More feasible programs from (non-constructive) proofs by the Light (Monotone) Dialectica interpretation.

Mircea Dan Hernest

Extraction de programmes optimisés à partir de preuves (non-constructives) par l'interprétation Dialectica (monotone) légère

Pour l'obtention du titre de docteur au cadre du
Laboratoire d'Informatique (LIX) de l'École Polytechnique et
en co-tutelle de thèse au cadre du
Groupe de Logique Mathématique de l'Université de Munich (GKLI)

Presentée par

Mircea-Dan Hernest

de Slatina, Roumanie

Co-Directeur de thèse:	Prof. Dr. Jean-Pierre Jouannaud
Co-Directeur de thèse:	Prof. Dr. Helmut Schwichtenberg
Rapporteurs:	Dr. Rer. Nat. Ulrich Berger
	Dr. Michel Parigot
Examinateurs:	Prof. Dr. Ulrich Kohlenbach
	Prof. Dr. Christine Paulin
Journée de la soutenance publique:	14 Décembre 2006

Laboratoire d'Informatique (LIX)
École Polytechnique
2006

Déclaration - Declaration

Je déclare que cette thèse a été conçue par moi-même, que l'ouvrage qu'elle contienne est le mien, à l'exception du chapitre de l'Annexe. Ce dernier est bâti sur des travaux en commun avec Ulrich Kohlenbach, ce qui est aussi explicitement indiqué dans le texte principal. Aucune partie des travaux de ma thèse n'a été soumise pour un autre degré ou qualification professionnelle, à l'exception d'un tiers du chapitre de l'Annexe, qui avait compté pour l'obtention de mon titre de Mastère de l'Université de Aarhus, Danemark.

I declare that this thesis was composed by myself, that the labour contained herein is my own, except for the Appendix chapter. The latter is based on joint work with Ulrich Kohlenbach, which is also explicitely declared in the main text. My work has not been submitted for any other degree or professional qualification except for one third of the Appendix chapter, which had counted for the obtention of my Master degree from the University of Aarhus, Denmark.

Mircea-Dan Hernest

Dédicace - Dedication

Cette thèse est dédiée à tous ceux qui ont envisagé son apparition. This thesis is dedicated to all who believed that it would eventually come into being.

Mircea-Dan Hernest

Instruction pour l'impression: veuillez tourner les feuilles anterieures et ensuite eliminer cette feuille d'instructions.

Instruction for printing: please turn the previous sheets and then discard this instruction sheet.

Résumé

Cette thèse présente une nouvelle optimisation de l'interprétation fonctionnelle (dite "Dialectica") de Gödel pour l'extraction de programmes plus efficaces à partir de preuves arithmétiques (classiques). La version dite "légère" de Dialectica se combine aussi et encore plus facilement avec l'optimisation dite "monotone" de Kohlenbach sur l'interprétation fonctionnelle de Gödel pour l'extraction de majorants et marges plus efficaces à partir de preuves monotones arithmétiques et mêmes analytiques (classiques). La Dialectica légère est obtenue par l'adaptation des quantificateurs "uniformes" (sans signification computationnelle) de Berger. En plus, sa présentation est faite dans le style de la Déduction Naturelle, comme amélioration de l'adaptation récente de Jørgensen sur la Dialectica originelle de Gödel. Un bon nombre d'exemples concrèts sont traités sur l'ordinateur par la nouvelle technique d'extraction de programmes. La comparaison en machine avec la technique de synthèse de programmes appelée "A-traduction raffinée" de Berger, Buchholz et Schwichtenberg montre une très bonne performance de la Dialectica légère. Cette dernière n'est dépassée que dans le cas du Lemme de Dickson. Nous développons aussi la théorie de la synthèse de programmes efficaces, de complexité calculatoire polynômiale en temps, par la nouvelle Dialectica légère (monotone). Dans ce but nous métissons deux structures préexistantes de Cook-Urquhart-Ferreira et respectivement Kohlenbach pour obtenir une "Analyse bornée par un polynôme en temps". Ici, le résultat théorique est prometteur mais des exemples pratiques restent encore à trouver pour montrer la différence de cette structure par rapport à l'"Analyse bornée par un polynôme" de Kohlenbach.

Mots clé: "Proof Mining", Extraction de programmes à partir de preuves (classiques), complexité computationnelle des programmes extraites, complexité des preuves, l'interprétation fonctionnelle "Dialectica" de Gödel, complexité computationnelle, fonctionnelles de type fini, logique combinatorielle, vérification du software et des systèmes, quantificateurs existentiels et universels sans signification computationnelle (uniformes).

Résumé Étendu en Français

Des résultats pratiques importants[1] ont été obtenus dans les années récentes dans le domaine de la théorie extractive de la preuve, intitulée suggestivement "Proof Mining" par Kohlenbach [81]. L'implémentation des algorithmes d'extraction issus de la pure recherche métamathématique vient de donner des programmes intéressants et plutôt inattendus dans la plupart des cas (voir par exemple [10, 15, 50, 55, 104, 106]). Des approches très variées de l'extraction de programmes à partir de preuves *classiques*[2] ont été développées pendant plusieurs années de recherche, voir [3, 5, 10, 20, 23, 68, 87, 88, 91, 93, 101, 102, 106]. Ces techniques d'extraction peuvent être classifiés à peu près dans deux grands groupes parmi lesquels on peut distinguer plusieurs sous-groupes. D'un coté on a la ligne de recherche historiquement plus récente, issue plutôt de l'informatique théorique, qui a pour but l'extension de la correspondance de Curry-Howard à la logique classique pleine [5, 20, 87, 102]. Cette direction de recherche vient de l'essai de donner une interprétation algorithmique aux principes variées de logique classique. Ainsi Griffin [46] avait déjà remarqué qu'á l'opérateur $\mathcal{C}$ de Felleisen [29, 30] peut être assigné le type du schéma de la stabilité $\neg\neg A \to A$. Un résultat pratique impressionnant récent dans cette direction a été obtenu par Raffalli [106] qui a réussi à extraire un algorithme étonnamment rapide (dans des proportions relatives) pour le Lemme de Dickson dans sa forme la plus générale par le moyen d'une "Logique Combinée" ("Mixed Logic", en anglais).

De l'autre côté on a la ligne plus traditionnelle de recherche (qui fait partie intégrante de la théorie de la preuve) sur ce qu'on appelle "les interprétations des preuves", voir [63] pour des références historiques commentées et struc-

[1] Ici, par "résultats pratiques" nous entendons à la fois de nouvelles preuves mathématiques concrètes comme celles présentées dans [63, 81] mais aussi de nouveaux algorithmes pour l'ordinateur comme ceux décrits dans [10, 15, 50, 55, 104, 106].

[2] Les preuves *constructives* ont un contenu calculatoire plus ou moins explicite, voir, e.g., [7].

turées à ce sujet. L'utilisation des interprétations des preuves au lieu de l'application directe de l'élimination des coupures a ouvert le chemin vers l'obtention de meilleurs résultats pratiques par le moyen de ces techniques modulaires (voir [50] pour une petite discussion à ce sujet). L'interprétation fonctionnelle "Dialectica" de Gödel [45, 4] s'applique directement sur des preuves beaucoup plus complexes que sa forme plus simple (et aussi moins forte) qu'est la realisabilité modifiée de Kreisel [85]. La combinaison de cette dernière avec des améliorations (comme par exemple [3, 10, 23]) de la A-traduction[3] de Friedman [36] ne peuvent réparer que de manière partielle cette disparité (voir [50] ou [63] pour des discussions à ce sujet). Les réalisants exacts primaires donnés par Dialectica sont néanmoins beaucoup plus complexes que ceux produits par la technique de Berger, Buchholz et Schwichtenberg [10] (comme exemple).

L'explication pour une telle situation semble être l'inclusion entière des formules de contraction qu'on appelle "Dialectica-pertinentes" dans les Dialectica-réalisants primaires (c'est-à-dire les termes extraits pas encore normalisés). Même si dans certains cas les réalisants primaires obtenus par les les deux techniques normalisent vers à peu près les mêmes programmes (voir [50] pour un tel exemple), la normalisation des programmes extraits par le moyen des interprétations des preuves est généralement beaucoup plus coûteuse que leur synthèse[4]. Il est donc souhaitable que le terme extrait en premier lieu soit le plus simple possible pour que le coût total de la production du réalisant final en forme normale baisse en conséquence. Afin de gérer le problème difficile de la contraction, Kohlenbach [68] a conçu son interprétation fonctionnelle monotone, qui est une adaptation de la technique de Gödel pour l'extraction de majorants et bornes uniformes pour les réalisants exacts (voir aussi [33] pour l'"interprétation fonctionnelle bornée" de Ferreira et Oliva). Quoique tellement prolifique (voir [63] ou [81]) dans le contexte des applications à l'Analyse mathématique (numérique), la Dialectica monotone utilisée littéralement et seule n'est généralement pas vraiment pertinente pour la synthèse des réalisants exacts dans le cadre des Mathématiques Discrètes. À l exception de certaines situations dans lesquelles la spécification d'entrée est

[3] Ce genre de A-traduction raffinée d'habitude combine une traduction en double négation de preuves classiques en preuves intuitionnistes avec ce qu'on appelle "la A-traduction de Friedman-Dragalin", voir [26] pour la version Russe, indépendamment découverte.

[4] Voir le chapitre de l'Annexe pour une exposition théorique détaillée sur la très basse complexité computationnelle des méta-algorithmes d'extraction "Dialectica".

monotone dans la variable censée être réalisée, la production d'un réalisant exact à partir d'un majorant donné nécessite une recherche bornée et n'est possible que dans certains cas. Voir la fin de la Section 4 de [56] pour un commentaire étendu à ce sujet ou [63] pour des détails plus complèts (d'ordre technique).

Dans cette thèse nous proposons un autre genre d'optimisation de l'interprétation fonctionnelle de Gödel, par l'élimination complète et déjà du terme extrait en premier (c'est-à-dire pendant le processus d'extraction même) d'un nombre de formules de contraction qui sont pertinentes pour Dialectica, mais qui sont identifiées comme "computationnellement redondantes" par le moyen d'une adaptation des quantificateurs dits *uniformes* de Berger [9] au contexte de l'interprétation fonctionnelle. Ces quantificateurs sont nommés "sans contenu computationnel" en [111] et ici on va les désigner comme des "quantificateurs sans signification computationnelle", ce qu'on abrège par "`ncm`" (un acronyme pour *non computational meaning*, en anglais).

Nous appelons ici "Dialectica légère" (ou "interprétation fonctionnelle légère", comme dans [51]) [5] cette technique optimisée d'extraction des réalisants exacts, basée sur l'interprétation *Dialectica* de Gödel. Nous nommons aussi "Dialectica légère monotone" ou "interprétation fonctionnelle monotone légère", comme dans [55], sa correspondante monotone (dans le sens de Kohlenbach [68]) qui est une technique optimisée de synthèse de majorants et bornes uniformes.

En suivant [51], nous allons dénoter dans cet thèse par $\overline{\forall}$ le quantificateur universel `ncm` et par $\overline{\exists}$ le quantificateur existentiel `ncm`. Tandis que notre $\overline{\exists}$ est identique[6] au $\{\exists\}$ de Berger, notre $\overline{\forall}$ demande un renforcement supplémentaire de la restriction imposée par Berger sur la règle d'introduction de son quantificateur $\{\forall\}^+$ dans [9], voir [51]. Ceci est requis pour prendre en compte l'inclusion de ce qu'on appelle "contractions persistantes" dans les termes extraits par la Dialectica légère. Par contre, comme les formules de contraction ne sont pas comprises dans les majorants extraits par la Dialectica légère monotone, dans le contexte monotone, $\overline{\exists}$ et $\overline{\forall}$ sont tous deux identiques aux quantificateurs de Berger. Aussi, pour les systèmes monotones il n'y aura pas de restriction sur la règle d'introduction de l'Implication. Cette restriction doit être ajoutée aux systèmes non-monotones pour assurer que la formule

[5] Ici *légère* est entendu comme l'opposé du *lourd*.

[6] Modulo la formulation comme axiomes à l'instar du [111] des règles de Berger pour son $\{\exists\}$ du [9].

d'introduction puisse être comprise effectivement dans le réalisant dans une situation de contraction persistante. Ainsi, les améliorations *monotone* et *légère* de Dialectica vont mieux se combiner dans un système arithmétique plus simple qui pourra être étendu plus facilement aux preuves de logique classique complète de la manière habituelle. Par contre, les systèmes "light" non-monotones ne s'étendront pas aux preuves classique complètes, sauf si on élimine les quantificateurs existentiels forts, et même avec cela une telle extension sera encore plus compliquée techniquement.

Résumé Étendu des Chapitres de cette thèse

Le Chapitre 1

Dans le premier Chapitre nous introduisons les systèmes logiques, arithmétiques et analytiques utilisés par la suite, à l'exception des "systèmes efficaces" de l'Arithmétique et l'Analyse du Chapitre 3. Ces derniers seront néanmoins présentés dans un cadre qui adapte celui introduit dans le Chapitre 1 aux contextes "efficaces" variés.

On construit donc d'abord une variante faiblement extensionnelle $\mathtt{WeZ}$ de l'Arithmétique intuitionniste $\mathtt{Z}$ de Berger, Buchholz et Schwichtenberg introduite en [10]. Par rapport à $\mathtt{Z}$, le système $\mathtt{WeZ}$ pose une certaine restriction sur l'extensionnalité de l'égalité. Pour mieux exprimer cette restriction, on va définir l'égalité comme une constante fonctionnelle (dénotée $\mathtt{Eq}$, voir la Définition 1.1) au lieu d'une constante predicative comme d'habitude. Ce traitement de l'égalité est tout à fait caractéristique de l'interprétation Dialectica de Gödel[7]. Nous introduisons aussi une suite d'extensions de $\mathtt{WeZ}$ en principe avec le quantificateur existentiel fort (intuitionniste) d'un côté et de l'autre côté avec les variantes dites *sans-signification-calculatoire* ($\mathtt{ncm}$) du quantificateur universel et (si présent) le quantificateur existentiel (les deux nouveaux quantificateurs $\mathtt{ncm}$ ont déjà été mentionnés dans le préambule ci-dessus). Nous avons donc les extensions dénotées $\mathtt{WeZ}^{\exists}$, $\mathtt{WeZ}^{\mathtt{nc}}$ et respectivement le système le plus complète $\mathtt{WeZ}^{\exists,\mathtt{nc}}$, voir la Section 1.3 ci-dessous. À son tour, le système $\mathtt{WeZ}$ est bâti par étapes, comme le plus complète de la séquence d'extensions $\mathtt{ML}_0 \subset \mathtt{ML} \subset \mathtt{IL}_0 \subset \mathtt{IL} \subset \mathtt{WeZ}$. Le système $\mathtt{ML}_0$ est une pure logique minimale predicative. Le système $\mathtt{ML}$ est une logique minimale

[7] Voir, e.g., les sections correspondantes de [123], [4] ou [63] pour des discussions détaillées à ce sujet.

bâtie au-dessus de $\mathtt{ML}_0$ en ajoutant les axiomes de la vérité et de la fausseté booléenne. Ensuite, $\mathtt{IL}_0$ ajoute à $\mathtt{ML}$ ce qu'on appelle l'"Axiome de l'Induction Booléenne" et aussi toutes les axiomes de l'égalité concernant les formules booléennes atomiques. Finalement, $\mathtt{IL}$ est une pure logique intuitionniste qui ajoute à $\mathtt{IL}_0$ l'équivalence logique entre la fausseté booléenne (déjà introduite) et la fausseté logique ($\bot$, qui n'était auparavant qu'un simple prédicat d'arité zéro). En plus, chacun des systèmes $\mathtt{ML}_0$, $\mathtt{ML}$, $\mathtt{IL}_0$ et $\mathtt{IL}$ a sa propre version avec $\exists$ ou/et les quantificateurs $\mathtt{ncm}$, ce qui donne le tableau suivant:

$$\begin{array}{ccccccccc}
\mathtt{ML}_0^{\exists} & \subset & \mathtt{ML}^{\exists} & \subset & \mathtt{IL}_0^{\exists} & \subset & \mathtt{IL}^{\exists} & \subset & \mathtt{WeZ}^{\exists} \\
\mathtt{ML}_0^{\mathtt{nc}} & \subset & \mathtt{ML}^{\mathtt{nc}} & \subset & \mathtt{IL}_0^{\mathtt{nc}} & \subset & \mathtt{IL}^{\mathtt{nc}} & \subset & \mathtt{WeZ}^{\mathtt{nc}} \\
\mathtt{ML}_0^{\exists,\mathtt{nc}} & \subset & \mathtt{ML}^{\exists,\mathtt{nc}} & \subset & \mathtt{IL}_0^{\exists,\mathtt{nc}} & \subset & \mathtt{IL}^{\exists,\mathtt{nc}} & \subset & \mathtt{WeZ}^{\exists,\mathtt{nc}} \quad .
\end{array}$$

Les Sections 1.2 et 1.3 dans la suite contiendront les définitions détaillées de tous ces systèmes. Les $\mathtt{ncm}$-arithmétiques $\mathtt{WeZ}^{\mathtt{nc}}$ et $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ seront alors étendues dans la Section 1.4 avec à la fois les principes logiques non-intuitionnistes habituels (eux-aussi étendus au nouvel contexte $\mathtt{ncm}$) mais aussi des nouveaux principes que, eux aussi sont directement réalisés par l'interprétation Dialectica légère. Ces nouveaux principes sont obtenus en remplaçant le $\exists$ fort par le $\exists^{\mathtt{cl}}$ faible dans certaines restrictions des axiomes habituels de l'Indépendance des Prémisses et Axiome du Choix. Voir la Section 1.4 pour plus de détails.

