non-realizer-free axiom scheme B of $\mathtt{ECL}^{\omega}_{+}+\mathtt{AC}_0$ is a theorem in $\mathtt{EIL}^{\omega}_{+}+\mathtt{AC}_0+\mathtt{MK}$ whose proof $\mathcal{P}'$ has the same structure for all instances of B, in particular the same depth; all formulas which appear in $\mathcal{P}'$ have size (depth) upper bounded by k' times the maximal size (depth) of B;

[39] Below $\mathtt{EIL}^{\omega}_{+}$-based systems will appear for verifying the functional interpretation of proofs in $\mathtt{ECL}^{\omega}_{+}$-based systems. In virtue of Remark 4.2 it should be obvious that A is a realizer-free axiom from $\mathbb{Th}_{\mathtt{rf}}$ of $\mathtt{ECL}^{\omega}_{+}$ (see Definition 3.10) if and only if $A^{\mathtt{N}}$ is a realizer-free axiom from $\mathbb{Th}_{\mathtt{rf}}$ of $\mathtt{EIL}^{\omega}_{+}$.

- any rule $A_1\,[,\,A_2] \vdash B$ of $\mathtt{ECL}^{\omega}_{+}{+}\mathtt{AC}_0$ is interpreted under N-translation to a proof $\mathcal{P}'$ of $B^{\mathtt{N}}$ from $A_1^{\mathtt{N}}\,[,\,A_2^{\mathtt{N}}]$; $\mathcal{P}'$ has the same structure for all instances of the rule, in particular the same depth; all formulas which appear in $\mathcal{P}'$ have size (depth) upper bounded by k' times the maximal size (depth) of $A_1\,[,\,A_2]$.

As an example we prove the above claim for $\mathtt{AC}_0$ and $\mathtt{QR}\forall$. The other axioms and rules are even easier.

Case $\mathtt{AC}_0$: We prove that there exists $k' \in \mathbb{N}$ constant such that for all A_0,

$$\mathtt{EIL}^{\omega}_{+}{+}\mathtt{AC}_0{+}\mathtt{MK} \vdash_{k'} [\forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x},\underline{y}) \rightarrow \exists \underline{Y}\, \forall \underline{x}\, A_0(\underline{x},\underline{Y}(\underline{x}))\,]^{\mathtt{N}} \tag{54}$$

By (53), the conclusion of (54) is implied by

$$\forall \underline{x}\, \neg\neg\, \exists \underline{y}\, A_0(\underline{x},\underline{y}) \;\rightarrow\; \exists \underline{Y}\, \forall \underline{x}\, \neg\neg\, A_0(\underline{x},\underline{Y}(\underline{x}))\;.$$

This follows from $\mathtt{MK}$ and $\mathtt{AC}_0$ with a $\mathtt{IEL}^{\omega}$–proof of constant depth.

Case $\mathtt{QR}\forall$: $B \rightarrow A \vdash B \rightarrow \forall \underline{z} A$. By induction hypothesis we have a proof of $\neg\neg(B^* \rightarrow A^*)$. Then we use (51) and $\mathtt{MP}$ to get $B^* \rightarrow \neg\neg A^*$ and by $\mathtt{QR}\forall$, $B^* \rightarrow \forall \underline{z} \neg\neg A^*$. If $A \not\equiv \forall y C$ then $\forall \underline{z} \neg\neg A^* \equiv (\forall \underline{z} A)^*$. If $A \equiv \forall \underline{x} A'$ with $A' \not\equiv \forall y C$ then $A^* \equiv \forall \underline{x} \neg\neg A'^*$ and using (52) we obtain $B^* \rightarrow \forall \underline{z},\underline{x}\, \neg\neg A'^*$ with $\forall \underline{z},\underline{x}\, \neg\neg A'^* \equiv (\forall \underline{z} A)^*$. In any case we obtain $\neg\neg(B \rightarrow \forall \underline{z} A)^*$ (also using (53)). Hence overall the deduction of $(B \rightarrow \forall \underline{z} A)^{\mathtt{N}}$ from $(B \rightarrow A)^{\mathtt{N}}$ has constant depth. $\square$

Remark 4.5 The new quantifier axioms of $\mathcal{P}^{\mathtt{N}}$ are of shape $\forall \underline{z} B(\underline{z}) \rightarrow B(\underline{z})$ and these can be realized with simple projectors $\mathit{\Pi}$ instead of the terms t_Z of Proposition 3.23.

Remark 4.6 Except for those triggered by $(A \rightarrow A \wedge A)^{\mathtt{N}}$, the contractions $\mathtt{CT}\wedge$ of $\mathcal{P}^{\mathtt{N}}$ are required by the N-translations of $A \vee \neg A$, $\mathtt{QA}\forall$ and $\mathtt{QR}\exists$. In the last two cases the verifying $\mathtt{CT}\wedge$ is brought by the critical implication

$$(\neg\neg A \;\rightarrow\; \neg\neg B) \;\rightarrow\; \neg\neg(A \;\rightarrow\; B) \tag{55}$$

of (51). The use of (55) can be avoided in the case of $\mathtt{IMP}$, $\mathtt{EXP}$ by using axiom versions of these rules[40], the non-critical converse of (55) and $\mathtt{MP}$.

Remark 4.7 The following holds: $\;((\mathcal{P}^{\mathrm{tr}})^{\mathtt{N}})^{\mathrm{tr}} \;=\; (\mathcal{P}^{\mathtt{N}})^{\mathrm{tr}}$.

We are now able to describe an efficient algorithm for extracting realizing terms from (complete) proofs $\mathcal{P}$ in $\mathtt{ECL}^{\omega}_{+}{+}\mathtt{AC}_0$. First $\mathcal{P}$ is transformed to $\mathcal{P}^{\mathrm{tr}}$ and then to $(\mathcal{P}^{\mathrm{tr}})^{\mathtt{N}}$ via the algorithm of Proposition 4.4. In a second phase $(\mathcal{P}^{\mathrm{tr}})^{\mathtt{N}}$ is transformed[41] to $((\mathcal{P}^{\mathrm{tr}})^{\mathtt{N}})^{\mathrm{tr}}$ and the algorithm of Theorem 3.26 is applied to it. Using Proposition 4.4, Theorems 3.38 and 3.41, Notation 3.24

[40] The axiom versions of $\mathtt{IMP}$ and $\mathtt{EXP}$ are simply realized with projectors $\mathit{\Pi}$. This follows immediately from the fact that $(\,A \rightarrow (B \rightarrow C)\,)^{\mathtt{D}} \equiv (\,A \wedge B \rightarrow C\,)^{\mathtt{D}}$. See also Lemma 3.16.

[41] Here only the parts produced by N-translation need to be transformed.

and the abbreviations $\partial :\equiv \partial(\mathcal{P})$, $S_c :\equiv S_c(\mathcal{P})$ and $S_m :\equiv S_m(\mathcal{P})$ we can state the following theorem.

Theorem 4.8 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of a formula A in $\mathtt{ECL}^{\omega}_{+}+\mathtt{AC}_0$ it produces as output $\underline{t}$ such that $\underline{t}$ Dr $A^{\mathtt{N}}$ and the following hold:

$$d(\underline{t}) \ \leq\ k \cdot (ls_{\mathrm{o}} + qs_{\mathrm{o}} \cdot \partial) \tag{56}$$

$$S(\underline{t}) \ \leq\ Sz(\underline{t}) \ \leq\ k \cdot S_c \ \leq\ k \cdot (ls_{\mathrm{o}} + \partial) \cdot (k \cdot qs_{\mathrm{o}})^{k \cdot \partial} \tag{57}$$

$$mdg(\underline{t}) \ \leq\ vdg_{\mathrm{o}} + k \cdot fid_{\mathrm{o}} \tag{58}$$

$$mar(\underline{t}) \ \leq\ var_{\mathrm{o}} + k \cdot qs_{\mathrm{o}} \cdot fid_{\mathrm{o}} \tag{59}$$

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \ \vdash_{k\cdot(ls_{\mathrm{o}}+\partial)} \ \{\!\!\{\, \underline{t}\,,\, A^{\mathtt{N}} \,\}\!\!\}\,. \tag{60}$$

The time overhead of the algorithm is upper bounded by $k \cdot qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m$. The $\tilde{\cdot}$ constants of $\mathtt{EIL}^{\omega}_{\mathtt{v}}$ in (60) are among those corresponding to terms occurring in the leaves of $\mathcal{P}^{\mathtt{N}}$.

Remark 4.9 In the above theorem we use the more general quantity ∂ instead of the more detailed ones $\partial_{\mathtt{QR}}$ and $\partial_{\mathtt{MP}}$ which appear in Theorem 3.26 . We do so because the N-translations of $\mathtt{QR}\forall$, $\mathtt{QR}\exists$, $\mathtt{EXP}$ and $\mathtt{IMP}$ trigger new $\mathtt{MP}$ instances needed for their verification in $\mathcal{P}^{\mathtt{N}}$. Hence $\partial_{\mathtt{MP}}(\mathcal{P}^{\mathtt{N}}) \geq \partial(\mathcal{P})$ already.

Corollary 4.10 Let $A \equiv \forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x}, \underline{y})$ with $\mathcal{V}_{\mathrm{f}}(A_0) = \{\underline{x}, \underline{y}\}$ and, as usual, A_0 quantifier-free. The theorem above holds also with A instead of $A^{\mathtt{N}}$, i.e., $\underline{t}$ Dr A with (56), (57), (58), (59) and $\mathtt{EIL}^{\omega}_{\mathtt{v}} \ \vdash_{k\cdot(ls_{\mathrm{o}}+\partial)} \ \forall \underline{x}\, A_0(\underline{x}, \underline{t}(\underline{x}))$.

Proof: There exists $k' \in \mathbb{N}$ constant such that, using (52) and (5),

$$\mathtt{EIL}^{\omega}_{+}+\mathtt{MK} \ \vdash_{k'\cdot\, ld(A_0)} \ (\forall \underline{x} \exists \underline{y} A_0(\underline{x}, \underline{y}))^{\mathtt{N}} \to \forall \underline{x} \exists \underline{y} A_0(\underline{x}, \underline{y}) \ .$$

From (13) it follows that the quantity $ld(A_0)$ gets absorbed into ls_{o} . $\square$

4.2 A quantitative analysis of monotone functional interpretation

The second author realized in [31] that a much simpler extraction procedure applies if the goal is to extract majorizing functionals $\underline{t^*}$ for the realizing terms $\underline{t}$ of $A^{\mathtt{D}}$, i.e., terms $\underline{t^*}$ such that

$$\mathtt{M}: \qquad \exists \underline{x}\ [\, \underline{t^*}\ maj\ \underline{x} \ \wedge\ \forall \underline{a}, \underline{y}\, A_{\mathtt{D}}(\underline{x}(\underline{a}), \underline{y}, \underline{a})] \qquad .$$

Here $\underline{y}\ maj\ \underline{x}\ :\equiv\ \wedge\, (y\ maj\ x)$ and maj is W.A. Howard's majorization relation (see [27]). This is of significance since $\underline{t^*}$ suffices for many (if not most) applications of functional interpretation. These range from conservation results (e.g., for weak König's lemma [30]) to the proof mining of concrete proofs [35]. We noticed in Section 3 that the contraction $A \to A \wedge A$ is by far the most complicated axiom in the usual functional interpretation. Monotone functional interpretation features a very simple treatment of $A \to A \wedge A$ and therefore the extraction process for $\underline{t^*}$ becomes much simpler than the one for $\underline{t}$.

Definition 4.11 Let $\mathtt{EIL}^\omega_\mathtt{M}$ be an extension of $\mathtt{EIL}^\omega$ with the following:

- An inequality relation $\geq_o$ for type-o-objects with the usual axioms plus $1 \geq_o \mathtt{I}\, x^o\, y^o$, $1 \geq_o \nu\, x^o\, y^o$ and $1 \geq_o \mathtt{E}\, x^o\, y^o$. Inequality for higher types is defined extensionally by $x \geq_{\underline{\sigma} o} y :\equiv \forall \underline{z}^{\underline{\sigma}} (x\, \underline{z} \geq_o x\, \underline{z})$. The *majorization* relation is defined by $x^*\ maj_{\underline{\sigma} o}\ x :\equiv \forall \underline{z}^{\underline{\sigma}}, \underline{y}^{\underline{\sigma}}\ (\underline{z}\ maj_{\underline{\sigma}}\ \underline{y} \ \rightarrow\ x^*\, \underline{z}\ maj_o\ x\, \underline{y})$, where $\underline{z}\ maj_{\underline{\sigma}}\ \underline{y}$ is an abbreviation for $\wedge_{\sigma \in \underline{\sigma}} (z\ maj_\sigma\ y)$ and $maj_o :\equiv\ \geq_o$.
- A *maximum* constant $\mathcal{M}_o$ of type ooo defined by the axioms
  $\mathtt{Ax}\mathcal{M}:\qquad \mathcal{M}_o\, x\, y \geq_o x \qquad \mathcal{M}_o\, x\, y \geq_o y \qquad \mathcal{M}_o\ maj\ \mathcal{M}_o$.
  Maximum constants for higher types are defined by
  $$\mathcal{M}_{\underline{\sigma} o} :\equiv \mathit{\Sigma}\, \mathcal{M}_o = \lambda x^{\underline{\sigma} o}, y^{\underline{\sigma} o}, \underline{z}^{\underline{\sigma}}.\, \mathcal{M}_o\, (x\, \underline{z})\, (y\, \underline{z}) \ .$$
- A schema of explicit definability for arbitrary quantifier-free formulas:
  $\mathtt{ED}[A_0]:\quad \exists Y\, \forall \underline{a}\ [\,(1 \geq_o Y(\underline{a})) \wedge (A_0(\underline{a}) \leftrightarrow Y(\underline{a}) =_o 0)\,]$.
- Axioms $\mathtt{S}\ maj\ \mathtt{S}$ and $\mathtt{0}\ maj\ \mathtt{0}$.