Ensuite, dans la Section 1.5 on va construire les variantes monotones des $\mathtt{ncm}$-arithmétiques intuitionnistes et étendues mentionnées ci-dessus. La plus importante qualité de ces nouvelles $\mathtt{ncm}$-arithmétiques monotones $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$, $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$, $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$ et $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+}$ est l'élimination de la restriction plustôt drastique sur la règle d'introduction de l'implication. Celle-ci n'est plus requise dans le contexte monotone, voir la Remarque 1.16. La Dialectica monotone légère pourra donc extraire plus de programmes à partir de plus de preuves qui utilisent les quantificateurs $\mathtt{ncm}$, voir le Théorème 2.12. Finalement, nous étendons les $\mathtt{ncm}$-arithmétiques à la fois monotones et non-monotones à la logique classique toute-entière, voir la Section 1.6. Nous verrons plus tard, dans la Section 2.3 que l'extraction par la Dialectica légère a des grandes difficultés à s'élargir dans un contexte de logique pleinement classique. Au contraire, la Dialectica monotone légère s'étend très rapidement à la logique classique pleine, de la manière habituelle, car il n'y a pas de restriction sur l'Introduction de l'Implication.

Selon ses auteurs, le système $\mathtt{Z}$ [8] est une extension du système **T** de Gödel

[8] Ainsi que son compagnon naturel $\mathtt{Z}^{\exists}$, qui ajoute simplement à $\mathtt{Z}$ les axiomes définissant

avec tout l'appareil logique et arithmétique qui le rend approprié et convenable pour l'extraction concrète de programmes à partir de preuves classiques par le moyen de la A-traduction raffinée de [10]. On peut dire la même chose des systèmes $\mathtt{WeZ}^{\exists}$, $\mathtt{WeZ}^{\mathtt{nc}}$ et $\mathtt{WeZ}^{\exists,\mathtt{nc}}$, relativement à l'extraction en machine par l'interprétation Dialectica légère qui fait l'objet de cette thèse. Les systèmes $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$, $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+}$ et $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ apparaissent également comme l'extension optimale d'ordre logique-et-arithmétique du système **T** pour l'extraction de majorants et bornes uniformes par la Dialectica monotone légère (éventuellement en combinaison avec les traductions négatives de Kouroda ou Gödel-Gentzen dans le cas de preuves entièrement classiques, voir la Section 2.3 ci-dessous).

Le Chapitre 2

Le deuxième Chapitre présente notre formulation de l'amélioration "légère" de l'interprétation fonctionnelle de Gödel et de sa variante monotone due à Kohlenbach. Pour la conception de la Dialectica légère, nous avons transformé la formulation en Déduction Naturelle de Jørgensen [61] en éliminant son "Lemme de la Contraction" [9] et aussi en permettant la présence des variables libres dans les termes extraits[10]. Nous avons aussi adapté les quantificateurs existentiel et universel "uniformes" de Berger [9] au contexte de l'extraction de programmes par Dialectica. L'emploi de ce genre de quantificateurs *sans signification calculatoire* permet l'identification et l'isolement des formules de contraction qui seraient autrement entièrement incluses de manière redondante dans les termes extraits par la pure Dialectica. Nous construisons aussi la combinaison de notre simplification de l'interprétation Dialectica de Gödel avec son adaptation pour la synthèse de majorants et bornes uniformes due à Kohlenbach dans une "interprétation fonctionnelle *monotone légère*". Dans le fin du Chapitre 2 nous étendons les deux nouvelles interprétations aux preuves classiques complètes. Maintenant on décrit plus largement le contenu du deuxième Chapitre.

Par "LD-interprétation" (une abréviation pour "interprétation Dialectica légère") nous appelons ci-dessous notre adaptation de l'interprétation fonc-

l'existentiel fort $\exists$. Le $\exists$ intuitionniste est présent déjà dans [10], mais plutôt de manière implicite. Il est néanmoins présenté et traité plus explicitement dans [111].

[9] Nous avons trouvé que cette construction purement technique est trop compliquée, particulièrement dans le but d'une implémentation sur ordinateur pour la synthèse de programmes par Dialectica (monotone, légère).

[10] Celle-ci s'avère plus appropriée dans le cadre de la Déduction Naturelle, voir [57] pour une discussion à ce sujet.

tionnelle dite "Dialectica" de Gödel [45] pour l'extraction de programmes plus efficaces à partir de preuves (éventuellement classiques). La LD-interprétation est une traduction syntaxique récurrente à partir de preuves en $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ (ou en $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$, modulo la traduction par double négation de Kuroda) vers des preuves dans $\mathtt{WeZ}^{\exists}$ ou même WeZ, de manière que les occurrences positives de $\exists$ et les occurrences négatives de $\forall$ dans la formule conclusion de la preuve d'entrée arrivent à être réalisées effectivement par des termes du système $\mathbf{T}$ de Gödel. Ces "termes réalisants" sont également appelés *les programmes extraits* par la LD-interprétation. Si l'intérêt n'est que l'obtention de programmes, ce processus de traduction est aussi appelé "extraction des programmes". La traduction de la preuve d'entrée est aussi appelée *la preuve vérificatrice*, car elle confirme le fait que le programme extrait réalise effectivement le coté existentiel de la LD-interprétation (LD-traduction) de la formule conclusion de la preuve d'entrée.

L'interprétation Dialectica de Gödel (abrégée "D-interprétation") devient relativement beaucoup plus compliquée quand elle doit faire face à la *Contraction* (calculatoirement pertinente[11]), qui en Déduction Naturelle se manifeste par le déchargement de plus qu'une copie d'une formule hypothèse pendant l'Introduction de l'Implication $\rightarrow^{+}$. Afin de confronter ce problème, Kohlenbach avait introduit dans [68] une méthode élégante pour la simplification du traitement de la Contraction quand le but final est l'extraction des fonctionnelles qui majorent (dans le sens de Howard [58]) les Dialectica-réalisants actuels. Il avait appelé "interprétation fonctionnelle monotone" cette variante de la D-interprétation qu'on abrège ici par "MD-interprétation".

La MD-interprétation a été utilisée avec grand succès pendant les dix dernières années pour la production des preuves d'importantes nouveaux théorèmes de l'Analyse fonctionnelle numérique. Les ouvrages [81] et [63] contiennent des vues générales d'ensemble de ce genre d'applications récentes de la MD-interprétation aux preuves mathématiques concrètes de la littérature - pour des références directes voir, e.g., [68, 62, 64, 66, 67, 75, 76, 78, 79, 80, 82, 41, 77]. Ces théorèmes sont complètement nouveaux dans le sens qu'ils n'ont pas été établis antérieurement par les moyens mathématiques plus habituels (de l'intérieur des théories), malgré les essais. Néanmoins, chaque fois qu'il s'agit

[11] Dans le sens de la Définition 1.18. La pure D-interprétation doit confronter une classe plus large de ce qu'on appelle "des contractions calculatoirement D-pertinentes". La caractéristique clé de l'optimisation apportée par la LD-interprétation est que il peut y avoir des contractions D-pertinentes qui s'avèrent ne pas être LD-pertinentes. Celles-ci sont donc LD-non-pertinentes et donc D-redondantes, voir la Remarque 1.19.

d'un problème d'extraction de réalisants exacts, la MD-interprétation n'apporte pas nécessairement des réponses efficaces. Il apparaît donc le besoin d'une optimisation de type différent de la D-interprétation de Gödel. Nous proposons dans ce chapitre la LD-interprétation comme une amélioration de la technique de Gödel pour l'extraction de programmes plus efficaces. En plus, la même optimisation s'applique également à l'extraction de majorants et bornes uniformes plus efficaces par ce qu'on appelle "l'interprétation fonctionnelle monotone légère", abrégée *la LMD-interprétation* (ou l'interprétation fonctionnelle bornée légère, en suivant [33] plutôt que [68]).

La D-interprétation a été introduite pour la première fois dans [45] pour une formulation de l'Arithmétique dans le style de Hilbert - voir aussi [123, 90, 25, 4, 63] pour d'autres expositions plus modernes utilisant aussi des systèmes à la Hilbert. Des formulations en Déduction Naturelle de la variante de Diller-Nahm de la D-interprétation ont été élaborées par les étudiants de Diller, Rath [107] et Stein [121]. Ce n'est qu'en 2001 que Jørgensen a publié dans [61] une première élaboration en Déduction Naturelle de la variante originale due à Gödel de l'Interprétation Fonctionnelle. Dans le cadre de Diller-Nahm, tous les choix entre les réalisants potentiels des contractions sont reportés jusqu'à la fin en collectant tous les candidats et en faisant un seul choix global à la fin. Au contraire, la formulation de Jørgensen respecte le traitement originel de Gödel pour la Contraction par des choix locaux immédiats. Jørgensen utilise alors un certain "Lemme de la Contraction" afin de manier, dans le contexte donné de Déduction Naturelle, le déchargement de plus qu'une copie d'une hypothèse pendant une Introduction de l'Implication $\rightarrow^+$. Si $n+1$ occurrences non-déchargées d'une hypothèse sont sur le point d'être supprimées pendant un $\rightarrow^+$, alors Jørgensen emploie n fois son "Lemme de la Contraction", tout en déplaçant les résultats partiels en avant et en arrière de *la porte de la preuve* $\vdash$. Nous considérons ceci non seulement inefficace du point de vue de la perspective de la synthèse appliquée de programmes, mais aussi inélégant, car la preuve de consistance de la D-interprétation se complique sans nécessité par rapport à la contraction.

Au lieu de tout cela, nous allons utiliser ce qu'on appelle "le n-sélecteur" $\mathtt{If}^n_\tau$ pour égaliser en une seule étape (composée) toutes les LD-traductions des $n+1$ cópies non-déchargées de l'hypothèse en cours de traitement, voir la preuve du Théorème 2.10. Quoiqu'il soit techniquement impossible d'avoir à sa disponibilité un n-sélecteur direct pour tout $n \in \mathbb{N}$, dans certaines implémentations améliorées, $\mathtt{If}^n_\tau$ pourrait être donné une définition plus directe pour tous

les $n \leq N$, pour une certaine borne supérieure N, au lieu de le simuler par n applications séquentielles du $\mathtt{If}^1_\tau$. Après tout, on n'utiliserait jamais en pratique plus qu'un certain nombre limité de n-sélecteurs. Le bénéfice pratique par rapport à la solution de Jørgensen est que le traitement de la Contraction est alors transporté directement et complètement du niveau des preuves au niveau des termes. Ainsi, le déplacement en avant et en arrière du $\vdash$ n'est plus requis dans la construction de la preuve vérificatrice.

Nous modifions la version de Jørgensen de la D-interprétation aussi en permettant des variables libres dans les termes extraits. Ce choix correspond à la formulation du système **T** de Gödel avec la λ-abstraction comme primitive et c'est plus naturel dans le contexte de la Déduction Naturelle. En addition nous incluons le traitement de notre adaptation des quantificateurs existentiel et universel uniformes de Berger [9] au contexte de l'extraction de programmes par Dialectica. En fait, ce sont exactement ces quantificateurs "sans signification calculatoire" (ou ncm tout court) qui apportent l'optimisation qu'on appelle "légère" de la Dialectica, car ils peuvent être utilisés pour l'isolement des certaines contractions calculatoirement redondantes dans les termes extraits primaires. Dans le contexte monotone, les quantificateurs ncm sont exactement ceux de Berger, sans aucune restriction supplémentaire sur l'Introduction de l'Implication ou sur l'Introduction du $\overline{\forall}$, voir la Définition 1.64. Même si le traitement de la Contraction est plus simple pour la Dialectica monotone, l'optimisation "légère" peut toujours apporter une sérieuse amélioration supplémentaire, voir le commentaire du dernier paragraphe de la Section 2.2. Ainsi, l'interprétation Dialectica monotone légère est une combinaison de succès des deux optimisations de la Dialectica, qui se métissent de manière très harmonieuse. En conséquence, cette LMD-interprétation s'étend aussi très facilement à l'extraction de programmes à partir de preuves pleinement classiques, contrairement à la LD-interprétation qui a de très grandes difficultés de faire de même, à cause des ncm-restrictions plus lourdes, voir la Section 2.3 pour tout le détail à ce sujet.

Le Chapitre 3

Dans le troisième Chapitre nous donnons une exposition de deux structures-cadre bien établies pour la spécification des programmes de complexité calculatoire polynomiale en temps, chacune ensemble avec l'adaptation de l'arithmétique correspondante pour l'extraction de ce genre de programmes efficaces par le moyen de la Dialectica (monotone, légère). Ainsi nous avons d'un coté la

structure de l'"Arithmétique polynomiale en temps" de Cook et Urquhart [22] et d'autre coté la structure de l'"Analyse bornée par polynômes" de Kohlenbach [69] (abrégée PBA). Ces structures sont présentées toutes les deux comme des adaptations des systèmes arithmétiques introduites dans le Chapitre 1. Des interprétations "Dialectica (monotone) légère" sont ensuite conçues pour les deux structures-cadre comme des adaptations des interprétations de preuves définies dans le Chapitre 2.

Dès les débuts de la recherche dans le domaine de l'extraction de programmes à partir de preuves, une attention particulière a été portée à l'égard du problème de l'efficacité des programmes extraits. Dans le cas de l'interprétation fonctionnelle de Gödel [45], il était devenu rapidement très clair que toute application mathématique concrète d'une telle technique puissante de la théorie de la preuve devra être confrontée au problème difficile de la contraction. En conséquence on a proposé plusieurs simplifications du traitement de la contraction. C'est comme ça que les variantes de la Dialectica de Diller-Nahm [25], l'interprétation "absence de contre exemple" et la realisabilité modifiée de Kreisel [85] sont apparues. Le grand désavantage de toutes ces simplifications proposées est la perte de l'expressivité logique des preuves à l'entrée. Dans le cas de la realisabilité modifiée, une réparation partielle a été apportée par la A-traduction de Friedman-Dragalin, qui permet une quantité limitée de raisonnement non-constructif dans les preuves qu'on veut interpréter constructivement. Aussi, l'interprétation fonctionnelle monotone [68] a été conçue pour donner un traitement plus simple de la contraction par l'utilisation du maximum des termes extraits antérieurement, au lieu de décider lequel choisir en fonction de la validité actuelle de la formule de contraction.

En parallèle avec ce travail sur la simplification de la structure logique des techniques d'extraction, une recherche liée a commencée plutôt du coté de l'informatique sur la simplification de la complexité calculatoire du système de termes, c'est-à-dire (en principe) le $\mathbf{T}$ de Gödel. Ensuite, des systèmes pleines logiques et arithmétiques ont été conçus pour la formalisation du raisonnement sur ce genre de fonctionnelles de complexité basse et très basse. L'ouvrage [22] introduit un système $\mathtt{PV}^\omega$ de termes qui dénotent des fonction(nelle)s de complexité calculatoire polynomiale en temps. Aussi, les systèmes arithmétiques correspondants, intuitionniste $\mathtt{IPV}^\omega$ et classique $\mathtt{CPV}^\omega$, ont été présentés pour permettre le raisonnement formel sur les propriétés de ce genre de termes. En plus, des procédures d'extraction de programmes par à la fois la realisabilité modifiée et l'interprétation Dialectica ont été exposées pour le système $\mathtt{IPV}^\omega$.

En conséquence, les programmes extraits de cette manière à partir de preuves dans (des extensions de) $\mathtt{IPV}^\omega$ dénotent des fonction(nelle)s calculables en temps polynomial et donc efficaces.