Remark 4.12 In the presence of a minimal amount of arithmetic $\mathtt{S}\ maj\ \mathtt{S}$ and $\mathtt{0}\ maj\ \mathtt{0}$ are immediately provable. Also the constants $\geq_o$, ν, $\mathtt{I}$, $\mathtt{E}$ and $\mathcal{M}_o$ can be defined such that the remaining axioms of Definition 4.11 become provable (see also Remark 2.5).

Remark 4.13 The formulas $\mathit{\Sigma}\ maj\ \mathit{\Sigma}$, $\mathit{\Pi}\ maj\ \mathit{\Pi}$ and $P\ maj\ P$ hold in $\mathtt{EIL}^\omega_\mathtt{M}$ with proofs of depths proportional with the arities of $\mathit{\Sigma}$, $\mathit{\Pi}$ and P respectively. Then $\mathcal{M}_\rho\ maj\ \mathcal{M}_\rho$ holds for arbitrary ρ with a formal proof of depth proportional with $ar(\rho) + 1$.

Lemma 4.14 There exists $k \in \mathbb{N}$ constant such that for any tuple of terms $\underline{s}$ of $\mathtt{EIL}^\omega_\mathtt{M}$ (with $\mathcal{V}(\underline{s}) = \{\underline{x}\}$) there exist corresponding terms $\underline{s^*}$ of $\mathtt{EIL}^\omega_\mathtt{M}$ (with $\mathcal{V}(\underline{s^*}) = \{\underline{x^*}\}$) such that
$$\mathtt{EIL}^\omega_\mathtt{M}\ \vdash\ \underline{x^*}\ maj\ \underline{x}\ \rightarrow\ \underline{s^*}\ maj\ \underline{s}\ . \tag{61}$$

Proof: The constants $\mathtt{0}$ and $\mathtt{S}$ trivially majorize themselves by the last clause of Definition 4.11. On the other hand, $\mathit{\Sigma}\mathcal{M} = \lambda z, \underline{x}, \underline{x'}.\, \mathcal{M}\, x\, x'$ majorizes $\mathcal{D}$ and $\mathit{\Pi} 1 = \lambda x^o, y^o.\, 1$ majorizes $\mathtt{I}$, ν and $\mathtt{E}$. Using Remark 4.13 we have that $\mathit{\Sigma}$, $\mathit{\Pi}$, P and $\mathcal{M}$ majorize themselves. The conclusion follows immediately by induction on $d(\underline{s})$. $\square$

Corollary 4.15 Let $\widetilde{s}, \widetilde{s^*}$ be constants associated to terms s, s^* like in Definition 3.10. From (61) it immediately follows that
$$\vdash\ \underline{\widetilde{s^*}}\ maj\ \underline{\widetilde{s}} \tag{62}$$

Definition 4.16 We denote by $\mathtt{EIL}^\omega_{\mathtt{M},+}$ the system $(\mathtt{EIL}^\omega_\mathtt{M})_+$ where "+" includes all formulas $\underline{\widetilde{s^*}}\ maj\ \underline{\widetilde{s}}$ as axioms. We take them as axioms because we consider that the (formal) proof in (62) is not created by monotone functional interpretation. Also let $\mathtt{EIL}^{\omega}_{\mathtt{M,v}}$ be the corresponding $(\mathtt{EIL}^{\omega}_{\mathtt{M}})_{\mathtt{v}}$.

In [31] realizing terms are presented for the monotone functional interpretation of all axioms of $\mathtt{EIL}^{\omega}_{\mathtt{M}}+\mathtt{AC}+\mathtt{IP}_{\forall}+\mathtt{MK}$. They are the same as for the usual functional interpretation, except that

- $A \to A \wedge A$ is realized by terms $\Sigma\mathcal{M} = \lambda \underline{a}, \underline{x}, \underline{y'}, \underline{y''}. \mathcal{M}\, y'\, y''$ and $\Pi = \lambda \underline{a}, \underline{x}. x$; compare this with the results of Proposition 3.22 ;
- $A \vee A \to A$ is realized by $\Sigma\mathcal{M} = \lambda \underline{a}, z, \underline{x}, \underline{x'}. \mathcal{M}\, x\, x'$ and Π ;
- $A \vee B \to B \vee A$ is realized by terms Π and $\Pi\, 1$;
- the schema ED itself is trivially realized by $\Pi\, 1 = \lambda \underline{a}. 1$;
- $\forall \underline{z} A(\underline{z}) \to A(\underline{s})$ is realized by terms obtained from the realizing terms of the usual functional interpretation by replacing the constants $\widetilde{s}$ with the corresponding $\widetilde{s^*}$ where $\underline{s^*}$ are given by Lemma 4.14 .

Using Remark 4.13 it follows that there exists $k \in \mathbb{N}$ constant such that the verifying proof for some axiom A of $\mathtt{EIL}^{\omega}_{\mathtt{M}}+\mathtt{AC}+\mathtt{IP}_{\forall}+\mathtt{MK}$ has depth upper bounded by $k \cdot qs(A)$. The verifying proof for $\mathtt{CT}\wedge$ makes use of ED .

Remark 4.17 The proof–size measure S_m is introduced in Definition 3.33 . The proof–depth measures $\partial_{\mathtt{MP}}$, $\partial_{\mathtt{QR}}$ and ∂ are introduced in Section 1.2 . In the following theorem we will abbreviate by $\partial_{\mathtt{MP}} :\equiv \partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}} :\equiv \partial_{\mathtt{QR}}(\mathcal{P})$, $\partial :\equiv \partial(\mathcal{P})$ and $S_m :\equiv S_m(\mathcal{P})$.

Since monotone functional interpretation uses the same algorithm as the usual functional interpretation for producing realizing terms for conclusions given the realizing terms for premises, the following analogue of Theorem 3.38 holds.

Theorem 4.18 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of A in $\mathtt{EIL}^{\omega}_{\mathtt{M},+}+\mathtt{AC}+\mathtt{IP}_{\forall}+\mathtt{MK}$ it produces as output $\underline{t^*}$ such that, with the notations 3.24, the following hold:

$$\begin{aligned} d(\underline{t}) \;&\leq\; k + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot \partial_{\mathtt{MP}} \\ S(\underline{t}) \;\leq\; Sz(\underline{t}) \;&\leq\; min\{k \cdot S_m \,,\, k \cdot \partial_{\mathtt{QR}} \cdot qs_{\mathrm{o}}^{\partial_{\mathtt{MP}}}\} \;\leq\; k \cdot qs_{\mathrm{o}}^{\partial} \qquad (63) \\ mdg(\underline{t}) \;&\leq\; k + vdg_{\mathrm{o}} + id_{\mathrm{o}} \\ mar(\underline{t}) \;&\leq\; k + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}} \\ \mathtt{EIL}^{\omega}_{\mathtt{M,v}} \;&\vdash_{k\cdot(qs_{\mathrm{o}}+\partial)}\; \exists \underline{x}\, (\underline{t^*}\ maj\ \underline{x} \,\wedge\, \forall \underline{a}, \underline{y}\, A_{\mathtt{D}}(\underline{x}(\underline{a}), \underline{y}, \underline{a})) \qquad (64) \end{aligned}$$

The time overhead of the algorithm is upper bounded by $k \cdot qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m$. The $\widetilde{\cdot}$ constants of $\mathtt{EIL}^{\omega}_{\mathtt{M,v}}$ in (64) are among those corresponding to terms occurring in the leaves of $\mathcal{P}$.

Proof: The rightmost inequality of (63) follows from a suitable adaptation of Remark 3.39 to the monotone case. We now only need to comment on (64). In order to build the verifying proof for MP we need to use the following lemma:

$$(\underline{y_1}\ maj\ \underline{x_1}) \wedge (\underline{y_3}\ maj\ \underline{x_3}) \to \wedge [\, \Sigma(y_3, \underline{y_1}, \underline{0})\ maj\ \Sigma(x_3, \underline{x_1}, \underline{0}) \,] \qquad (65)$$

Using Remark 4.13 it follows that there exists $k' \in \mathbb{N}$ such that for all its instances, lemma (65) has a proof of depth at most $k' \cdot |y_3, \underline{y_1}, \underline{0}|$. When used for verifying $\mathtt{MP}$, we have $|y_3, \underline{y_1}, \underline{0}| \le qs_o$, hence (64) follows immediately. □

Remark 4.19 If (65) were taken as axiom, the depth of verifying $\mathtt{MP}$ would be upper bounded by a constant, just like in the case of usual functional interpretation. On the other hand (65), $\Sigma\ maj\ \Sigma$, $\Pi\ maj\ \Pi$, $P\ maj\ P$ and $\mathcal{M}\ maj\ \mathcal{M}$ would have constant-depth proofs in $\mathtt{EIL}^{\omega}_{\mathtt{M}}$ if the underlying logical system handled tuples of conjunctions more smoothly. In such a case (64) could be replaced with $\mathtt{EIL}^{\omega}_{\mathtt{M,v}} \vdash_{k\cdot\partial} \exists \underline{x}\, (\underline{t^*}\ maj\ \underline{x} \wedge \forall \underline{a}, \underline{y}\, A_{\mathtt{D}}(\underline{x}(\underline{a}), \underline{y}, \underline{a}))$. Hence the bound on verifying proof depth would be better than in the usual functional interpretation case, see (43). The smoother treatment of tuples of conjunctions would actually be normal in our context with free use of tuples in both quantifier axioms/rules and the extensionality rule $\mathtt{ER_0}$.

Let $\mathtt{ECL}^{\omega}_{\mathtt{M},+}$ be the classical variant of $\mathtt{EIL}^{\omega}_{\mathtt{M},+}$. Combined with N-translation, monotone functional interpretation carries over to $\mathtt{ECL}^{\omega}_{\mathtt{M},+} + \mathtt{AC_0}$ and the upper bounds on size and proof depth are smaller than in the functional interpretation case. The following analogue of Theorem 4.8 + Corollary 4.10 holds.

Theorem 4.20 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of A in $\mathtt{ECL}^{\omega}_{\mathtt{M},+} + \mathtt{AC_0}$, it produces as output $\underline{t^*}$ such that, with the notations 3.24 and the abbreviations $\partial :\equiv \partial(\mathcal{P})$ and $S_m :\equiv S_m(\mathcal{P})$ the following hold :

$$d(\underline{t^*}) \le k \cdot qs_o \cdot \partial$$

$$S(\underline{t^*}) \le Sz(\underline{t^*}) \le k \cdot S_m \le (k \cdot qs_o)^{k\cdot\partial}$$

$$mdg(\underline{t^*}) \le vdg_o + k \cdot fid_o$$

$$mar(\underline{t^*}) \le var_o + k \cdot qs_o \cdot fid_o$$

$$\mathtt{EIL}^{\omega}_{\mathtt{M,v}} \vdash_{k\cdot(qs_o+\partial)} \exists \underline{x}\, [\, \underline{t^*}\ maj\ \underline{x} \wedge \forall \underline{a}, \underline{y}\, (A^{\mathtt{N}})_{\mathtt{D}}(\underline{x}(\underline{a}), \underline{y}, \underline{a}) \,] \qquad (66)$$

For $A \equiv \forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x}, \underline{y})$ with A_0 quantifier-free and $\{\underline{x}, \underline{y}\} = \mathcal{V}_{\mathrm{f}}(A_0)$, (66) can be replaced with

$$\mathtt{EIL}^{\omega}_{\mathtt{M,v}} \vdash_{k\cdot(ld(A_0)+qs_o+\partial)} \exists \underline{Y}\, [\, \underline{t^*}\ maj\ \underline{Y} \wedge \forall \underline{x}\, A_0(\underline{x}, \underline{Y}(\underline{x})) \,] \qquad (67)$$

The time overhead of the algorithm is upper bounded by $k \cdot qs_o \cdot ls_o \cdot S_m$. The $\tilde{\ }$ constants of $\mathtt{EIL}^{\omega}_{\mathtt{M,v}}$ in (66, 67) are among those corresponding to terms occurring in the leaves of $\mathcal{P}^{\mathtt{N}}$.