Ferreira introduit dans [32] un système BTFA d'*Analyse Efficace* pour lequel les fonctions Skolem correspondantes aux énoncés de forme Π^0_2 dénotent des fonctions calculables en temps polynomial. BTFA est une théorie de second ordre et la méthode utilisée par Ferreira pour la lecture des fonctions de Skolem vient de la théorie des modèles. Plus récemment, une liaison entre les systèmes de Cook et Urquhart d'un coté et le système de Ferreira vient d'être faite par Oliva dans [95]. La section 3.2 du [95] montre comment BTFA peut être simulée presque entièrement dans le système $\mathtt{CPV}^\omega + \mathtt{AxAC}_0$ (voir la Section 2.3 de cette thèse pour l'Axiome du Choix restreint aux formules sans quantificateurs, dénoté comme d'habitude par $\mathtt{AxAC}_0$). En principe, ce n'est que la version la plus générale du *Principe de Ramassage Borné* $\Sigma^b_\infty - \mathtt{BC}$ ("Bounded collection Principle", en anglais), présent dans BTFA qui n'est pas démontrable dans $\mathtt{CPV}^\omega + \mathtt{AxAC}_0$. Seulement une variante plus faible du $\Sigma^b_\infty - \mathtt{BC}$ est démontrée être prouvable dans $\mathtt{CPV}^\omega + \mathtt{AxAC}_0$, voir [95]. Soit $\Pi^0_1 - \mathtt{WKL}$ le schéma axiomatique du Lemme faible de König pour des arbres définis par des formules de type Π^0_1 de $\mathtt{CPV}^\omega$. Le résultat principal de [95] est que, à partir des preuves $\mathtt{CPV}^\omega + \mathtt{AxAC}_0 + \Pi^0_1 - \mathtt{WKL} \vdash \forall x \exists y A_0(x, y)$, on peut extraire par des moyens purement syntaxiques des termes de $\mathtt{PV}^\omega + \mathcal{B}$ (ici $\mathcal{B}$ est une constante spéciale, voir la suite) qui dénotent des fonctions calculables en temps polynomial h telles que $A_0(x, hx)$ est vraie dans le modèle standard. Ce résultat est formulé comme le corollaire 5.2 dans [95]. L'extraction syntaxique est effectuée par une traduction en double négation suivie par une extension de l'interprétation Dialectica de Cook et Urquhart pour le système $\mathtt{IPV}^\omega$. La fonction h est dénotée par un terme du $\mathtt{PV}^\omega$ étendu par une forme binaire plus simple de la simplification de Howard [59] de la bar-récurrence de Spector [119], introduite en [95] comme une nouvelle constante dénotée par $\mathcal{B}$. La fonctionnelle dénotée par $\mathcal{B}$ n'est pas exprimable comme un terme du $\mathtt{PV}^\omega$, car elle représente une certaine forme de recherche non-bornée. Néanmoins, $\mathcal{B}$ est interprétable dans la structure de types $\mathcal{C}$ de Scarpellini [108] qui contient toutes les fonctionnelles continues de la théorie des ensembles et dans laquelle $\mathcal{B}$ peut être éliminée en l'exprimant en fonction de la récurrence limitée sur la notation. L'exécution de la fonction de type Skolem de complexité polynomiale en temps h n'est donc pas entièrement syntaxique par la méthode de [95]. Une technique d'extraction complètement syntaxique a été ensuite développée

par Ferreira et Oliva dans [33, 34] de manière que pas seulement la lecture, mais aussi l'exécution des fonctions de Skolem devienne strictement algorithmique. D'abord, on extrait des bornes pour les réalisants potentiels [33], mais avec une preuve vérificatrice exprimée dans le $\mathtt{IPV}^\omega$ pur. Des réalisants exacts (pour la spécification d'entrée originale) peuvent alors être produits à partir de cette preuve dans $\mathtt{IPV}^\omega$ par le moyen de la technique originelle de Cook et Urquhart, voir [34].

En parallèle un système arithmétique G_2A^ω a été imaginé par Kohlenbach [69] comme membre d'une hiérarchie complète des "Arithmétiques Grzegorczyk" $(G_nA^\omega)_{n\in\mathbb{N}}$ qui ont pour propriété caractéristique que les majorants et bornes extraites par la Dialectica monotone sont réductibles syntaxiquement vers des polynômes aux coefficients entiers et dénotent donc, eux aussi, des fonctions calculables en temps polynomial. En plus, une "Analyse héréditairement bornée par polynômes" `PBA` est bâtie au-dessus de G_2A^ω et une comparaison avec le système `BTFA` de Ferreira est donnée dans [72]. Le système `PBA` est défini comme $G_2A^\omega + \mathtt{AxAC}_0 + \Delta$, où Δ sont des ensembles d'axiomes analytiques de forme logique $\forall x^\delta \exists y \leq_\rho sx \forall z^\tau B_0(x,y,z)$ (tout comme dans la Définition 1.67, mais avec la racine - purement `ncm` - réduite à une formule sans quantificateurs) qui couvrent des sous-ensembles importants de l'Analyse classique (voir [69, 70, 72]). Il s'avère qu'une bonne partie de l'Analyse peut être formalisée dans `PBA` (voir [72]), mais par contre beaucoup moins avait pu être formalisé dans `BTFA` (voir, e.g., [31]). Néanmoins, les deux structures-cadre sont incomparables en général, voir [72].

Vers la fin du Chapitre 3 une nouvelle structure-cadre syncrétique est proposée, en combinant les deux structures de Cook-Urquhart-Ferreira-Oliva et respectivement Kohlenbach. L'avantage du nouveau système d'"Arithmétique et Analyse efficaces" qu'on appellera "bornées en temps polynomial" est que des programmes généralement calculables en temps polynomial peuvent maintenant être extraits par l'interprétation Dialectica monotone légère et donc pas seulement des programmes-polynômes, comme c'était avant le cas pour la `PBA` de Kohlenbach. La structure que nous proposons est donc (au moins théoriquement) plus expressive que celle de Kohlenbach, en ajoutant des éléments de la structure de Cook et Urquhart.

Le Chapitre 4

Le quatrième Chapitre présente une comparaison extensive, à la fois pratique et théorique, de la Dialectica légère avec l'autre technique bien-établie

(dans le domaine des interprétations de preuves) pour l'extraction de programmes à partir de preuves (semi-)classiques, c'est-à-dire la `A`-traduction raffinée de Berger, Buchholz et Schwichtenberg (`BBS`). Cette dernière est en effet un outil pour la traduction des preuves dans la Logique Minimale de l'existence faible vers des preuves intuitionnistes de l'existence forte correspondante. La realisabilité modifiée de Kreisel [85] est ensuite employée pour la lecture d'un réalisant concret pour l'existentiel fort. La comparaison des deux interprétations de preuves est donnée à la fois en théorie mais aussi dans la pratique. Pas moins de quatre études de cas sont traitées sur l'ordinateur, dans le système `MinLog` de Schwichtenberg [49, 116]. Celles-ci sont l'exemple "*hsh*" de Berger, la preuve de l'existence classique des nombres de Fibonacci, l'exemple de la racine carrée et finalement le Lemme de Dickson-2-2. À la fois en théorie et en pratique, la Dialectica légère se montre au moins assez efficace que la `A`-traduction raffinée de `BBS`. En fait, la Dialectica légère est meilleure pour *hsh* et beaucoup meilleure pour la racine carrée. Par contre, dans l'exemple du Lemme de Dickson-2-2, la pure Dialectica de Gödel a une beaucoup pire performance que la technique de Berger, Buchholz et Schwichtenberg. Malheureusement, l'optimisation "légère" (c'est-à-dire le sujet central de cette thèse) ne s'applique pas dans ce cas. Nous mentionnons aussi brièvement d'autres méthodes pour la synthèse de programmes à partir de preuves classiques et nous tentons une comparaison avec la Dialectica légère, particulièrement pour le cas du Lemme de Dickson.

Les techniques de synthèse de programmes du cadre de la théorie de la preuve peuvent être classifiées dans deux larges catégories: d'un côté on a celles basées sur l'élimination des coupures et de l'autre côté celles basées sur des interprétations de preuves. Malgré la complexité hyper-exponentielle (dans le pire cas) de l'élimination des coupures, les interprétations de preuves donnent des algorithmes d'extraction efficaces dans la pratique (voir [56]), en principe à cause de leur modularité. La normalisation des termes (qui sont nos programmes) extraits par les interprétations de preuves égaliserait néanmoins la complexité calculatoire globale (dans le pire cas) des deux classes de techniques. La séparation entre l'extraction de programmes et l'élimination des coupures (qui correspond à la normalisation) par le moyen des interprétations de preuves se montre plus utile dans la pratique, car la normalisation des programmes extraits n'est pas forcément requise dans toutes les applications. Il existe des situations importantes où la normalisation n'intervient du tout, voir [63]. Aussi, des améliorations variées du processus de normalisation peu-

vent être apportées pendant le roulement de l'extraction, voir [53] pur un tel exemple.

Parmi les interprétations de preuves on distingue la realisabilité modifiée de Kreisel [85] et l'interprétation fonctionnelle de Gödel [45, 4]. Les deux ne diffèrent que dans le traitement de l'implication logique. Contrairement à la realisabilité modifiée, la technique de Gödel prend en compte aussi les contre-exemples et s'étend donc plus directement aux preuves classiques (arithmétiques) par l'intermède des traductions négatives. Cette extension peut aller encore plus loin, vers des preuves (classiques) analytiques, voir [68]. Des nouveaux théorèmes remarquables de l'Analyse numérique ont été obtenus récemment par le moyen de (la variante monotone de Kohlenbach) de l'interprétation Dialectica de Gödel, voir [81, 63] pour des présentations d'ensemble à ce sujet. Dissemblablement à la `D`-interprétation, le traitement des preuves non-intuitionnistes par la realisabilité modifiée requiert une `A`-traduction de Friedman-Dragalin comme étape intermédiaire après une traduction en double négation et la combinaison des trois interprétations ne peut manier que des preuves classiques des formules de forme Π^0_2.

Des affinages de la composition des trois traductions ont été proposés dans les années récentes, voir comme exemple [10, 23]. Un but de ces améliorations à été la simplification de la complexité calculatoire des programmes extraits. De l'autre coté on a essayé d'étendre la classe des preuves admissibles à l'entrée des meta-algorithmes d'extraction. La `A`-traduction raffinée de Berger, Buchholz et Schwichtenberg (`BBS`, tout brièvement) a eu du succès dans les deux directions. D'un coté, elle peut interpréter directement des preuves classiques des formules dans une classe qui étend Π^0_2. La complexité des types attachés aux programmes extraits est considérablement réduite par rapport à la combinaison non-raffinée des trois traductions. La technique `BBS` avait été implémentée dans l'assistant de preuves `MinLog` [116] et a été utilisée avec succès pour l'extraction de programmes à partir de preuves classiques, voir [15] comme exemple. Néanmoins, la `BBS`-interprétation ne peut pas manipuler directement toutes les preuves des spécifications Π^0_2 dans le système semi-classique arithmétique `Z` de Berger, Buchholz et Schwichtenberg [10]. Contrairement à la `A`-traduction raffinée de `BBS`, la Dialectica (monotone, légère) peut interpréter directement toutes les preuves $\mathcal{P}$ du système `WeZ` dans le sens qu'elle peut synthétiser des réalisants (ou majorants) pour les quantificateurs existentiels de la *traduction* de la formule conclusion du $\mathcal{P}$. Le système `Z` peut être regardé comme une restriction de l'Arithmétique de Peano dans toutes les

types finis $\mathtt{PA}^\omega$ au langage sans le $\exists$ fort. Il peut être vu aussi comme une surstructure de l'Arithmétique de Heyting dans toutes les types finis $\mathtt{HA}^\omega$ sans $\exists$. La construction du système `Z` et de l'interprétation fonctionnelle en Déduction Naturelle [51] provoquent l'élimination du besoin d'une pré-transformation en double négation pour l'interprétation de ce sous-système de $\mathtt{PA}^\omega$. Néanmoins, le traitement du système $\mathtt{PA}^\omega$ tout entier sous la `(L,M)D`-interprétation requiert une traduction négative intermédiaire, tout comme la technique de Berger, Buchholz et Schwichtenberg. La différence (assez grande!) est que Dialectica arrive à pouvoir manipuler de cette manière *toutes* les preuves de la $\mathtt{PA}^\omega$.

Le Chapitre 5

Le cinquième Chapitre décrit un des exemples les plus complexes qu'on a traité sur l'ordinateur comme étude de cas pour la synthèse de programmes par la nouvelle interprétation Dialectica monotone légère. En principe, on extrait au cadre de l'assistant de preuves `MinLog` [49, 116] quelques modules de continuité uniforme pour des termes concrets du système $\mathbf{T}$ de Gödel de type $(\mathbb{N}\to\mathbb{N})\to(\mathbb{N}\to\mathbb{N})$. Ceux-ci sont en fait les premiers trois éléments d'une suite de termes génériques. L'humain peut alors inférer immédiatement une solution complète pour tout élément de cette suite à partir de la sortie limitée de l'ordinateur. La nouveauté ici est qu'on a utilisé une certaine variante de la Dialectica monotone légère qui produit des termes dans une forme normale (par rapport à la Normalisation par Évaluation) par le moyen d'une technique récurrente partielle de normalisation par évaluation. Cette forme d'évaluation partielle a été strictement nécessaire pour l'obtention des résultats pratiques sur l'ordinateur dans un laps de temps raisonnable, qui est en fait très petit (moins d'une minute pour le cas le plus complexe). La synthèse automatique de ces modules de continuité uniforme s'effectue à partir d'une preuve de l'égalité héréditairement extensionnelle ($\approx$) du terme $\mathbf{t}$ à soi-même, donc une preuve dans une variante faiblement extensionnelle du système Z de Berger, Buchholz et Schwichtenberg de l'énoncé $\mathbf{t}\approx_{(\mathbb{N}\to\mathbb{N})\to(\mathbb{N}\to\mathbb{N})}\mathbf{t}$. Nous utilisons une implémentation sur l'ordinateur, dans le système `MinLog` de Schwichtenberg, de l'interprétation Dialectica monotone légère, introduite dans la Section 2.2. Ne prenant pas en compte les quantificateurs sans signification calculatoire, celle-ci est une adaptation non-littérale de l'interprétation fonctionnelle monotone de Kohlenbach. En plus, nous utilisons ici une certaine variante de la `LMD`-interprétation qui produit des termes en forme `NbE`-normale par le moyen d'une Normalisation par Évaluation récurrente partielle. Quoique les quantifi-

cateurs sans signification computationnelle (`ncm`) n'ont pas été employés pour les résultats pratiques exposés dans ce chapitre, nous croyons que leur utilisation pourrait réduire le temps de roulement du méta-algorithme d'extraction monotone dans ce cas. Néanmoins, ils n'auront aucune influence sur le programme extrait en forme normale que nous obtenons de tout façon dans un intervalle de temps raisonnable - moins d'une minute.

Le Chapitre A

Dans le Chapitre A de l'Annexe nous présentons une analyse quantitative de l'interprétation fonctionnelle de Gödel et de sa variante monotone. Nous obtenons des bornes supérieures sur la profondeur, la taille, le degré maximal et l'arité maximale d'un type pour les termes extraits, ainsi que sur la profondeur de la preuve vérificatrice, toutes comme fonctions de mesures quantitatives de base sur les preuves à l'entrée. Des termes de taille linéaire dans la taille de la preuve à l'entrée peuvent être extraits dans tous les cas et les méta-algorithmes d'extraction correspondants ont une complexité calculatoire cubique dans le pire cas. Les preuves vérificatrices ont une profondeur linéaire dans la profondeur de la preuve d'entrée et dans la taille maximale d'une formule de cette preuve. Quoique ces résultats ont été obtenus pour des systèmes de type Hilbert et combinatorielles, ils peuvent être immédiatement adaptés au type de systèmes que nous utilisons partout dans le corps principal de cette thèse, c'est-à-dire des système de Déduction Naturelle avec la lambda-abstraction comme primitive.

Ce chapitre étudie la complexité calculatoire des meta-algorithmes d'extraction de données effectives (telles que des programmes et des bornes) à partir des preuves. Ces algorithmes sont fournies par l'interprétation fonctionnelle "Dialectica" de Gödel et par sa variante monotone due à Kohlenbach. Les deux techniques ont été utilisées avec succès dans la synthèse des programmes et des bornes numériques, tout comme pour l'obtention de résultats de conservation. Elles s'appliquent à la fois aux preuves (semi-)intuitionnistes et (en combinaison avec des traductions négatives) aux preuves pleinement classiques. Les preuves d'entrée peuvent être formalisées dans des systèmes qui s'étendent à partir des systèmes faibles de base et jusqu'à l'Arithmétique et l'Analyse entières (et dans des nombreuses de leurs sous-systèmes).

Le sujet de l'extraction de programmes à partir des preuves a déjà une longue histoire. Les techniques d'extraction peuvent être classées à peu près

dans deux grandes catégories, selon si elles sont basées sur l'élimination des coupures, la normalisation ou d'autres méthodes liées à ces deux, ou si elles utilisent ce qu'on appelle "des interprétations de preuves". Les dernières utilisent de manière caractéristique des fonctionnelles avec des types de plus-haut-ordre. Telles interprétations de preuves proéminentes sont les techniques de Realisabilité, en particulier la realisabilité modifiée de Kreisel [86] (voir [122] pour une vue d'ensemble) et l'interprétation Dialectica de Gödel (publiée pour la première fois dans [45], en allemand - l'ouvrage [4] contient une très bonne vue d'ensemble, en anglais). La "non-contre-exemple interprétation" (*n.c.i.*, de l'anglais "no-counterexample interpretation") due elle-aussi à Kreisel [83, 84] est parfois regardée comme une simplification de l'interprétation Dialectica, car elle n'utilise que des types de degré ≤ 2. En fait, la *n.c.i.* ne constitue pas une alternative réelle à Dialectica, car elle a des difficultés à interpréter la règle de l'élimination de l'implication (Modus Ponens, abrégée `MP`). Ce mauvais comportement ne peut être surmonté que par l'utilisation de l'interprétation Dialectica (voir [73] pour une explication complète).