In concrete applications of monotone functional interpretation, $\mathtt{EIL}^{\omega}_{\mathtt{M}}$ will be extended by certain arithmetical (and even analytical) principles (see Section 5 below).

In the presence of a modest amount of arithmetic we can make use of t^* extracted by monotone functional interpretation in the following way. Let $\underline{x}$, $\underline{y}$ be of type o. Then (67) implies $\forall \underline{x} \exists \underline{y} \leq \underline{t^*}(\underline{x})\, A_0(\underline{x}, \underline{y})$ and therefore, using bounded search applied to $\underline{t^*}$ and a characteristic term t_{A_0} for A_0 one easily constructs $\underline{t}$ such that $\forall \underline{x}\, A_0(\underline{x}, \underline{t}(\underline{x}))$. This also works for $\underline{x}$ of type 1 using the construction $x^{\mathtt{M}}(i) :\equiv max_{j \leq i}\, x(j)$ since $x^{\mathtt{M}}$ *maj* x. Moreover, for sentences of the form $\forall x^1 \forall z \leq_1 s\, \exists y^o\, A_0(x, z, y)$ with s closed term one can easily obtain a type-2-term $\widehat{t}$ from t^* such that $\vdash \forall x^1 \forall z \leq_1 s\, \exists y \leq_o \widehat{t}(x)\, A_0(x, z, y)$ by taking $\widehat{t}(x) :\equiv t^*(x^{\mathtt{M}}, s^*)$ where s^* is a majorizing term for s. The term $\widehat{t}$ provides a uniform bound on y which is independent from z. See [30] for more details. This feature of monotone functional interpretation is of crucial importance in applications to numerical analysis [35] where $\{z \mid z \leq_1 s\}$ is used to represent compact Polish spaces. Since $A_0(x, z, y)$ is monotone (i.e., $A_0(x, z, y_1) \wedge y_2 \geq y_1 \rightarrow A_0(x, z, y_2)$) in most applications, the term $\widehat{t}$ will not be only a bound but actually a realizer for $\exists y$. Hence in this context monotone functional interpretation even provides a realizer which is independent from z and of simpler structure than realizers produced by the usual functional interpretation (see [31] for more on this).

5 Extensions to Arithmetic and fragments of Analysis

Both Gödel's functional interpretation and the monotone functional interpretation apply to intuitionistic and, via the negative translation, also classical arithmetic [22,32,55] (even in finite types) and fragments thereof [8,32,43]. Let $\mathtt{PRA}^\omega$ be Feferman's system [13] of primitive recursive arithmetic in all finite types, where only quantifier-free induction and ordinary Kleene-primitive recursive functionals are included. Let $\mathtt{PRA}^\omega_{\mathtt{i}}$ be its intuitionistic variant, formulated over $\mathtt{EIL}^\omega$ (see Section 5.1 of [26]). All the quantitative results proved above in Theorems 3.26, 4.18 and Theorems 4.8, 4.20 carry on to $\mathtt{PRA}^\omega_{\mathtt{i}}$, respectively $\mathtt{PRA}^\omega$ in the obvious way. The system $\mathtt{PRA}^\omega + \mathtt{AC}_0$ allows to derive the schemata of Σ^0_1-induction and Δ^0_1-comprehension (see [30]) and therefore contains the system $\mathtt{RCA}_0$ known from reverse mathematics (see [50]). Let us denote by $\mathtt{WKL}$ the binary König's lemma. This important[42] analytical principle simply asserts that every infinite binary tree has an infinite path. The second author has proved in [30] by means of a combination of functional interpretation and majorizability (a precursor of monotone functional interpretation) that $\mathtt{PRA}^\omega + \mathtt{AC}_0 + \mathtt{WKL}$ (which contains Friedman's system[43] $\mathtt{WKL}_0$ of [14,50]) is Π^0_2-conservative over $\mathtt{PRA}^\omega_{\mathtt{i}}$. Moreover, a witnessing term can be pro-

[42] A comprehensive discussion of the vast mathematical applicability of $\mathtt{WKL}$ is in[50].
[43] Theorem I.10.3 of [50] gives a summary of important mathematical statements which are theorems of $\mathtt{WKL}_0$. We only mention here the Heine-Borel covering lemma, the separable Hahn-Banach theorem and Brouwer's fixed point theorem.

vided. We give below a quantitative version of this result. We follow closely the proof in Section 7 of [3] which is a simplification of the more general method of [30]. Let $\mathtt{PRA}^\omega$ be formulated over $\mathtt{ECL}^\omega_{\mathtt{M}}$. We use the following convenient formulations of the binary König's lemma :

$$\mathtt{WKL} : \forall f\, [\forall k\, \neg Bnd(BTr(f), k) \;\rightarrow\; \exists b\, \forall\, k\, (InSeg(Bin(b), k) \in BTr(f))\,]$$

$$\mathtt{WKL}' : \forall f\, \exists b\, \forall\, k\, [\, \neg Bnd(BTr(f), k) \;\rightarrow\; InSeg(Bin(b), k) \in BTr(f)\,]$$

where (see Section 7 of [3] for full details)

- *Bin* and *BTr* are primitive recursive functionals which transform their argument to a binary function, respectively a binary tree ;
- *InSeg* is a primitive recursive functional which produces the length k initial segment of the binary function $Bin(b)$;
- *Bnd* is a primitive recursive predicate which expresses that the given binary tree $BTr(f)$ has depth at most k .

Remark 5.1 Below A_0 is quantifier-free, $\underline{x}, \underline{y}$ are type o and $\{\underline{x}, \underline{y}\} = \mathcal{V}_f(A_0)$.

The following theorem expresses the fact that the WKL–elimination and term extraction procedure from WKL–based proofs as developed in [30] is feasible both w.r.t. the size of the extracted terms and the depth of the verifying WKL–free proof. Although the feasibility of WKL–elimination was already proved (independently) in [23] and [2] for fragments of second-order arithmetic, the techniques employed there do not provide any term extraction procedure.

Theorem 5.2 There exists $k \in \mathbb{N}$ constant and an algorithm based on Gödel's functional interpretation which does the following. Given as input a proof

$$\mathcal{P} : \quad \mathtt{PRA}^\omega + \mathtt{AC}_0 \;\vdash_\partial\; \mathtt{WKL} \;\rightarrow\; \forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x}, \underline{y})$$

it produces at output realizing terms $\underline{t}$ such that $Sz(\underline{t}) \leq k \cdot S_c(\mathcal{P})$ and

$$\mathtt{PRA}^\omega_{\mathtt{i}} \;\vdash_{k \cdot (ls_o + \partial)}\; \forall \underline{x}\, A_0(\underline{x}, \underline{t}(\underline{x})) \;. \tag{68}$$

The time overhead of the algorithm is upper bounded by $k \cdot qs_o \cdot ls_o \cdot S_m(\mathcal{P})$.

Proof: The first step is to transform $\mathcal{P} : \mathtt{PRA}^\omega + \mathtt{AC}_0 \vdash_\partial \mathtt{WKL} \rightarrow \forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x}, \underline{y})$ to $\mathcal{P}^{\mathtt{N}} : \mathtt{PRA}^\omega_{\mathtt{i}} + \mathtt{AC}_0 + \mathtt{MK} \vdash_{k' \cdot (ld(A_0) + \partial)} \mathtt{WKL}' \rightarrow \forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x}, \underline{y})$ such that all sta–tements on $\mathcal{P}^{\mathtt{N}}$ in Proposition 4.4 hold. Here $\mathcal{P}^{\mathtt{N}}$ is obtained by a slight transformation within $\mathtt{PRA}^\omega_{\mathtt{i}} + \mathtt{MK}$ of the output from the N-translation algorithm carried on $\mathcal{P}$. There exist fixed proofs (hence with constant complexity) in $\mathtt{PRA}^\omega_{\mathtt{i}}$ of $\mathtt{WKL}' \rightarrow \mathtt{WKL}$ and $\mathtt{WKL} \rightarrow \mathtt{WKL}^{\mathtt{N}}$. See also Lemmas 7.3.1 and 7.3.3 of [3] .

The second step is to transform $\mathcal{P}^{\mathtt{N}}$ to the proof in (68) via a technique based on functional interpretation and majorization. This technique is described in Lemmas 7.4.1 and 7.4.2 of [3] and is an adaptation of the more general technique of [30] . The elimination of $\mathtt{WKL}'$ is achieved by weakening $\mathtt{WKL}'$ to a formula which is provable in $\mathtt{PRA}^\omega_{\mathtt{i}}$. Since we are here interested also in the realizing term for $\exists \underline{y}$ and not only in the WKL–conservation, we use a tuple-extended variant of Lemma 7.4.1 from [3] where a realizer for $\exists y$ is provided as well. $\Box$

Corollary 5.3 (quantitative WKL-conservation) There exists an algorithm which transforms proofs $\mathcal{P} : \mathtt{PRA}^\omega + \mathtt{AC}_0 \vdash_\partial \mathtt{WKL} \to \forall \underline{x} \, \exists \underline{y} \, A_0(\underline{x}, \underline{y})$ into proofs $\mathcal{P}' : \mathtt{PRA}^\omega_\mathtt{i} \vdash_{k \cdot (ls_o + \partial)} \forall \underline{x} \, \exists \underline{y} \, A_0(\underline{x}, \underline{y})$.

Remark 5.4 We could alternatively use a monotone functional interpretation version of Lemma 7.4.1 from [3] in the lines of our Theorem 4.20 . Then we would first obtain a majorizing tuple $\underline{t^*}$ for $\exists \underline{y}$ and we could produce a realizer by bounded search up to $\underline{t^*}(\underline{x})$ along the predicate $t_{A_0}(\underline{x}, \underline{y}) = 0$. Theorem 5.2 would now hold with (68) replaced by $\mathtt{PRA}^\omega_\mathtt{i} \vdash_{k \cdot (ld(A_0) + qs_o + \partial)} \forall \underline{x} \, A_0(\underline{x}, \underline{t}(\underline{x}))$. In many cases A_0 is monotone in $\underline{y}$ and therefore bounded search is actually not needed – see also the remarks following Theorem 4.20 . In such a case we would obtain terms $\underline{t}$ with $Sz(\underline{t}) \leq k \cdot S_m(\mathcal{P})$, time overhead at most $k \cdot fid_o \cdot qs_o \cdot S_m(\mathcal{P})$ and

$$\mathtt{PRA}^\omega_\mathtt{i} \vdash_{k \cdot (qs_o + \partial)} \forall \underline{x} \, A_0(\underline{x}, \underline{t}(\underline{x})) \tag{69}$$

hence a full better performance than the algorithm of Theorem 5.2 .

Remark 5.5 There are three ways to produce a variant of Theorem 5.2 where the input proof is $\mathcal{P} : \mathtt{PRA}^\omega + \mathtt{AC}_0 + \mathtt{WKL} \vdash_\partial \forall \underline{x} \, \exists \underline{y} \, A_0(\underline{x}, \underline{y})$. One way to overcome the failure of the deduction theorem for weakly extensional $\mathtt{PRA}^\omega$ is via the elimination-of-extensionality procedure from [39] . This applies when $\mathcal{P}$ contains only[44] variables of type 0 or 1 . In fact this is the case in most applications. We conjecture that the aforementioned procedure is feasible and hence the overall term extraction and WKL-conservation is still a feasible process. However if we are interested in the term extraction more than in the WKL-conservation we can state a variant of Theorem 5.2 based on the monotone functional interpretation with the verifying proof in $\mathtt{PRA}^\omega_\mathtt{i} + \widetilde{\mathtt{WKL}}$ and of the same depth as (69), where

$$\widetilde{\mathtt{WKL}} : \; \exists B \, \forall f \, \forall k \, [\neg Bnd(BTr(f), k) \; \to \; InSeg(Bin(B(f)), k) \in BTr(f)\,]$$

is a strengthening of $\mathtt{WKL}'$. If we are satisfied with a partial WKL-conservation then we can use the fact that premises of $\mathtt{ER}_0$ are realizer-free and hence any WKL instance used in the proof of such a $\mathtt{ER}_0$-premise gets discarded in the pre-processing phase of the (monotone) functional interpretation algorithm. We can thus consider that the input proof is in $\mathtt{PRA}^\omega + \mathtt{AC}_0 \oplus \mathtt{WKL}$ (see [30], p. 1246 for the meaning of $\oplus$ in this context). For this system the deduction theorem holds w.r.t. $\oplus$ and we obtain (68) with $\mathtt{PRA}^\omega_\mathtt{i}$ extended with the N-translations of conclusions of those $\mathtt{ER}_0$ instances in $\mathcal{P}$ whose sub-proof-trees use WKL . See

[44] Under this type restriction we can allow the use of (full) extensionality axiom EA, see also Remark 2.6 . Hence in this setting we work with the fully extensional $\mathtt{PRA}^\omega$ which features the deduction theorem.

also Remark 3.32.

Remark 5.6 Even though the term extraction procedure of Theorem 5.2 is extremely feasible, the normalization of the extracted terms into ordinary primitive recursive functions and the verification in (plain) primitive recursive arithmetic would however trigger a non-elementary-recursive complexity.