L'élimination des coupures, la normalisation et la méthode liée dénommée "ε-substitution" rebâtissent de façon globale la preuve d'entrée, ce qui incrémente sa profondeur d'une manière potentiellement non-élémentairement récurrente. Des exemples de bornes inférieures hyper-exponentielles ont été donnés par Statman [120], Orevkov [99, 100] et Pudlak [105] - voir aussi [124] et les ouvrages plus récentes à ce sujet de Gerhardy [39, 37, 38]. Au contraire, les interprétations des preuves synthétisent les termes-témoins par réccurence sur l'arbre de la preuve d'entrée, dont la structure reste essentiellement inchangée pendant tout le processus d'extraction. Ces techniques présentent donc une modularité complète: les réalisants globaux d'une preuve peuvent être calculés à partir des réalisants des lemmes employés dans la preuve. Ceci suggère une complexité calculatoire radicalement plus basse des interprétations des preuves et en conséquence une taille radicalement plus petite des programmes extraits. Même si ces derniers ne seraient pas dans une forme normale, ils peuvent néanmoins être utilisés de manière substantielle dans plusieurs façons sans avoir à les normaliser. Il suffit d'exploiter des propriétés qui peuvent être établies inductivement sur leur structure en utilisant des relations logiques, telles que la majorisabilité de Howard [58].

Plusieurs résultats pratiques concrèts ont été obtenus récemment dans l'Analyse Numérique en appliquant l'interprétation fonctionnelle monotone

de Kohlenbach, voir [63, 81] pour des vues d'ensemble. La question naturelle que se pose est si ce genre d'applications qu'ont été obtenues manuellement pourraient être automatisées ou au moins assistées par l'ordinateur en implémentant les interprétations fonctionnelles. Afin d'évaluer la faisabilité de ce genre d'outils automatiques il est important d'analyser les aspects de la complexité calculatoire des interprétations Dialectica. Dans ce chapitre nous obtenons des bornes supérieures sur la taille des termes qui expriment les programmes extraits. Les méta-algorithmes de traduction ne font qu'écrire les termes extraits, roulant de manière récurrente par rapport à la structure de la preuve d'entrée, voir la section A.2. En conséquence, leur temps d'exécution est proportionnel avec la taille des termes extraits. Nous obtenons donc la complexité calculatoire temporelle des algorithmes d'extraction dans la suite de notre analyse quantitative. Soit n la taille de la preuve d'entrée (dénotée $\mathcal{P}$) et soit m la taille maximale d'une formule de la preuve $\mathcal{P}$. À cause de la modularité des interprétations fonctionnelles, leurs algorithmes ont une complexité calculatoire presque linéaire, c'est-à-dire $O(m^2 \cdot n)$, même quand ils ont à traiter des preuves classiques et analytiques. Le mot *presque* est utilisé car m est généralement beaucoup plus petit que n dans la plupart des applications pratiques. En tout cas, cette complexité temporelle est $O(n^3)$ dans le pire cas, un résultat obtenu antérieurement par Alexi dans [1] pour une technique d'extraction ad-hoc dans le cas des preuves intuitionnistes exclusivement. Comme Alexi avait conçu sa technique par souci des coûts optimales, *cubique* doit être la meilleure complexité du pire cas qu'on peut attendre de toute méthode d'extraction de programmes à partir des preuves. Nous donnons aussi des bornes supérieures sur la profondeur des preuves vérificatrices. Celles-ci peuvent être utilisées pour l'obtention des résultats de conservation quantitative. Nous obtenons en particulier, par une méthode de traduction purement syntaxique, l'efficacité de l'élimination du Lemme Faible de König pour des énoncés de forme Π^0_2 sur l'Arithmétique Primitivement Récurrente dans tous les types finis.

Optimized programs from (non-constructive) proofs by the light (monotone) Dialectica interpretation

(revised version of doctoral thesis)

Mircea–Dan Hernest

14 March 2007

Abstract

This thesis presents a new optimization of Gödel's Dialectica interpretation for the extraction of more efficient exact realizers from (classical) arithmetical proofs. The "light" variant of Dialectica also combines and even more smoothly with Kohlenbach's "monotone" optimization of Gödel's functional interpretation for the extraction of more efficient majorants and bounds from (classical) monotonic arithmetical and even analytical proofs. "Light Dialectica" is obtained by adapting Berger's "uniform" or "non-computational" quantifiers. Moreover, its presentation is given in Natural Deduction style, as an improvement of Jørgensen's recent adaptation of pure Gödel's Dialectica. A number of concrete examples are treated on the computer by means of the novel technique. The machine comparison with the more established program-synthesis technique of refined `A`-translation shows a very good performance of Light Dialectica, which is outperformed only in the case of Dickson's Lemma. Also the theory of synthesis of feasible, poly-time computable programs is developed for the new "Light Monotone Dialectica" extraction technique. Two pre-existent frameworks due to Cook-Urquhart-Ferreira and respectively Kohlenbach are crossbreeded for this purpose into a "poly-time bounded Analysis". The theoretical result is promising, yet practical examples are to be found for the difference with the pure Kohlenbach's "polynomially bounded Analysis".

Keywords: Program extraction from (classical) proofs, Proof Mining, complexity of extracted programs, Gödel's functional *Dialectica* interpretation, non-computational-meaning (uniform) existential and universal quantifiers, proof complexity, functionals of finite type, software and systems verification, combinatorial logic, computational complexity, proof-carrying code.

Contents

Acknowledgements

In the first place I would like to thank my thesis co-directors, Prof. Jean-Pierre Jouannaud and Prof. Helmut Schwichtenberg for having accepted to supervise my work on the original topic of adapting the non-computational quantifiers to the Dialectica Interpretation in a Natural Deduction setting. Thanks also to Prof. Ulrich Kohlenbach for having advised me throughout the whole period of my doctoral studies. I am grateful to Ulrich Berger and to Michel Parigot for having accepted to referee this thesis and also to Prof. Christine Paulin for her participation and presiding of my examination board.

A few of my colleagues contributed indirectly to my work by proof-reading and discussions: Philipp Gerhardy, Laurenţiu Leuştean, Paulo Oliva, Monika Seisenberger and Daniele Varacca. There is a long list of people to whom I am indebted concerning my formation. This list includes teachers from high-school in Slatina, college in Bucharest (Romania) and graduate schools in Aarhus (Denmark), Munich (Germany) and Paris (France). I here mention only a few of them: Prof. Olivier Danvy, Prof. Gilles Dowek, Prof. George Georgescu, Prof. Gheorghe Ştefănescu and Prof. Marin Toloşi.

I owe a deep debt of gratitude for having provided me a wonderful work environment to the scientific and administrative collectives from the Chair of Fundamentals of Computer Science of the University of Bucharest (Romania), from BRICS/DAIMI in Aarhus (Denmark), from GKLI/the Munich Logic group (Germany) and from the LogiCal group at LIX-Ecole Polytechnique in Paris (France). For a supporting attitude I would like to thank in particular to the following persons: Luca Castelli Aleardi, Janne Kroun Christensen, Laura Crosilla, Daniel Damian, Claudiu Dobre, Felix Joachimski, Maciej Koprowski, Gyesik Lee, Pierre Letouzey, Ioana Leuştean, Ian Mackie, Catherine Moreau, Karen Kjaer Möller, Julien Narboux, Mogens Nielsen, Diana Raţiu, George Rodolakis, Grigore Roşu, Jiri Srba, Frank Valencia, Luminiţa Vîţâ, Bogdan Warinschi and Benjamin Werner.

Last but not least, I hereby thank to my family members for their understanding and support during the adventure of my doctoral studies.

Thanks to all who helped me directly or indirectly!

Mircea-Dan Hernest

Introduction

[1] Important practical results[2] have been obtained in recent years in the field of extractive Proof Theory (also dubbed *proof mining* [81]). The implemented algorithms coming from metamathematical research have yielded interesting and in many cases quite unexpected programs (see, e.g., [10, 15, 50, 55, 104, 106]). Various approaches to program extraction from *classical*[3] proofs have been developed over years of research [3, 5, 10, 20, 23, 68, 87, 88, 91, 93, 101, 102, 106]. These can be roughly divided into two main groups inside which one may distinguish many more sub-groups. On one hand we have the historically more recent line of research coming more from theoretical computer science, which aims at extending the Curry-Howard correspondence to full classical logic [5, 20, 87, 102]. This research direction originates in the attempt of giving algorithmic interpretations to various classical principles, since Griffin [46] noticed that Felleisen's $\mathcal{C}$ operator [29, 30] can be typed as the stability scheme $\neg\neg A \to A$. One recent impressive practical result in this line was obtained by Raffalli [106], which extracted an amazingly (keeping the relative proportions) fast algorithm for Dickson's Lemma in its most general form by means of a so-called "Mixed Logic".

On the other hand we have the more traditional proof-theoretic line of research on so-called "Proof Interpretations", see [63] for many nicely organized and commented historical references. The use of proof interpretations instead of the direct application of cut-elimination has opened the path to the obtaining of better practical results by means of such modular techniques, see [50] for a discussion on this issue. Gödel's functional "Dialectica" interpreta-

[1] The first two chapters of this thesis constitute a large extension of the work reported in [51], including its technical appendix [52].

[2] Here by *practical* we understand both concrete new mathematical proofs like the ones presented in [63, 81] but also new computer algorithms like those reported in [10, 15, 50, 55, 104, 106].

[3] *Constructive* proofs have a more or less explicit computational content, see, e.g., [7].

tion [45, 4] directly applies to (far) more complex proofs than its simpler, but weaker form, Kreisel's Modified Realizability [85]. The latter's combination with refinements [3, 10, 23] of Friedman's A-translation[4] [36] only partly repairs this disparity (see [50] or [63] for discussions on this issue). However, the (rough) exact realizers yielded by the Dialectica interpretation are generally (far) more complex than those produced by, e.g., the technique of Berger, Buchholz and Schwichtenberg [10].

The main reason for such a situation seems to be the inclusion of the *Dialectica-relevant* contraction formulas in the rough (i.e., not yet normalized) Dialectica realizing terms. Even though in some cases the primary realizers extracted via the two techniques normalize to basically the same programs (see [50] for such an example), the normalization of programs extracted by proof interpretations is generally far more expensive than their synthesis[5]. The simpler the rough extracted term is, the lower the overall cost of producing the final normalized realizer becomes. In order to handle this contraction problem of the Dialectica interpretation, Kohlenbach [68] devised the *monotone* functional interpretation, an adaptation of Gödel's technique to the extraction of majorants/uniform bounds for the exact realizers (see also [33] for the more recent *bounded* functional interpretation of Ferreira and Oliva). While so prolific (see [63] or [81]) in the context of mathematical Analysis, Kohlenbach's monotone functional interpretation, used literally and alone, is generally not as practically useful for the synthesis of exact realizers in discrete mathematics. Except for some situations when the input specification is monotonic in the variable to be realized, the production of an exact realizer, given a majorant, requires a bounded search and is possible only in some cases. See the end of Section 4 in [56] for an extended comment or [63] for more complete (and technical) details.

We propose in this thesis a different kind of optimization of Gödel's functional interpretation by the elimination already from the primary extracted terms (i.e., during the extraction process) of a number of *Dialectica-relevant* contraction formulas which are identified as "computationally redundant" by means of an adaptation of Berger's *uniform* quantifiers from [9] to the context of Dialectica interpretation. These are called "quantifiers without computa-

[4] Such *refined* A*-translations* usually combine some double-negation translation from classical to intuitionistic proofs with the so-called Friedman-Dragalin A-translation, see [26] for the Russian (independently discovered) variant of the latter.

[5] See the appendix Chapter A for a detailed theoretical exposition on the (very low) computational complexity of the Dialectica extraction meta-algorithms.

tional content" in [111] and, following [51], we will here call them quantifiers *without computational meaning* (abbreviated `ncm`, with `n` from *non*).

We here name *Light*[6] *Dialectica* (or "light functional interpretation", like in [51]) this optimized exact realizer extraction technique which is based on Gödel's Dialectica interpretation. We also name *Light Monotone Dialectica*, or "light monotone functional interpretation", like in [55], its monotone (in the sense of Kohlenbach [68]) counterpart, which is an optimized technique for the extraction of majorants/uniform bounds.

Following [51], we will denote in this thesis by $\overline{\forall}$ the `ncm` universal quantifier and by $\overline{\exists}$ the `ncm` existential quantifier. While our $\overline{\exists}$ is identical[7] to Berger's $\{\exists\}$, our $\overline{\forall}$ requires a further strengthening of the restriction set by Berger on his $\{\forall\}^+$ introduction rule in [9], see [51]. This is because we must take into account the inclusion of the so-called "computationally persistent" contractions into the Light Dialectica realizing terms. Since these are no longer included into the Light Monotone Dialectica majorizing terms, in the monotone setting both $\overline{\exists}$ and $\overline{\forall}$ are identical to Berger's. Also for the monotone systems there will be no restriction on the Implication Introduction rule. This has to be added to the non-monotonic systems in order to ensure that in a persistent-contraction situation the introduction formula can effectively be included into the realizing terms. Thus the "monotone" and the "light" optimizations of Dialectica will combine much better in a much simpler arithmetical system which can be more easily extended to full classical proofs, as usual. On the contrary, the `ncm` non-monotonic systems will not extend to full classical proofs unless we eliminate from them the strong existential quantifiers, and even then such an extension is far more technically complicated.

We generally abbreviate (both regular and `ncm`) *quantifier free* by `qfr`. We will use expressions like "`qfr` formula", "`ncm` variable" or "`ncm` quantifiers" with the obvious meanings.

Outline of the following sections

In Chapter 1 we introduce all logical, arithmetical and analytical systems which will be used throughout the whole thesis, except for the so-called *feasible* systems of Arithmetic and Analysis from Chapter 3. The latter will nevertheless be presented in a framework which immediately adapts the one given in

[6] Where "light" is to be understood as the opposite of "heavy".

[7] Modulo the formulation as axioms like in [111] of Berger's rules for $\{\exists\}$ from [9].

Chapter 1 to the various feasible settings. The reader should be cautious that a great number of intermediate logical systems are here introduced in order to clarify which logical principles are strictly necessary to prove certain particular principles. Their use is limited to Chapter 1 only, hence the reader may want to skip the detail at a first reading and come back by need, during the reading of the core Chapter 2. The Index section placed after the Bibliography may get very useful for this purpose.

Chapter 2 presents our Natural Deduction formulations of the "light" refinements of both Gödel's functional interpretation and Kohlenbach's Monotone Dialectica. For the design of Dialectica Light we transformed Jørgensen's Natural Deduction formulation [61] of the pure Gödel Dialectica translation by eliminating his "Contraction Lemma" [8] and by allowing free variables in the extracted terms[9]. We also adapted Berger's uniform existential and universal quantifiers from [9] to the Dialectica-extraction context. The use of such quantifiers *without computational meaning* permits the identification and isolation of contraction formulas which would otherwise become redundantly included in the pure-Dialectica extracted terms. We also give the combination of our simplification of Gödel's Dialectica interpretation with its adaptation to the extraction of majorants/uniform bounds due to Kohlenbach into a *light monotone* functional interpretation. In the end, we extend to classical proofs.

In Chapter 3 we give an exposition of two well-established frameworks for specifying poly-time computable programs, together with the adaptation of the corresponding arithmetical systems to the Dialectica-extraction of such feasible programs from proofs. We thus have Cook and Urquhart's "poly-time Arithmetic" [22] and Kohlenbach's "polynomially bounded Analysis" (abbreviated `PBA`), both presented as adaptations of the arithmetical systems exposed in Chapter 1. Light Dialectica interpretations are designed for both frameworks as adaptations of the proof interpretations defined in Chapter 2. In the end, a syncretic framework is proposed, which combines the two. The advantage of our proposed system of "feasible Arithmetic and Analysis" is that generally poly-time computable programs are allowed to be extracted by the (light) monotone Dialectica, and not just polynomials (as is the case for Kohlenbach's `PBA`). Hence our framework is more expressive than Kohlenbach's, by adding the elements due to Cook and Urquhart.

[8] Which we found too complicated, particularly for the practical purpose of a computer-implemented (light, monotone) Dialectica extraction.

[9] Which is more suitable in a Natural Deduction setting, see [57] for a discussion on this.

Chapter 4 presents an extensive comparison of (light) Dialectica with the other well-established proof-interpretational technique for extracting programs from (semi-)classical proofs, namely the refined A-translation due to Berger, Buchholz and Schwichtenberg. The latter is actually a tool for translating minimal logic proofs of weak existence into intuitionistic proofs of strong existence. Modified Realizability is then used for reading the concrete realizer. The comparison is given both in theory and in practice. Not less than four case-studies are performed on the computer, in Schwichtenberg's MinLog system. These are Berger's *hsh*, the classical Fibonacci and the square-root example. In both theory and practice, Dialectica Light appears to be at least as good as the BBS A-translation, actually even better for Berger's *hsh*, and much better for the square-root. In contrast, for the Dickson-2-2 Lemma example, Gödel's Dialectica has a much worse performance than the BBS technique and unfortunately the "light" optimization (which is the central topic of this thesis) does not apply in this case. We also mention other methods for program synthesis from classical proofs and outline a tentative comparison with Light Dialectica, particularly in the afore-described case of Dickson's Lemma.

The Chapter 5 outlines one of the most complex examples which we treated on the computer as a case-study for program-extraction by the new Light Monotone Dialectica interpretation. One basically extracts in the proof-system MinLog [49, 116] some moduli of uniform continuity for concrete terms of Gödel's $\mathbf{T}$ of type $(\mathbb{N}\to\mathbb{N})\to(\mathbb{N}\to\mathbb{N})$. These are just the first three elements in a sequence of terms, for which the generic solution can then be immediately inferred by the human from the limited machine output. The novelty here is that we used a certain variant of the Light Monotone Dialectica which produces terms in NbE-normal form by means of a recurrent partial NbE-normalization. This form of partial evaluation was strictly necessary in order to obtain the machine results in due time.

In the appendix Chapter A we give a quantitative analysis of Gödel's functional interpretation and its monotone variant. 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. Although these results were obtained for Hilbert-style and combinatorial systems, they immediately

adapt to the kind of systems which we use throughout the main body of this thesis, i.e. Natural Deduction systems with lambda-abstraction as primitive.

Last, but not least, the source code of our computer implementation of the Light (Monotone) Dialectica, together with all the examples treated mostly in comparison with the `BBS` refined `A`-translation is public available for download at `http://www.brics.dk/~danher/MinLogForDialectica` . Enjoy! :)

Chapter 1

Arithmetical systems for Gödel functionals

[1] We devise a weakly extensional variant $\mathtt{WeZ}$ of the intuitionistic arithmetical system $\mathtt{Z}$ of Berger, Buchholz and Schwichtenberg from [10]. Relative to $\mathtt{Z}$, system $\mathtt{WeZ}$ simply restricts extensionality of equality. For this reason, equality in $\mathtt{WeZ}$ is handled by means of a functional constant (denoted $\mathtt{Eq}$, see Definition 1.1) instead of the more usual predicate constant. Such treatment of equality is peculiar to Gödel's Dialectica interpretation[2]. We also give a suite of extensions of $\mathtt{WeZ}$ basically with the strong (intuitionistic) existential quantifier on one hand and, on the other hand, with the so-called *non-computational-meaning* (abbreviated $\mathtt{ncm}$) variants of the universal and (if present) existential quantifiers, which were mentioned above in the Introduction chapter. We thus have the extensions denoted $\mathtt{WeZ}^{\exists}$, $\mathtt{WeZ}^{\mathtt{nc}}$ and respectively the most complete $\mathtt{WeZ}^{\exists,\mathtt{nc}}$, see Section 1.3. At its turn, system $\mathtt{WeZ}$ is built in stages, as the most complete in the sequence of extensions $\mathtt{ML}_0 \subset \mathtt{ML} \subset \mathtt{IL}_0 \subset \mathtt{IL} \subset \mathtt{WeZ}$. System $\mathtt{ML}_0$ is a pure predicate minimal logic system. System $\mathtt{ML}$ is minimal logic $\mathtt{ML}_0$ enhanced with the axioms for boolean

[1] The reader should beware that a great number of the intermediate logical systems which are introduced in this chapter have a limited local use. Only the more important, arithmetical systems like $\mathtt{WeZ}$, ..., $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ are later used in the core Chapter 2, hence the reader may want to skip part of the technical details of the present chapter. The intermediate, local systems are intended to clarify which logical principles are strictly necessary to prove certain particular purely logical principles. The Index section placed after the Bibliography may get very useful in the reading of both Chapters 1 and 2.

[2] See,e.g., the corresponding sections of [123], [4] or [63] for large and detailed discussions on this issue.

truth and falsity and system $\mathtt{IL}_0$ further adds the so-called "boolean Induction Axiom" and thus contains all axioms governing the boolean atomic formulas. Finally, system $\mathtt{IL}$ is the pure predicate intuitionistic logic which simply adds to $\mathtt{IL}_0$ the equivalence between boolean and logical falsity. Moreover, each of $\mathtt{ML}_0$, $\mathtt{ML}$, $\mathtt{IL}_0$ and $\mathtt{IL}$ has its version with $\exists$ or/and $\mathtt{ncm}$ quantifiers, hence we have the systems:

$$\begin{array}{ccccccccc}
\mathtt{ML}_0^{\exists} & \subset & \mathtt{ML}^{\exists} & \subset & \mathtt{IL}_0^{\exists} & \subset & \mathtt{IL}^{\exists} & \subset & \mathtt{WeZ}^{\exists} \\
\mathtt{ML}_0^{\mathtt{nc}} & \subset & \mathtt{ML}^{\mathtt{nc}} & \subset & \mathtt{IL}_0^{\mathtt{nc}} & \subset & \mathtt{IL}^{\mathtt{nc}} & \subset & \mathtt{WeZ}^{\mathtt{nc}} \\
\mathtt{ML}_0^{\exists,\mathtt{nc}} & \subset & \mathtt{ML}^{\exists,\mathtt{nc}} & \subset & \mathtt{IL}_0^{\exists,\mathtt{nc}} & \subset & \mathtt{IL}^{\exists,\mathtt{nc}} & \subset & \mathtt{WeZ}^{\exists,\mathtt{nc}} \quad .
\end{array}$$

Sections 1.2 and 1.3 in the sequel contain the detailed definitions of all the above systems. The $\mathtt{ncm}$ arithmetics $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ will then be extended in Section 1.4 with both the usual non-intuitionistic principles directly realized by the Dialectica interpretation, here extended to the new $\mathtt{ncm}$ setting but also with new principles. The latter are obtained by replacing the strong $\exists$ with the weak $\exists^{\mathrm{cl}}$ in (certain restrictions of) the usual axioms of Independence of Premises and Axiom of Choice. See Section 1.4 for full details.

We also further device in Section 1.5 the monotonic variants of the above $\mathtt{ncm}$ intuitionistic and extended Arithmetics. The most important quality of these new monotone $\mathtt{ncm}$ Arithmetics $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$, $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$, $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$ and $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+}$ is that the quite drastic restriction on the Implication Introduction rule is no longer necessary, see Remark 1.16. The light Monotone Dialectica interpretation will thus be able to mine more proofs which make use of the $\mathtt{ncm}$ quantifiers, see Theorem 2.12. In the end we extend both monotonic and non-monotonic $\mathtt{ncm}$ arithmetics with full classical logic, in Section 1.6. We will see later in Section 2.3 that the Light Dialectica extraction has big difficulties to extend to a full classical logic context. On the contrary, the Light Monotone Dialectica quickly extends to full classical logic in the usual way, due to the lack of restriction on Implication Introduction.

According to its authors, system $\mathtt{Z}^3$ is an extension of Gödel's $\mathbf{T}$ with the logical and arithmetical apparatus which renders it suitable to the applied program extraction from classical proofs by means of the refined $\mathtt{A}$-translation of [10]. A corresponding statement can be made about the triple of systems $\mathtt{WeZ}^{\exists}$, $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$, relative to the machine extraction by the hereby pre-

[3] Together with its natural companion $\mathtt{Z}^{\exists}$, which simply adds to $\mathtt{Z}$ the axioms defining the strong $\exists$. The intuitionistic $\exists$ is already present in [10], but rather implicitly. It is nevertheless somewhat more explicitly presented and treated in [111].

sented Light Dialectica interpretation. Similarly, systems $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$, $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ appear to be the optimal logical-and-arithmetical extensions of $\mathbf{T}$ in view of majorant and bound extraction by means of the Light Monotone Dialectica (in combination with Kuroda or Gödel-Gentzen negative translations for full classical logic, see Section 2.3).

1.1 Languages, types, terms and formulas

Finite types are inductively generated from base types by the rule that if σ and τ are types then $(\sigma\tau)$ is a type. For simplicity we take as *base* types only the type ι for natural numbers and o for booleans. Types are normally denoted by the symbols $\delta, \gamma, \rho, \sigma$ and τ, which are in principle reserved for such purpose. We make the convention that concatenation is right associative and consequently omit unnecessary parenthesis, writing $\delta\,\sigma\,\tau$ instead of $(\delta(\sigma\tau))$. We denote tuples of types by $\underline{\sigma} :\equiv \sigma_1, \ldots, \sigma_n$. We abbreviate by $\underline{\sigma}\tau$ the type $\sigma_1 \ldots \sigma_n\tau$. It is immediate that every type τ can be written as either $\tau \equiv \underline{\sigma}\iota$ or $\tau \equiv \underline{\sigma}o$.

The *term system*, which we denote by $\mathcal{T}$, is a variant of Gödel's $\mathbf{T}$ formulated over the finite types with lambda-abstraction as primitive. This is most appropriate in a Natural Deduction context. Terms are hence built from variables and term constants by lambda-abstraction and application. We represent the latter as concatenation and we agree that it is left-associative in order to avoid excessive parenthesizing. All variables and constants have an *a priori* fixed type and terms have a type fixed by their formation. Written term expressions are always assumed to be well-formed in the sense that types match in all applications between sub-terms.

Definition 1.1 As particular *term constants* we distinguish the following:

- $\mathtt{tt}^o$ and $\mathtt{ff}^o$ which denote boolean truth and falsity;
- for each type τ the *selector* $\mathtt{If}_\tau$ of type $o\,\tau\,\tau\,\tau$ which denotes choice according to a boolean condition with the usual *if-then-else* semantics;
- $\mathtt{0}^\iota$ (zero), $\mathtt{Suc}^{\iota\iota}$ (successor) and Gödel's recursor $\mathtt{R}_\tau$ of type $\tau\,(\iota\,\tau\,\tau)\,\iota\,\tau$;
- equality $\mathtt{Eq}^{\iota\,\iota\,o}$ - a functional constant and not predicate in our system.

Variables are denoted by $a, b, c, p, q, u, v, x, y, z, U, V, X, Y, Z$ such that, if not otherwise specified, a, b, c are free and u, v, x, y, z are bound variables of

type ι. Here *free* and *bound* are meant w.r.t. the formula quantifiers, introduced later in the sequel. Also p, q denote variables of type o (be them free or bound) and U, V, X, Y, Z are *functional* variables (i.e., not of base type). We denote terms by r, s, t, S, T. Relative to lambda-abstraction we distinguish the sets of *lambda-free* and *lambda-bound* variables of a term.

Convention 1.2 We use sub- or super- scripts to enlarge the classes of symbols. We use underlined letters to denote tuples of corresponding objects. Tuples are just comma-separated lists of objects. If $\underline{t} \equiv t_1, \ldots, t_n$ we denote by $s(\underline{t})$ or even $s\underline{t}$ the term $st_1 \ldots t_n$, i.e., $((st_1)\ldots)t_n$ by the left-associativity convention. Also $\underline{s}(\underline{t})$ and $\underline{s}\,\underline{t}$ denote the tuple $s_1(\underline{t}), \ldots, s_m(\underline{t})$.

Definition 1.3 As particular *terms* we distinguish the following:

- Boolean conjunction, implication and disjunction, which have their usual boolean semantics, all defined in terms of $\mathtt{tt}$, $\mathtt{ff}$ and $\mathtt{If}_o$ only:

$$\mathtt{And}^{ooo} :\equiv \lambda p, q.\, \mathtt{If}_o\, p\, q\, \mathtt{ff} \tag{1.1}$$

$$\mathtt{Imp}^{ooo} :\equiv \lambda p, q.\, \mathtt{If}_o\, p\, q\, \mathtt{tt} \tag{1.2}$$

$$\mathtt{Or}^{ooo} :\equiv \lambda p, q.\, \mathtt{If}_o\, p\, \mathtt{tt}\, q \tag{1.3}$$

- For each higher-order type $\tau \equiv \underline{\sigma}\delta$ with $\delta \in \{\iota, o\}$ we define the *zero* term $\mathtt{O}_\tau$ of type τ to be $\lambda \underline{x}^{\underline{\sigma}}.\, \mathtt{O}_\delta$, where $\mathtt{O}_\iota :\equiv \mathtt{O}$ and $\mathtt{O}_o :\equiv \mathtt{ff}$. Hence every type is inhabited by a zero term.

- For each positive integer n and type τ we define the *n-selector* $\mathtt{If}^n_\tau$ of type $\overbrace{o \ldots o}^{n}\,\overbrace{\tau \ldots \tau}^{n}\,\tau\,\tau$ by $\mathtt{If}^1_\tau :\equiv \mathtt{If}_\tau$ and for $n \geq 2$,

$$\begin{aligned}\mathtt{If}^n_\tau \;:\equiv\; & \lambda p_1, \ldots, p_n, x_{n+1}, x_n, \ldots, x_1. \\ & \mathtt{If}_\tau\, p_1\, (\mathtt{If}^{n-1}_\tau\, p_2\, \ldots\, p_n\, x_{n+1}\, x_n\, \ldots\, x_2)\, x_1\end{aligned} \tag{1.4}$$

  hence generally $\mathtt{If}^n_\tau(r_1, \ldots, r_n, t_{n+1}, t_n, \ldots, t_1)$ selects the first t_i with $i \in \overline{1, n}$ for which r_i is false, if it exists, otherwise (if all $\{r_i\}_{i=1}^n$ are true) it selects t_{n+1}.

Our base logical systems $\mathtt{IL}$, $\mathtt{IL}^\exists$, $\mathtt{IL}^{\mathtt{nc}}$ and $\mathtt{IL}^{\exists,\mathtt{nc}}$ will be Natural Deduction formulations of Intuitionistic Logic which are inspired mostly by [10], [111] and

also by the more recent paper [9], see Section 1.2 below. We will also distinguish[4] the Minimal Logic[5] subsystems $\mathtt{ML}_0/\mathtt{ML}_0^{\exists}/\mathtt{ML}_0^{\mathtt{nc}}/\mathtt{ML}_0^{\exists,\mathtt{nc}}$, $\mathtt{ML}/\mathtt{ML}^{\exists}/\mathtt{ML}^{\mathtt{nc}}/\mathtt{ML}^{\exists,\mathtt{nc}}$ and $\mathtt{IL}_0/\mathtt{IL}_0^{\exists}/\mathtt{IL}_0^{\mathtt{nc}}/\mathtt{IL}_0^{\exists,\mathtt{nc}}$. Details are in Definitions 1.20, 1.24 and finally 1.26.

We will therefore use $\wedge$ (logical conjunction), $\to$ (logical implication), $\forall$ (forall) and *optionally* $\exists$ (strong, "intuitionistic" exists) as base logical constants. The only predicate symbols are the unary $\mathtt{at}$, which takes a single boolean argument, and the zeroary $\bot$ which denotes *logical falsity*. If t^o is a boolean term then $\mathtt{at}(t)$ is the *atomic formula* which (informally) denotes the fact that t is true. *Prime formulas* are the atomic formulas $\{\mathtt{at}(t)\,|\,t \text{ term of type } o\}$ plus the logical falsity $\bot$. The prime formula $\bot$ is not a priori equivalent to the so-called *boolean falsity* $\mathtt{at}(\mathtt{ff})$. It becomes so only after the explicit inclusion of $\bot \to \mathtt{at}(\mathtt{ff})$ in the intuitionistic systems, see Definition 1.24. The *weak* ("classical") existential quantifier $\exists^{\mathtt{cl}}$ is defined in terms of $\forall$ and $\bot$ as

$$\exists^{\mathtt{cl}} x A(x) \quad :\equiv \quad (\forall x.\, A(x) \to \bot) \to \bot \tag{1.5}$$

Negation $\neg$ and equivalence $\leftrightarrow$ are defined as usual, i.e., $\neg A :\equiv A \to \bot$ and $A \leftrightarrow B :\equiv ((A \to B) \wedge (B \to A))$. Disjunction is defined by

$$A \vee B \quad :\equiv \quad \exists p^o\,[\,(p =_o \mathtt{tt} \to A) \,\wedge\, (p =_o \mathtt{ff} \to B)\,] \tag{1.6}$$

Here *predicate equality* at base types is *defined* for boolean terms s and t by

$$s =_o t \quad :\equiv \quad \mathtt{at}(s) \leftrightarrow \mathtt{at}(t) \tag{1.7}$$

and for natural terms s and t by $s =_\iota t :\equiv \mathtt{at}(\mathtt{Eq}\, s\, t)$. Equality between terms s and t of type $\tau \equiv \sigma_1 \ldots \sigma_n \sigma$, with $\sigma \in \{o, \iota\}$, is extensionally defined as

$$s =_\tau t \quad :\equiv \quad \forall x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\,(s\, x_1 \ldots x_n =_\sigma t\, x_1 \ldots x_n) \tag{1.8}$$

We also denote by $s \neq_\tau t :\equiv \neg(s =_\tau t)$ the *non-equality* between s and t. We will often omit to specify the type τ of equality or non-equality wherever there is no ambiguity.