The results obtained above for systems based on $\mathtt{PRA}^\omega$ immediately carry on to the corresponding systems based on $\mathtt{PA}^\omega$ for suitable formulations of Peano Arithmetic in all finite types $\mathtt{PA}^\omega$, see Section 5.3 of [26] for more on this.

Acknowledgements: The authors are grateful to Paulo Oliva, Philipp Gerhardy and Daniele Varacca for their comments and suggestions on an earlier draft of this work. The exposition has consistently improved due to the referees comments. We also thank Prof. Helmut Schwichtenberg for useful comments.

References

[1] W. Alexi. Extraction and verification of programs by analysis of formal proofs. *Theoretical Computer Science*, 61:225–258, 1988.

[2] J. Avigad. Formalizing forcing arguments in subsystems of second-order arithmetic. *Annals of Pure and Applied Logic*, 82:165–191, 1996.

[3] J. Avigad and S. Feferman. Gödel's functional ('Dialectica') interpretation. In [7], pages 337–405.

[4] J. Barwise, editor. *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, Amsterdam, New York, Oxford, 1977.

[5] U. Berger, W. Buchholz, and H. Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.

[6] W. Burr. Functional interpretation of Aczel's constructive set theory. *Annals of Pure and Applied Logic*, 104:31–75, 2000.

[7] S.R. Buss, editor. *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics.* Elsevier, 1998.

[8] S. Cook and A. Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63:103–200, 1993.

[9] T. Coquand and M. Hofmann. A new method for establishing conservativity of classical systems over their intuitionistic version. *Mathematical Structures in Computer Science*, 9(4):323–333, 1999.

[10] J. Diller and W. Nahm. Eine Variante zur Dialectica Interpretation der Heyting Arithmetik endlicher Typen. *Arch. Mathem. Logik und Grundl.*, 16:49–66, 1974.

[11] A.G. Dragalin. New kinds of realisability and the Markov rule. *Dokl. Akad. Nauk. SSSR*, 251:534–537, 1980. In Russian, the English Translation is [12].

[12] A.G. Dragalin. New kinds of realisability and the Markov rule. *Soviet Math. Dokl.*, 21:461–464, 1980.

[13] S. Feferman. Theories of finite type related to mathematical practice. In [4], pages 913–972.

[14] H. Friedman. Systems of second order arithmetic with restricted induction, I, II (abstracts). *The Journal of Symbolic Logic*, 41:557–559, 1976.

[15] H. Friedman. Classical and intuitionistically provably recursive functions. In G.H. Müller and D.S. Scott, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–27. Springer Verlag, 1978.

[16] P. Gerhardy. The Role of quantifier alternations in Cut Elimination. To appear in *Notre Dame Journal of Formal Logic.*

[17] P. Gerhardy. Improved Complexity Analysis of Cut Elimination and Herbrand's Theorem. Master's thesis, University of Aarhus, Department of Computer Science, 2003.

[18] P. Gerhardy. Refined Complexity Analysis of Cut Elimination. In Matthias Baaz and Johann Makovsky, editors, *Proceedings of the 17th International Workshop CSL 2003*, volume 2803 of *LNCS*, pages 212–225. Springer-Verlag, Berlin, 2003.

[19] P. Gerhardy and U. Kohlenbach. Extracting Herbrand disjunctions by functional interpretation. To appear in *Archive for Mathematical Logic.*

[20] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérior.* PhD thesis, Université de Paris VII, 1972.

[21] K. Gödel. Zur intuitionistischen Arithmetik und Zahlentheorie. *Ergebnisse eines Mathematischen Kolloquiums*, 4:34–38, 1933.

[22] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287, 1958.

[23] P. Hájek. Interpretability and fragments of arithmetic. In *Arithmetic, proof theory, and computational complexity (Prague, 1991)*, volume 23 of *Oxford Logic Guides*, pages 185–196. Oxford Univ. Press, New York, 1993.

[24] S. Hayashi and H. Nakano. *PX: A Computational Logic.* MIT Press, 1988.

[25] M.-D. Hernest. A comparison between two techniques of program extraction from classical proofs. In M. Baaz, J. Makovsky, and A. Voronkov, editors, *LPAR 2002: Short Contributions and CSL 2003: Extended Posters*, volume VIII of *Kurt Gödel Society's Collegium Logicum*, pages 99–102. Springer Verlag, 2004.

[26] M.-D. Hernest and U. Kohlenbach. A complexity analysis of functional interpretations. Technical report BRICS RS-03-12, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark, Feb 2003. To appear in a slightly revised form in *Theoretical Computer Science – B.*

[27] W.A. Howard. Hereditarily majorizable functionals of finite type. In [55], pages 454–461.

[28] K.F. Jørgensen. Finite type arithmetic. Master's thesis, University of Roskilde, Departments of Mathematics and Philosophy, 2001.

[29] U. Kohlenbach. Proof Interpretations and the Computational Content of Proofs. *Lecture Course*, latest version in the author's web page.

[30] U. Kohlenbach. Effective bounds from ineffective proofs in analysis: an application of functional interpretation and majorization. *The Journal of Symbolic Logic*, 57(4):1239–1273, 1992.

[31] U. Kohlenbach. Analysing proofs in Analysis. In W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, editors, *Logic: from Foundations to Applications, Keele, 1993*, European Logic Colloquium, pages 225–260. Oxford University Press, 1996.

[32] U. Kohlenbach. Mathematically strong subsystems of analysis with low rate of growth of provably recursive functionals. *Arch. Math. Logic*, 36:31–71, 1996.

[33] U. Kohlenbach. On the no-counterexample interpretation. *The Journal of Symbolic Logic*, 64:1491–1511, 1999.

[34] U. Kohlenbach. A note on Spector's quantifier–free rule of extensionality. *Archive for Mathematical Logic*, 40:89–92, 2001.

[35] U. Kohlenbach and P. Oliva. Proof mining: a systematic way of analysing proofs in Mathematics. *Proceedings of the Steklov Institute of Mathematics*, 242:136–164, 2003.

[36] G. Kreisel. On the interpretation of non-finitist proofs, part I. *The Journal of Symbolic Logic*, 16:241–267, 1951.

[37] G. Kreisel. On the interpretation of non-finitist proofs, part II: Interpretation of number theory. *The Journal of Symbolic Logic*, 17:43–58, 1952.

[38] G. Kreisel. On weak completeness of intuitionistic predicate logic. *The Journal of Symbolic Logic*, 27:139–158, 1962.

[39] H. Luckhardt. *Extensional Gödel Functional Interpretation*, volume 306 of *Lecture Notes in Mathematics*. Springer Verlag, 1973.