Definition 1.4 (Quantifier-free formula) By quantifier-free (abbreviated $\mathtt{qfr}$) formula we understand a formula built from prime formulas $\mathtt{at}(t^o)$ and

[4] We do so because of the prominent place the minimal arithmetic $\mathtt{Z}_0$ has in the mechanism of the refined $\mathtt{A}$-translation extraction technique of [10]. This distinction will be very useful for the comparison with the $\mathtt{BBS}$ refined $\mathtt{A}$-translation exposed in Section 4.1.1.

[5] The historically first Minimal Logic system is due to Johansson [60].

$\bot$ by means of $\wedge$, $\rightarrow$ and, if $\exists$ is available, also $\vee$. All `qfr` formulas will be decidable in our systems, as soon as these include intuitionistic logic, see Lemma 1.36. The `qfr` formulas will be denoted by A_0, B_0, C_0, etc. ...

Definition 1.5 For a type σ we define the *degree* (also known as *level*) of σ, denoted $dg(\sigma)$, by $dg(\iota) := dg(o) := 0$ and $dg(\rho\tau) := max\{dg(\rho) + 1\,,\, dg(\tau)\}$. The definition extends to terms t^σ (i.e., t of type σ) by $dg(t) :\equiv dg(\sigma)$.

Definition 1.6 We denote by $mdg(t)$, $d(t)$ and $S(t)$ the maximal type degree of a subterm, the depth and respectively the size of t.

We also introduce in our system an adaptation of Berger's [9] *uniform* quantifiers, here denoted $\overline{\forall}$ (forall `ncm`) and $\overline{\exists}$ (exists `ncm`) to the extraction of (more efficient) programs by Gödel's Dialectica interpretation. From a logic viewpoint $\overline{\forall}$ and $\overline{\exists}$ behave exactly like $\forall$ and $\exists$ - a metatheorem stating that "the (purely syntactic) replacement of $\overline{\forall}$ and $\overline{\exists}$ with their *computationally meaningful*[6] (or *regular*) correspondents in a proof $\mathcal{P}$ yields a(nother) proof in the same system" can be established. However, the converse to this metatheorem does not hold (in general) because of the (necessary) restriction which is set on the introduction rule of the `ncm`-universal quantifier (see Section 1.2 below). The special rôle of $\overline{\forall}$ and $\overline{\exists}$ is played in the program-extraction process only. There they act like some kind of labels for parts of the proof at input which are to be ignored since they are *a priory* (i.e., at the proof-building stage) distinguished as having no computational content.

They also bring an important optimization with respect to the maximal type degree of programs extracted from those proofs for which the use of the computationally meaningful correspondents would have just brought an unjustified increase of this maximal type degree (upon which the run-time complexity of the normalization algorithm directly depends, regardless of the reduction strategy, see Berger's paper [9] for more on this).

Definition 1.7 We denote by $\mathcal{L}$ the language which does not contain any of $\exists$, $\overline{\exists}$, $\overline{\forall}$ and by $\mathcal{F}$ the corresponding set of formulas. By $\mathcal{L}^{\exists}/\mathcal{F}^{\exists}$ and $\mathcal{L}^{\mathtt{nc}}/\mathcal{F}^{\mathtt{nc}}$ the languages / respectively sets of formulas which may contain $\exists$ but none of $\overline{\exists}$, $\overline{\forall}$ and respectively which may contain $\overline{\forall}$ but none of $\exists$, $\overline{\exists}$. Finally, by $\mathcal{L}^{\exists,\mathtt{nc}}/\mathcal{F}^{\exists,\mathtt{nc}}$ we denote the full language / set of formulas, in which all of $\exists$, $\overline{\exists}$, $\overline{\forall}$ are allowed.

[6] Notice that $\forall$ is as computationally meaningful as $\exists$ in the context of program extraction by the (light, monotone) Dialectica interpretation, see Definition 2.1 and Theorem 2.10.

Definition 1.8 Similar to $\exists^{\mathtt{cl}}$ - see (1.5) - we define a *weak* `ncm`-existential quantifier $\overline{\exists^{\mathtt{cl}}}$ by

$$\overline{\exists^{\mathtt{cl}}} x A(x) \quad :\equiv \quad (\overline{\forall} x.\, A(x) \to \bot) \to \bot \tag{1.9}$$

In order to avoid excessive parenthesizing we make the usual conventions that $\forall$, $\overline{\forall}$, $\exists^{\mathtt{cl}}$, $\exists$, $\overline{\exists^{\mathtt{cl}}}$, $\overline{\exists}$, $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$ is the decreasing order of precedence and that $\to$ is right associative. For (more efficient) program-extraction purposes we will impose that all axioms are closed formulas and for optimization purposes - at the example of Schwichtenberg's `MinLog` system [111, 116] - their closure is ensured with $\overline{\forall}$ rather than $\forall$. The only (but notable) exception[7] to this rule is the Induction Axiom `IA`, see Section 1.3 for the various definitions of Induction within system $\mathtt{WeZ}^{\exists,\mathtt{nc}}$. Therefore it will be understood that even though a formula presented below as axiom is literally open, in fact the axiom it denotes is the $\overline{\forall}$ closure of the respective formula.

Positive and negative occurrences of quantifiers $\forall, \exists, \overline{\forall}, \overline{\exists}$ in formulas

Definition 1.9 (Quantifier-counting meta-functions) Given that Q is a universal or existential (`ncm`) quantifier – hence $Q \in \{\forall, \exists, \overline{\forall}, \overline{\exists}\}$, we define by induction on the structure of the formula A the following meta-functions:

- $occ^{\star}(Q, A)$ – which counts all occurrences of the quantifier Q in A
- $occ^{+}(Q, A)$ which counts all the so-called *positive* occurrences of Q in A
- $occ^{-}(Q, A)$ which counts all the so-called *negative* occurrences of Q in A

Hence the (mutually recursive) definition is as follows:

1. For prime formulas P :

$$\begin{aligned} occ^{\star}(Q, P) &:= 0 \\ occ^{+}(Q, P) &:= 0 \\ occ^{-}(Q, P) &:= 0 \end{aligned}$$

[7] This exception is necessary (only) in the context of Light Dialectica extraction (presented in Chapter 2) because the Dialectica realizers of `IA` integrate the induction formula via an $\to^{+}$ *with contraction*, see Section 1.3.5 and the proof of Theorem 2.10 below.

2. For the "same-quantified" formulas $Qx\,A(x)$:

$$\begin{aligned} occ^{\star}(Q, Qx\,A(x)) &:= occ^{\star}(Q, A(x)) + 1 \\ occ^{+}(Q, Qx\,A(x)) &:= occ^{+}(Q, A(x)) + 1 \\ occ^{-}(Q, Qx\,A(x)) &:= occ^{-}(Q, A(x)) \end{aligned}$$

3. For the "different-quantified" formulas $Q'x\,A(x)$, where $Q' \in \{\forall, \exists, \overline{\forall}, \overline{\exists}\}$ such that $Q \not\equiv Q'$:

$$\begin{aligned} occ^{\star}(Q, Q'x\,A(x)) &:= occ^{\star}(Q, A(x)) \\ occ^{+}(Q, Q'x\,A(x)) &:= occ^{+}(Q, A(x)) \\ occ^{-}(Q, Q'x\,A(x)) &:= occ^{-}(Q, A(x)) \end{aligned}$$

4. For the conjunctive formulas:

$$\begin{aligned} occ^{\star}(Q, A \wedge B) &:= occ^{\star}(Q, A) + occ^{\star}(Q, B) \\ occ^{+}(Q, A \wedge B) &:= occ^{+}(Q, A) + occ^{+}(Q, B) \\ occ^{-}(Q, A \wedge B) &:= occ^{-}(Q, A) + occ^{-}(Q, B) \end{aligned}$$

5. For the implicative formulas:

$$\begin{aligned} occ^{\star}(Q, A \to B) &:= occ^{\star}(Q, A) + occ^{\star}(Q, B) \\ occ^{+}(Q, A \to B) &:= occ^{-}(Q, A) + occ^{+}(Q, B) \\ occ^{-}(Q, A \to B) &:= occ^{+}(Q, A) + occ^{-}(Q, B) \end{aligned}$$

Definition 1.10 (Positive/negative occurrence/position of Q in A) We say that "Q has (at least) a positive occurrence in A", which we abbreviate $Occ^{+}(Q, A)$, if $occ^{+}(Q, A) > 0$ and also that "Q has (at least) a negative occurrence in A", abbreviated $Occ^{-}(Q, A)$, if $occ^{-}(Q, A) > 0$. The instances of the quantifiers Q which count up into occ^{+} are named "occurrences of Q in positive positions" and also symmetrically "occurrences of Q in negative positions" when such instances of Q count up into occ^{-}.

Remark 1.11 The meta-predicates $Occ^{+}(Q, A)$ and $Occ^{-}(Q, A)$ can also be recursively defined more directly.

Proposition 1.12 (Soundness of the *occ* meta-functions) For every quantifier $Q \in \{\forall, \exists, \overline{\forall}, \overline{\exists}\}$ and formula A, the following equality holds:

$$occ^{\star}(Q, A) \;=\; occ^{+}(Q, A) + occ^{-}(Q, A)$$

Proof: An occurrence of Q in A is either positive or negative. □

Definition 1.13 (Strictly positive occurrence/position of Q in A) An occurrence of the quantifier Q in the formula A is said to be *strictly* positive (or "in a *strictly* positive position") if it counts up into the modified variant of occ^{+}, which defines $occ^{+}(Q, A \to B) := occ^{+}(Q, B)$.

Remark 1.14 (Relation with Berger's homonymic notions) The notions of Definition 1.10, although related to the similar notions of Q-negative and Q-positive formulas (with $Q \in \{\forall, \exists\}$) from Berger [9], are nonetheless different. Basically, for Berger, a formula A is Q-negative iff $occ^{\star}(Q, A) = occ^{-}(Q, A)$ and is Q-positive iff $occ^{\star}(Q, A) = occ^{+}(Q, A)$. On the other hand, our notion from Definition 1.13 of "strictly positive" occurrence/position of Q in a formula coincides with Berger's similar notion from [9].

1.2 Logical axioms and rules and Boolean axioms

We begin by adapting the set of rules for the first-order Minimal predicate Logic from [111] and/or [10] [8] to the setting of program-extraction by the light (monotone) Dialectica interpretation (defined in Chapter 2). First of all we define the following two *variable conditions* which will be used to constrain the rules concerning the (`ncm`-)universal quantifier:

- $\mathtt{VC}_1(z)$: the variable z does not occur free in any of the undischarged assumptions of the proof of the premise of the rule;
- $\mathtt{VC}_2(z, t)$: the term t is "free for" z in the conclusion, i.e., no free variable of t gets quantified after substituting $\{z \leftarrow t\}$ in the conclusion.

We also define an ***ncm*-*formula condition*** which is required to constrain the rule of Implication Introduction (abbreviated "Imp Intro") which has that kind

[8] These purely logical rules are the same as those from the first-order restriction of the Uniform Arithmetic $\mathtt{HA}^{\mathtt{u}}$ of [9].

of contraction formula which is computationally relevant to the Light Dialectica (see below for this terminology). Such a restriction will be necessary in order to attain soundness of program-extraction by the light (monotone) Dialectica interpretation in Theorems 2.10 and respectively 2.12 from Chapter 2.

Definition 1.15 (The $\mathtt{ncm}-\mathtt{FC}$ restriction) For some given formula A, if A contains (at least) a positive universal or a negative existential (regular) quantifier, then A does not contain any $\mathtt{ncm}$ quantifier. We abbreviate this "$\mathtt{ncm}$-Formula Condition" restriction set on A by the shorter expression $\mathtt{ncm}-\mathtt{FC}(A)$.

Remark 1.16 (Rôle of $\mathtt{ncm}-\mathtt{FC}$ during $\mathtt{LD}$-extraction) If the $\mathtt{ncm}-\mathtt{FC}(A)$ restriction holds, then the $\mathtt{LD}$-interpretation of A (see Definition 2.1) will have a quantifier-free base formula $A_{\mathtt{D}}$. Therefore $A_{\mathtt{D}}$ will be decidable, cf. Proposition 1.35 / Lemma 1.36. If A is the formula-type of the (parcel) assumption variable u which is to be cancelled at some Imp Intro with $\mathtt{LDR}$-contraction (see below), the decidability of $A_{\mathtt{D}}$ will be strictly and unavoidably necessary for determining, according to actual values, which of the terms Light Dialectica realizing the multiple instances of A is chosen to Light Dialectica realize the new single implicative assumption A of the conclusion formula of such an Imp Intro. See Definition 2.6 and the proof of Theorem 2.10 for details.

The logical rules of our systems are then as follows:

1. Deduction from an (arbitrary, undischarged) assumption: $A \vdash A$.
2. Conjunction elimination left: $\dfrac{A \wedge B}{A}\ \wedge_l^-$, Conjunction elimination right: $\dfrac{A \wedge B}{B}\ \wedge_r^-$ and Conjunction introduction: $\dfrac{A\,,\,B}{A \wedge B}\ \wedge^+$.
3. Implication elimination: $\dfrac{A\,,\,A \to B}{B}\ \to^-$ (Modus Ponens).
4. Implication introduction: $\dfrac{[u\colon A]\ \ldots\ /B}{A \to B}\ \to^+$, where a particular (possibly empty) class u [9] of instances of the *introduction formula* A among

[9] This is called a "parcel" in [43], which is just a different terminology for the very same notion of "assumption variable" from [111].

the undischarged assumptions of the proof of B gets discharged. If at least two such instances of A get discharged one says that the $\rightarrow^+$ is *with (logical) contraction* and that A is its *contraction formula*. In such a situation the $\mathtt{ncm}-\mathtt{FC}(A)$ restriction applies. If none or just one instance of A gets cancelled, then one says that the $\rightarrow^+$ is *without (logical) contraction*.

Definition 1.17 (*n*-contraction) Let $n \in \mathbb{N}$. We say that an $\rightarrow^+$ is with n-contraction if n is the number of instances of its introduction formula which are discharged in the respective $\rightarrow^+$. For $n \in \{0, 1\}$ no (logical) contraction occurs, but it is technically useful to specify in this way the lack of (even) logical contraction.

Definition 1.18 (computationally relevant contraction formula) A contraction formula A is said to be *computationally relevant* to the LD-interpretation (abbreviated "computationally LD-relevant") if the premise of $\mathtt{ncm}-\mathtt{FC}(A)$ holds, i.e., if A contains at least a positive universal or a negative existential (regular) quantifier. If this is the case, we say that A is a "Light Dialectica relevant contraction" (abbreviated "LD-relevant contraction" and even shorter, "LDR-contraction") formula. We also say about the corresponding $\rightarrow^+$ that it is "with LD-relevant contraction". Otherwise, the formula A is said to be *computationally irrelevant* to the LD-interpretation and the corresponding $\rightarrow^+$ is "without LD-relevant contraction" or, equally, that it is "with LD-irrelevant contraction". Notice that, relative to the LD-interpretation, the only "true" contractions are the LD-relevant contractions. All the other, "purely logical" contractions have absolutely no computational impact under the LD-interpretation.

Remark 1.19 (computationally *D*-redundant contraction) Let A be a contraction formula. Because of the $\mathtt{ncm}-\mathtt{FC}(A)$, if A is computationally LD-relevant then it contains regular quantifiers only. Hence if A contains (at least) a positive universal or a negative existential ncm quantifier then it must be computationally LD-irrelevant. Such *D-relevant* contraction formula A would certainly have some computational content under the pure Gödel Dialectica interpretation. However, its computational contribution to the extracted terms of interest gets eliminated in the very final normalization process, see Section 4.5 for such an example. Such computationally LD-irrelevant contraction formulas are thus truly *computationally redundant* under the pure Dialectica interpretation.

5. ForAll elimination: $\dfrac{\forall z\, A(z)}{A(t)}\ \forall^-_{z,t}$, such that $\mathtt{VC}_2(z,t)$.

6. ncm-ForAll elimination: $\dfrac{\overline{\forall} z\, A(z)}{A(t)}\ \overline{\forall}^-_{z,t}$, such that $\mathtt{VC}_2(z,t)$.

7. ForAll introduction: $\dfrac{A(z)}{\forall z\, A(z)}\ \forall^+_z$, such that $\mathtt{VC}_1(z)$.

8. ncm-ForAll introduction: $\dfrac{\mathcal{P}\colon A(z)}{\overline{\forall} z\, A(z)}\ \overline{\forall}^+_z$, such that $\mathtt{VC}_1(z)$ and $\mathtt{VC}_3(z,\mathcal{P})$.

   The latter (third) *variable condition* applies to the $\overline{\forall}$-quantified variables only. Although partly the same as the pre-condition set by Berger on his $\{\forall\}^+$ rule in [9], an addition peculiar to light-Dialectica extraction is necessary, but also a relaxation is possible in both kinds of situation:

   - $\mathtt{VC}_3(z,\mathcal{P})$: the variable z does not occur free in the instantiating terms t involved by the ForAll eliminations $\forall^-_{\bullet,t}$ from the proof $\mathcal{P}$ (so far Berger's restriction) and z is also not free in the computationally LD-relevant contraction formulas involved by Implication Introductions $\rightarrow^+$ from $\mathcal{P}$ (see Definition 1.18), except[10] for the cases when at some point in $\mathcal{P}$ subsequent to such a $\forall^-$ or $\rightarrow^+$, the variable z gets to no longer be free in any of the uncancelled assumption or conclusion formulas of such a sub-proof of $\mathcal{P}$.

Definition 1.20 (Minimal Logics $\mathtt{ML}^{\mathtt{nc}}_0/\mathtt{ML}_0$, $\mathtt{ML}^{\mathtt{nc}}/\mathtt{ML}$ and $\mathtt{IL}^{\mathtt{nc}}_0/\mathtt{IL}_0$)
We denote by $\mathtt{ML}^{\mathtt{nc}}_0$ the above ncm minimal logic deduction system and by $\mathtt{ML}_0$ the system $\mathtt{ML}^{\mathtt{nc}}_0$ without the rules $\overline{\forall}^-$ and $\overline{\forall}^+$. Hence the languages of both $\mathtt{ML}^{\mathtt{nc}}_0$ and $\mathtt{ML}_0$ do include the ncm quantifiers and also the $\mathtt{ncm}-\mathtt{FC}$ restriction on $\rightarrow^+$, but $\mathtt{ML}_0$ does not include $\overline{\forall}^-$ and $\overline{\forall}^+$. We denote by $\mathtt{ML}^{\mathtt{nc}}$ and ML the extensions of $\mathtt{ML}^{\mathtt{nc}}_0$ and respectively $\mathtt{ML}_0$ with the following two *boolean axioms*:

[10] This exception is not necessary for the soundness of the "light" program extraction, but allows an enlargement of the class of admissible proofs. The condition is easy to verify by always keeping a list of variables which are free in the uncancelled assumptions or in the conclusion formula of the current (sub-)proof. It is effectively possible that z disappears from such a dynamic list, subsequent to its introduction by a $\forall^-_{\bullet,t}$ or a $\rightarrow^+$, hence it makes sense to spend some (few, linear overhead) resources on verifying this exception.

$\mathtt{AxFLS}$: $\mathtt{at}(\mathtt{ff}) \to A$ (Boolean Ex-Falso-Quodlibet)

$\mathtt{AxTRH}$: $\mathtt{at}(\mathtt{tt})$ (Boolean Truth Axiom)

Systems $\mathtt{IL}_0^{\mathtt{nc}}$ and $\mathtt{IL}_0$ are obtained by extending $\mathtt{ML}^{\mathtt{nc}}$, respectively $\mathtt{ML}$ with the following third boolean axiom:

$\mathtt{AxBIA}$: $A(\mathtt{tt}) \wedge A(\mathtt{ff}) \;\to\; \forall p^o A(p)$ (Boolean Induction Axiom)

and also all equality axioms introduced in the subsequent Section 1.3 restricted to terms built from variables and constants of type o only. The latter addition is purely technical – we make it in order to enhance the expressiveness of our non-arithmetical systems so that equivalence between boolean and logical conjunction, implication and (if available) disjunction can be proved in the minimal systems already, see Lemma 1.34.

Remark 1.21 Modulo the $\mathtt{ncm}-\mathtt{FC}$ restriction, system $\mathtt{ML}_0$ is the regular Minimal Logic system. Deduction in $\mathtt{ML}_0$ is thus simply regular Minimal Logic deduction if the language of the formulas involved does not contain any $\mathtt{ncm}$ quantifier. On the contrary, if this language contains $\mathtt{ncm}$ quantifiers, then the $\mathtt{ncm}-\mathtt{FC}$ restriction applies to all $\to^+$ with contraction of such $\mathtt{ML}_0$ deductions.

Remark 1.22 Axioms $\mathtt{AxFLS}$ and $\mathtt{AxTRH}$ give the desired logical behaviour of the boolean constants $\mathtt{ff}$, respectively $\mathtt{tt}$. They are also essential for proving the aforementioned Lemma 1.34. Whereas $\mathtt{AxTRH}$ establishes the equivalence of boolean and logical truth, $\mathtt{AxFLS}$ only implies that boolean falsity is stronger than logical falsity, see also Remark 1.25 below.

Remark 1.23 Axiom $\mathtt{AxBIA}$ is necessary for attaining in our formulation the usual logical behaviour of $\vee$. Only the "disjunction introduction" theorems $A \to A \vee B$ and $B \to A \vee B$ can be ensured in $\mathtt{ML}^{\exists}$ by the definition (1.6) of $\vee$ alone. The proof of the "elimination of disjunction" schema

$$A \to C\ ,\ B \to C \;\vdash\; (A \vee B) \to C$$

requires the further addition of $\mathtt{AxBIA}$, see Proposition 1.33 below.

Definition 1.24 (Intuitionistic Logics $\mathtt{IL}^{\mathtt{nc}}/\mathtt{IL}$) Systems $\mathtt{IL}^{\mathtt{nc}}$ and $\mathtt{IL}$ are obtained by extending $\mathtt{IL}_0^{\mathtt{nc}}$, respectively $\mathtt{IL}_0$ with the *logical* axiom

$\mathtt{AxEFQ}$: $\bot \to \mathtt{at}(\mathtt{ff})$ (Logical Ex-Falso-Quodlibet)

Remark 1.25 The addition of $\mathtt{AxEFQ}$ is essential for attaining the equivalence of boolean and logical falsity in the systems $\mathtt{IL}^{\mathtt{nc}}/\mathtt{IL}$, see Remark 1.22 above. For this reason, axiom $\mathtt{AxEFQ}$ will also be essential in the treatment of quantifier-free formulas as atomic formulas, see Proposition 1.35.

Following [10] rather than [9] we consider that the strong existential quantifiers are not automatically part of minimal or intuitionistic logics. They can be modularly adjoined to the above logical systems, via the following axioms:

$$\begin{array}{lll}
\mathtt{Ax}\exists^{-}: & \exists z_1\, A(z_1) \,\wedge\, \forall z_2\,[\,A(z_2) \,\to\, B\,] \,\to\, B & (\exists \text{ elimination}) \\
\mathtt{Ax}\exists^{+}: & \forall z_1\,[\,A(z_1) \,\to\, \exists z_2\, A(z_2)\,] & (\exists \text{ introduction}) \\
\mathtt{Ax}\overline{\exists}^{\,-}: & \overline{\exists} z_1\, A(z_1) \,\wedge\, \overline{\forall} z_2\,[\,A(z_2) \,\to\, B\,] \,\to\, B & (\overline{\exists} \text{ elimination}) \\
\mathtt{Ax}\overline{\exists}^{\,+}: & \overline{\forall} z_1\,[\,A(z_1) \,\to\, \overline{\exists} z_2\, A(z_2)\,] & (\overline{\exists} \text{ introduction})
\end{array}$$

with the usual restriction that z_2 is not free in B. We thus obtain the following:

Definition 1.26 (Minimal|Intuitionistic Logics with strong existence) Systems $\mathtt{ML}_0^{\exists}/\mathtt{ML}^{\exists}/\mathtt{IL}_0^{\exists}|\mathtt{IL}^{\exists}$ are obtained by adding $\mathtt{Ax}\exists^{-}$ and $\mathtt{Ax}\exists^{+}$ to $\mathtt{ML}_0/\mathtt{ML}/\mathtt{IL}_0$, respectively $\mathtt{IL}$. Systems $\mathtt{ML}_0^{\exists,\mathtt{nc}}/\mathtt{ML}^{\exists,\mathtt{nc}}/\mathtt{IL}_0^{\exists,\mathtt{nc}}|\mathtt{IL}^{\exists,\mathtt{nc}}$ are obtained by adding to $\mathtt{ML}_0^{\mathtt{nc}}/\mathtt{ML}^{\mathtt{nc}}/\mathtt{IL}_0^{\mathtt{nc}}|\mathtt{IL}^{\mathtt{nc}}$ all four axioms defining $\exists$ and $\overline{\exists}$: $\mathtt{Ax}\exists^{-}$, $\mathtt{Ax}\exists^{+}$, $\mathtt{Ax}\overline{\exists}^{\,-}$, $\mathtt{Ax}\overline{\exists}^{\,+}$.

Notation. We will denote by $\mathtt{IL}^{(\exists)} :\equiv \mathtt{IL}/\mathtt{IL}^{\exists}$, i.e., $\mathtt{IL}^{(\exists)}$ acts like a placeholder for both $\mathtt{IL}$ and $\mathtt{IL}^{\exists}$. Similarly $\mathtt{IL}_0^{(\exists)} :\equiv \mathtt{IL}_0/\mathtt{IL}_0^{\exists}$ and $\mathtt{IL}_0^{(\mathtt{nc})} :\equiv \mathtt{IL}_0/\mathtt{IL}_0^{\mathtt{nc}}$.

The strong existential quantifiers (including $\vee$) could have equally been introduced by means of rules instead of axioms, see, e.g., the presentations from the more recent paper [9] or the more complete book [123]. We here follow [111] in choosing a formulation which we also find more suitable for the computer-applied program extraction from (classical) proofs, particularly when using the implemented $\mathtt{MinLog}$ proof-system [49, 116].

1.2.1 Stability, Case Distinction, Decidability and Disjunction Introduction/Elimination

Notational Convention. For simplicity we here-on abbreviate $\mathtt{at}(\mathtt{tt})$ by $\mathtt{T}$ and $\mathtt{at}(\mathtt{ff})$ by $\mathtt{F}$. We also abbreviate by $\rceil A :\equiv A \to \mathtt{F}$ the so-called "boolean negation" of the formula A.

Lemma 1.27 The following hold in minimal logics $\mathtt{ML}$, respectively $\mathtt{IL_0}$:

$$\mathtt{ML} \ \vdash \ \forall q.\, q =_o \mathtt{tt} \leftrightarrow \mathtt{at}(q) \tag{1.10}$$

$$\mathtt{ML} \ \vdash \ \forall q.\, q =_o \mathtt{ff} \leftrightarrow (\mathtt{at}(q) \to \mathtt{F}) \equiv \rceil\mathtt{at}(q) \tag{1.11}$$

$$\mathtt{ML} \ \vdash \ \forall q.\, \rceil\mathtt{at}(q) \to \neg\mathtt{at}(q) \tag{1.12}$$

$$\mathtt{ML} \ \vdash \ [\forall q.\, \neg\mathtt{at}(q) \to \rceil\mathtt{at}(q)\,] \leftrightarrow (\bot \to \mathtt{F}) \tag{1.13}$$

$$\mathtt{IL_0} \ \vdash \ [\forall q.\, \neg\neg\mathtt{at}(q) \to \mathtt{at}(q)\,] \leftrightarrow (\bot \to \mathtt{F}) \tag{1.14}$$

Proof: Even without using $\mathtt{AxBIA}$, (1.10), (1.11) and (1.12) are simply immediate from the definition (1.7) of $=_o$, $\mathtt{AxTRH}$ and $\mathtt{AxFLS}$. Also without $\mathtt{AxBIA}$, at (1.13) the direction "$\leftarrow$" follows immediately and the direction "$\rightarrow$" follows from $\mathtt{AxTRH}$ after setting $q := \mathtt{tt}$. The direction "$\leftarrow$" of (1.14) can only be proved by boolean induction $\mathtt{AxBIA}$ on q. Hence for $q \equiv \mathtt{tt}$, $\neg\neg\mathtt{at}(\mathtt{tt}) \to \mathtt{at}(\mathtt{tt})$ follows from $\mathtt{AxTRH}$. On the other hand, for $q \equiv \mathtt{ff}$, $\neg\neg\mathtt{at}(\mathtt{ff}) \to \mathtt{at}(\mathtt{ff})$ rewrites as $((\mathtt{F} \to \bot) \to \bot) \to \mathtt{F}$ which is immediately seen to be equivalent to $\bot \to \mathtt{F}$ in minimal logic $\mathtt{ML}$. This also proves the direction "$\rightarrow$" of (1.14). $\Box$

Corollary 1.28 (Boolean/Logical Stability of atomic formulas)
In the same lines of the above proof of (1.14) it immediately follows that:

$$\mathtt{IL_0} \ \vdash \ \forall q^o.\, \rceil\rceil\mathtt{at}(q) \to \mathtt{at}(q) \tag{1.15}$$

On the other hand, from (1.14) it immediately follows that

$$\mathtt{IL} \ \vdash \ \forall q^o.\, \neg\neg\mathtt{at}(q) \to \mathtt{at}(q) \tag{1.16}$$

where the axiom $\mathtt{AxEFQ} : \bot \to \mathtt{F}$ is strictly necessary in the proof of (1.16).

Lemma 1.29 (Boolean/Logical Case Distinction on atomic formulas)
The following hold in minimal logic $\mathtt{IL_0}$ for every formula A of $\mathcal{F}^{\exists,\mathtt{nc}}$:

$$\mathtt{IL_0} \ \vdash \ \forall q^o.\, (q =_o \mathtt{tt} \to A) \wedge (q =_o \mathtt{ff} \to A) \ \to \ A$$

$$\mathtt{IL_0} \ \vdash \ \forall q^o.\, (\mathtt{at}(q) \to A) \wedge (\rceil\mathtt{at}(q) \to A) \ \to \ A$$

$$\mathtt{IL_0} \ \vdash \ \forall q^o.\, (\mathtt{at}(q) \to A) \wedge (\neg\mathtt{at}(q) \to A) \ \to \ A$$

Proof: Immediate by $\mathtt{AxBIA}$, $\mathtt{AxTRH}$, $\mathtt{AxFLS}$, (1.10), (1.11) and the definitions of $\rceil$, respectively $\neg$. Since all $\to^+$ involved in these proofs are without contraction, the $\mathtt{ncm{-}FC}$ restriction is satisfied even when A contains $\mathtt{ncm}$ quantifiers. $\Box$

Lemma 1.30 (Boolean Multiple Distinction over atomic formulas) Let $p_1, \ldots, p_n$ be variables of type o. Then for every formula B of $\mathcal{F}^{\exists,\mathtt{nc}}$, the following sentence is a theorem of $\mathtt{IL_0}$ (below one omits the display of the leading $\forall$ quantification over $p_1, \ldots, p_n$, which is self-understood):

$$\begin{array}{rcl} (\,\rceil\mathtt{at}(p_1) \to B\,) \to & [\,\mathtt{at}(p_1) \wedge \rceil\mathtt{at}(p_2) \to B\,] \to & \ldots \\ \ldots \to & [\,\wedge_{j=1}^{i-1}\mathtt{at}(p_j) \wedge \rceil\mathtt{at}(p_i) \to B\,] \to & \ldots \\ \ldots \to & [\,\wedge_{j=1}^{n-1}\mathtt{at}(p_j) \wedge \rceil\mathtt{at}(p_n) \to B\,] \to & \wedge_{i=1}^{n}\mathtt{at}(p_i) \to B \end{array}$$

Proof: Multiple boolean induction over $p_1, \ldots, p_n$ and use of `AxTRH`, `AxFLS`. □

Lemma 1.31 (Boolean/Logical Decidability of atomic formulas) The following hold in minimal logic $\mathtt{IL}_0^{\exists}$:

$$\begin{array}{rcl} \mathtt{IL}_0^{\exists} & \vdash & \forall q^o.\ q =_o \mathtt{tt} \vee q =_o \mathtt{ff} \\ \mathtt{IL}_0^{\exists} & \vdash & \forall q^o.\ \mathtt{at}(q) \vee \rceil\mathtt{at}(q) \\ \mathtt{IL}_0^{\exists} & \vdash & \forall q^o.\ \mathtt{at}(q) \vee \neg\mathtt{at}(q) \end{array}$$