[40] C. Murthy. *Extracting Constructive Content from Classical Proofs*. PhD thesis, Cornell University, 1990.

[41] V. P. Orevkov. Lower bounds on the increase in complexity of deductions after cut elimination. *Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov (LOMI)*, 88:137–162, 1979.

[42] V. P. Orevkov. Complexity of proofs and their transformations in axiomatic theories. In D. Louvish, editor, *Translations of Mathematical Monographs*, volume 128. American Mathematical Society, Providence, RI, USA, 1993.

[43] C. Parsons. On n-quantifier induction. *J. of Symbolic Logic*, 37:466–482, 1972.

[44] P. Pudlak. The lengths of proofs. In [7], pages 547–637.

[45] P. Rath. *Eine verallgemeinerte Funktionalinterpretation der Heyting Arithmetik endlicher Typen*. PhD thesis, Universität Münster, 1978.

[46] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924.

[47] H. Schwichtenberg. An arithmetic for polynomial-time computation. To be published by *Theoretical Computer Science*. Available in the author's web page.

[48] H. Schwichtenberg and S. Bellantoni. Feasible computation with higher types. In H. Schwichtenberg and R. Steinbrüggen, editors, *Proof and System–Reliability*, Proceedings of the NATO Advanced Study Institute, Marktoberdorf, 2001, pages 399–415. Kluwer Academic Publisher, 2002.

[49] H. Schwichtenberg and Others. Proof– and program–extraction system MINLOG. Free code and documentation at http://www.minlog-system.de.

[50] S.G. Simpson. *Subsystems of Second Order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, 1999.

[51] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In J.C.E. Dekker, editor, *Recursive function theory*, volume 5 of *Symposia in Pure Mathematics*, pages 1–27, 1962.

[52] R. Statman. Lower bounds on Herbrand's theorem. *Proceedings of the American Mathematical Society*, 75(1):104–107, 1979.

[53] M. Stein. *Interpretation der Heyting-Arithmetik endlicher Typen*. PhD thesis, Universität Münster, 1976.

[54] A.S. Troelstra. Realisability. In [7], pages 407–473.

[55] A.S. Troelstra, editor. *Metamathematical investigation of intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin - Heidelberg - New York, 1973.

[56] A.S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2nd edition, 2000.

A Index of Notations

The tables are to be read from left to right and then top–down.

Distribution of Definitions and Notations in (Sub)Sections							
Def. Not.	Section	Def. Not.	Section	Def. Not.	Section	Def. Not.	Section
2.1	2	2.4	2.3	2.12	2.3	3.1	3
3.5	3	3.10	3.1	3.24	3.3	3.27	3.3
3.33	3.4	3.34	3.4	3.38	3.4	4.11	4.2

Association of Definitions, Notations or (Sub)Sections to Symbols					
Name	Defined in	Name	Defined in	Name	Defined in
$\vdash_n$	Section 1.2	$\{\!\vert \cdot , \cdot \vert\!\}$	Definition 3.5	$\vert \cdot \vert$	Section 1.2
$\tilde{\cdot}$	Definition 3.10	$\geq_\sigma$	Definition 4.11	$=_\sigma , \neq$	Section 2.2
$ar(\cdot)$	Definition 2.1 + Section 2.2	$car(\cdot)$	Section 2.2	$car_1(\cdot)$	Notation 3.27
$cdg(\cdot)$	Section 2.2	$cdg_1(\cdot)$	Notation 3.27	$\mathcal{C}(\cdot)$	Section 2.2
$d(\cdot)$	Section 2.2	$d_{\mathcal{S}}(\cdot)$	Section 2.2 $\mathcal{S}$ meta-var.	$dg(\cdot)$	Definition 2.1 + Section 2.2
$\mathcal{D}$	Section 2.3	$\cdot^{\mathtt{D}} , \cdot_{\mathtt{D}}$	Definition 3.1	$\cdot$ Dr $\cdot$	Definition 3.5
$\partial(\cdot)$	Section 1.2	$\partial_L(\cdot)$	Section 1.2 L meta-var.	$\mathtt{E}$	Section 2.3
$fd(\cdot)$	Section 2.2	$fd_1(\cdot)$	Notation 3.24	$fid(\cdot)$	Section 2.2
$id(\cdot)$	Section 2.2	$id_0(\cdot)$	Notation 3.24	$\mathtt{I}$	Section 2.3
k_0	Section 1.2	$\lambda x.\, t$	Definition 2.12	$L(\cdot)$	Section 1.2
$ld(\cdot)$	Section 2.2	$ld_1(\cdot)$	Notation 3.24	$ls(\cdot)$	Section 2.2
$ls_0(\cdot)$	Notation 3.24	$ls_1(\cdot)$	Notation 3.24	$\mathrm{Lv}(\cdot)$	Section 1.2
$mdg(\cdot)$	Section 2.2	$mar(\cdot)$	Section 2.2	$\cdot\ maj_\sigma\ \cdot$	Definition 4.11
$\mathcal{M}_\sigma\ \cdot\ \cdot$	Definition 4.11	$\cdot\ ^{\mathtt{N}}$	Definition 4.1	$\mathtt{O}_\rho$	Section 2.3
Σ , Π	Definition 2.4 + Section 2.3	P	Section 2.3	$PR[\cdot]$	Definition 3.5
$qs(\cdot)$	Section 2.2	$qs_0(\cdot)$	Notation 3.24	$RR[\cdot]$	Definition 3.5
$RTS[\cdot]$	Definition 3.5	$S(\cdot)$	Section 2.2	$\mathtt{S}$	Section 2.3
$S_i(\cdot)$	Definition 3.33	$S_c(\cdot)$	Definition 3.33	$S_m(\cdot)$	Definition 3.33
$Sz(\cdot)$	Theorem 3.38	$Sz'(\cdot)$	Definition 3.34	$typ(\cdot)$	Section 2.2
$td(\cdot)$	Section 2.2	$ts(\cdot)$	Section 2.2	ν	Section 2.3
$var(\cdot)$	Section 2.2	$var_0(\cdot)$	Notation 3.24	$vdg(\cdot)$	Section 2.2
$vdg_0(\cdot)$	Notation 3.24	$\mathcal{V}(\cdot)$	Section 2.2	$\mathcal{V}_b(\cdot)$	Section 2.2
$\mathcal{V}_f(\cdot)$	Section 2.2	$\mathrm{Vt}(\cdot)$	Section 1.2	$wd(\cdot)$	Section 2.2
$wd_1(\cdot)$	Notation 3.27	$ws(\cdot)$	Section 2.2	$ws_1(\cdot)$	Notation 3.27