Proof: By unfolding the definition (1.6) of $\vee$ and using (1.10), (1.11) and the definitions of $\rceil$, respectively $\neg$, the conclusions become

$$\begin{array}{rcl} \mathtt{IL}_0^{\exists} & \vdash & \forall q\,\exists p\,.\,(\,\mathtt{at}(p) \to \mathtt{at}(q)\,) \wedge [\,(\mathtt{at}(p) \to \mathtt{F}) \to (\mathtt{at}(q) \to \mathtt{F})\,] \\ \mathtt{IL}_0^{\exists} & \vdash & \forall q\,\exists p\,.\,(\,\mathtt{at}(p) \to \mathtt{at}(q)\,) \wedge [\,(\mathtt{at}(p) \to \mathtt{F}) \to (\mathtt{at}(q) \to \bot)\,] \quad (1.17) \end{array}$$

which are immediate after we set $p :\equiv q$, also using `AxFLS` for (1.17). □

Lemma 1.32 (Boolean/Logical Multiple Decidability of atmc fmlas) Let $p_1, \ldots, p_n$ be variables of type o. Then the following hold both in $\mathtt{IL}_0^{\exists}$:

$$\begin{array}{rcl} \vdash & \rceil\mathtt{at}(p_1) \vee [\mathtt{at}(p_1) \wedge \rceil\mathtt{at}(p_2)] \vee \ldots [\wedge_{j=1}^{i-1}\mathtt{at}(p_j) \wedge \rceil\mathtt{at}(p_i)] \ldots \vee [\wedge_{i=1}^{n}\mathtt{at}(p_i)] \\ \vdash & \neg\mathtt{at}(p_1) \vee [\mathtt{at}(p_1) \wedge \neg\mathtt{at}(p_2)] \vee \ldots [\wedge_{j=1}^{i-1}\mathtt{at}(p_j) \wedge \neg\mathtt{at}(p_i)] \ldots \vee [\wedge_{i=1}^{n}\mathtt{at}(p_i)] \end{array}$$

Proof: Multiple boolean induction over $p_1, \ldots, p_n$, unfolding of the definition (1.6) of $\vee$ and use of `AxTRH`, `AxFLS`. □

Proposition 1.33 (Disjunction Introduction and Elimination) For all $n \in \mathbb{N}$, all $i \in \overline{1,n}$, and $B, A_i \in \mathcal{F}^{\exists,\mathtt{nc}}$ the following hold:

$$\begin{array}{rcll} \mathtt{ML}^{\exists} & \vdash & A_i \to \vee_{i=1}^{n} A_i & (1.18) \\ \mathtt{IL}_0^{\exists} & \vdash & (A_1 \to B) \to \ldots \to (A_n \to B) \to (\vee_{i=1}^{n} A_i \to B) & (1.19) \end{array}$$

Proof: In both minimal logic proofs which we provide for (1.18) and (1.19), the $\rightarrow^+$ are without (logical) contraction, hence the $\mathtt{ncm{-}FC}$ restriction is obviously satisfied. In both cases it is sufficient to establish such proofs for $n = 2$. These immediately give the induction step in the meta-proof for a more general n. For (1.18) we have

$$\begin{aligned} \mathtt{ML} \;&\vdash\; A_1 \;\rightarrow\; (\mathtt{tt} =_o \mathtt{tt} \rightarrow A_1) \wedge (\mathtt{tt} =_o \mathtt{ff} \rightarrow A_2) \\ \mathtt{ML} \;&\vdash\; A_2 \;\rightarrow\; (\mathtt{tt} =_o \mathtt{ff} \rightarrow A_1) \wedge (\mathtt{tt} =_o \mathtt{tt} \rightarrow A_2) \end{aligned}$$

from which (1.18) follows immediately by $\mathtt{Ax}\exists^+$. After using $\mathtt{Ax}\exists^-$ once, for (1.19) we only need to prove that the following holds in $\mathtt{IL_0}$:

$$A_1 \rightarrow B\,,\; A_2 \rightarrow B \;\vdash\; \forall p^o.\; (p =_o \mathtt{tt} \rightarrow A_1) \wedge (p =_o \mathtt{ff} \rightarrow A_2) \;\rightarrow\; B$$

which reduces to finding a proof of B from assumptions $A_1 \rightarrow B$, $A_2 \rightarrow B$, $p =_o \mathtt{tt} \rightarrow A_1$ and $p =_o \mathtt{ff} \rightarrow A_2$. It is then enough to find a proof of B from the assumptions $p =_o \mathtt{tt} \rightarrow B$ and $p =_o \mathtt{ff} \rightarrow B$. This is just a Case Distinction. □

Lemma 1.34 The following lemmas hold in the minimal logic $\mathtt{IL_0}$, respectively $\mathtt{IL_0^\exists}$. They establish the equivalence for terms of boolean and logical conjunction, implication and respectively disjunction. This will permit the treatment of $\mathtt{qfr}$ formulas as atomic formulas in $\mathtt{IL}^{(\exists)}$, see Proposition 1.35.

$$\begin{aligned} &\mathtt{LmAND}: && \mathtt{IL_0} \;\vdash\; \forall p^o, q^o.\; \mathtt{at}(\mathtt{And}\, p\, q) \;\leftrightarrow\; \mathtt{at}(p) \wedge \mathtt{at}(q) \\ &\mathtt{LmIMP}: && \mathtt{IL_0} \;\vdash\; \forall p^o, q^o.\; \mathtt{at}(\mathtt{Imp}\, p\, q) \;\leftrightarrow\; \mathtt{at}(p) \rightarrow \mathtt{at}(q) \\ &\mathtt{LmOR}\;: && \mathtt{IL_0^\exists} \;\vdash\; \forall p^o, q^o.\; \mathtt{at}(\mathtt{Or}\, p\, q) \;\leftrightarrow\; \mathtt{at}(p) \vee \mathtt{at}(q) \end{aligned}$$

Proof: All three lemmas can be proved directly by boolean induction on p, using the definitions (1.1), (1.2) and (1.3), the rewrite rules for $\mathtt{AxIf}_o$ and general logical deduction in $\mathtt{IL_0^{(\exists)}}$. They can also be established as immediate consequences in $\mathtt{IL_0^{(\exists)}}$ of the following more general $\mathtt{IL_0}$-lemma, denoted $\mathtt{LmIF}$, which gives the full logical behaviour of the if-then-else boolean constant $\mathtt{If}_o$:

$$\forall p^o, q_1^o, q_2^o.\; \mathtt{at}(\mathtt{If}_o\, p\, q_1\, q_2) \;\leftrightarrow\; (\mathtt{at}(p) \rightarrow \mathtt{at}(q_1)) \wedge [(\mathtt{at}(p) \rightarrow \mathtt{F}) \rightarrow \mathtt{at}(q_2)]$$

At its turn, $\mathtt{LmIF}$ is immediately proved in $\mathtt{IL_0}$ by boolean induction on p. □

Proposition 1.35 (Equivalence between $\mathtt{qfr}$ and atomic formulas) There exists a unique bijective association of boolean terms to quantifier-free formulas $A_0 \mapsto \mathtt{t}_{A_0}$ such that $\mathtt{IL} \vdash A_0 \leftrightarrow \mathtt{at}(\mathtt{t}_{A_0})$ holds for all A_0 without $\vee$ and otherwise $\mathtt{IL^\exists} \vdash A_0 \leftrightarrow \mathtt{at}(\mathtt{t}_{A_0})$ holds for all quantifier-free A_0.

Proof: By induction on the structure of A_0. Obvious for atomic formulas $\mathtt{at}(t^o)$. For the prime formula $\bot$, we have $\mathtt{IL} \vdash \bot \leftrightarrow \mathtt{at}(\mathtt{ff})$, hence $\mathtt{t}_\bot :\equiv \mathtt{ff}$. For composed formulas we use `LmAND`, `LmIMP` and, if $\exists$ is available, also `LmOR`. □

Lemma 1.36 (Stability, Case Distinction and Decidability for `qfr`) The following hold as immediate consequences of Proposition 1.35 applied to Lemmas 1.28, 1.29 and 1.31 :

$$\mathtt{IL} \;\vdash\; \neg\neg A_0 \;\to\; A_0 \tag{1.20}$$
$$\mathtt{IL} \;\vdash\; (A_0 \to A) \wedge (\neg A_0 \to A) \;\to\; A \tag{1.21}$$
$$\mathtt{IL}^{\exists} \;\vdash\; A_0 \vee \neg A_0 \tag{1.22}$$

for every quantifier-free formula A_0 of $\mathcal{F}$ and every formula A of $\mathcal{F}^{\exists,\mathtt{nc}}$. In the case when A_0 does not contain $\bot$ then also the following can be established as immediate consequences of Lemmas 1.27, 1.29 and 1.31:

$$\mathtt{IL_0} \;\vdash\; ⌉⌈ A_0 \;\to\; A_0$$
$$\mathtt{IL_0} \;\vdash\; (A_0 \to A) \wedge (⌈ A_0 \to A) \;\to\; A$$
$$\mathtt{IL_0^{\exists}} \;\vdash\; A_0 \vee ⌈ A_0$$

Lemma 1.37 (Multiple Case Distinction and Decidability for `qfr`) Let $A_0^1, \ldots, A_0^n$ be quantifier-free formulas. Then for every formula B of $\mathcal{F}^{\exists,\mathtt{nc}}$, the following "multiple Case Distinction over quantifier-free formulas" formula holds as theorem of `IL` :

$$\begin{array}{rcl} (\,⌈ A_0^1 \to B\,) \;\to & [\,A_0^1 \wedge ⌈ A_0^2 \to B\,] \;\to\; \ldots & \\ \ldots \;\to & [\,\wedge_{j=1}^{i-1} A_0^j \wedge ⌈ A_0^i \to B\,] \;\to\; \ldots & \\ \ldots \;\to & [\,\wedge_{j=1}^{n-1} A_0^j \wedge ⌈ A_0^n \to B\,] \;\to\; (\wedge_{i=1}^{n} A_0^i \;\to\; B) \;\to\; B \end{array} \tag{1.23}$$

Also the following sentence of "multiple Decidability for quantifier-free formulas" holds in $\mathtt{IL}^{\exists}$:

$$\vdash\; ⌉ A_0^1 \;\vee\; (\,A_0^1 \wedge ⌈ A_0^2\,) \vee \ldots \vee\; (\,\wedge_{j=1}^{i-1} A_0^j \wedge ⌈ A_0^i\,) \vee \ldots \vee\; (\,\wedge_{i=1}^{n} A_0^i\,) \tag{1.24}$$

Proof: One combines Proposition 1.35 with Lemma 1.30 and respectively Lemma 1.32. □

Lemma 1.38 The following hold for arbitrary $\mathcal{F}^{\exists,\mathtt{nc}}$ formulas A and B :

$$\mathtt{ML_0} \vdash A \to \neg\neg A \tag{1.25}$$
$$\mathtt{ML_0} \vdash A \to \rceil\rceil A \tag{1.26}$$
$$\mathtt{ML_0} \vdash \neg\neg\forall x \neg\neg A(x) \leftrightarrow \forall x \neg\neg A(x) \tag{1.27}$$
$$\mathtt{ML_0} \vdash \rceil\rceil\forall x \rceil\rceil A(x) \leftrightarrow \forall x \rceil\rceil A(x) \tag{1.28}$$
$$\mathtt{ML_0^{nc}} \vdash \neg\neg\overline{\forall} x \neg\neg A(x) \leftrightarrow \overline{\forall} x \neg\neg A(x) \tag{1.29}$$
$$\mathtt{ML_0^{nc}} \vdash \rceil\rceil\overline{\forall} x \rceil\rceil A(x) \leftrightarrow \overline{\forall} x \rceil\rceil A(x) \tag{1.30}$$

Schemas (1.27), (1.28), (1.29) and (1.30) hold also for x a tuple of variables.

Proof: By the definitions of $\neg$, respectively $\rceil$, (1.25) and (1.26) are simply obvious. We here give in detail only the proof of (1.29). This is actually similar to the usual proof of (1.27) (see, e.g., [111, 123]) because the special conditions for the $\mathtt{ncm}$ universal quantifier introduction are satisfied. The implication $\overline{\forall} x \neg\neg A(x) \to \neg\neg\overline{\forall} x \neg\neg A(x)$ is just an instance of (1.25). Below we display the $\mathtt{ML_0^{nc}}$ proof of the converse:

$$\dfrac{\mathcal{P}\left\{\begin{array}{l} \dfrac{\dfrac{[1:\overline{\forall} x \neg\neg A(x)] \quad y}{\neg\neg A(y)} \quad [2:\neg A(y)]}{\bot} \to_2^+ \\ \dfrac{\boxed{\neg\neg\overline{\forall} x \neg\neg A(x)} \quad \dfrac{\bot}{\neg\neg\overline{\forall} x \neg\neg A(x)} \to_1^+}{\bot} \end{array}\right.}{\overline{\forall} y \neg\neg A(y)} \overline{\forall}_y^+$$

Notice that $\mathtt{VC}_3(y, \mathcal{P})$ is immediately satisfied because $\mathcal{P}$ contains no $\forall^-$ and both $\to_1^+$ and $\to_2^+$ are without contraction. Also $\mathtt{VC}_1(y)$ holds because the only undischarged assumption of $\mathcal{P}$ is $\neg\neg\overline{\forall} x \neg\neg A(x)$ (see it framed in $\mathcal{P}$ above). □

Lemma 1.39 The following hold for arbitrary $\mathcal{F}^{\exists}$ formulas A and B :

$$\mathtt{ML_0} \vdash \neg\neg(A \wedge B) \leftrightarrow (\neg\neg A) \wedge (\neg\neg B) \tag{1.31}$$
$$\mathtt{IL} \vdash \neg\neg(A \to B) \leftrightarrow (A \to \neg\neg B) \leftrightarrow (\neg\neg A \to \neg\neg B) \tag{1.32}$$

However, none of (1.31), (1.32) scales to the language $\mathcal{L}^{\exists,\mathtt{nc}}$, in the sense that there exist formulas A and B of $\mathcal{F}^{\exists,\mathtt{nc}}$ such that at least one of A, B contains at least an $\mathtt{ncm}$ quantifier and (1.31), respectively (1.32) do not hold in $\mathtt{ML_0^{nc}}/\mathtt{IL^{nc}}$.

Proof: Both (1.31) and (1.32) have straightforward generic Natural Deduction proofs which can be found, e.g., in [111, 123]. No meta-induction on the structure of formulas is needed. At (1.32) only the more difficult implication

$$(\neg\neg A \to \neg\neg B) \;\to\; \neg\neg(A \to B) \tag{1.33}$$

requires an $\mathtt{AxEFQ}$, all other implications hold in $\mathtt{ML}_0$. The directions

$$\neg\neg(A \wedge B) \;\to\; (\neg\neg A) \wedge (\neg\neg B) \tag{1.34}$$

for (1.31) and (1.33) for (1.32) cannot be proved without contractions over the formulas $\neg\neg(A \wedge B)$ and respectively $\neg(A \to B)$. It is very easy to build such formulas which violate the $\mathtt{ncm{-}FC}$ restriction on $\to^+$ and therefore (1.31)/(1.32) do not hold in $\mathtt{ML}_0/\mathtt{IL}$ in such cases. The addition of the rules $\overline{\forall}^+$ and $\overline{\forall}^-$ does not bring any change to this situation. □

Lemma 1.40 (Existential-free Stability for $\mathcal{L}$) The schema $\neg\neg C \to C$ holds in $\mathtt{IL}$ for all formulas C of $\mathcal{F}$. However, the result does not scale to proofs in $\mathtt{IL}^{\mathtt{nc}}$ of formulas of $\mathcal{F}^{\mathtt{nc}}$, in the sense that there exists a C which contains at least a $\overline{\forall}$, for which $\neg\neg C \to C$ does not hold in $\mathtt{IL}^{\mathtt{nc}}$.

Proof: We first prove the affirmative statement for formulas without $\overline{\forall}$. The proof is by induction on the structure of the formula C. The case $C \equiv A \wedge B$ follows immediately using the implication (1.34) of (1.31). Similarly, for $C \equiv A \to B$ one uses the implication $\neg\neg(A \to B) \to (A \to \neg\neg B)$ of (1.32) and the induction hypothesis for B only, i.e., $\neg\neg B \to B$. For the case $C \equiv \forall x A(x)$ we have the following minimal logic proof:

$$\dfrac{\dfrac{\overbrace{\begin{array}{c}\emptyset\\ \vdots\\ \neg\neg A(y) \to A(y)\end{array}}^{\mathcal{P}'} \qquad \dfrac{\boxed{\neg\neg\forall x A(x)} \qquad \dfrac{\dfrac{\dfrac{[2:\neg A(y)] \quad \dfrac{[1:\forall x A(x)] \quad y}{A(y)}}{\bot}\,\to_1^+}{\neg\forall x A(x)}}{\bot}\,\to_2^+}{\neg\neg A(y)}}{A(y)}\,\Big\}\,\mathcal{P}}{\forall y A(y)}$$

The condition $\mathtt{VC}_1(y)$ is satisfied for the final $\forall_y^+$ because $\neg\neg\forall x A(x)$ (which is displayed framed in $\mathcal{P}$ above) is the only undischarged assumption of $\mathcal{P}$ and y is a fresh variable w.r.t. $A(x)$.

For the negative statement, let us first remark that one can build formulas $C \equiv A \wedge B$ of $\mathcal{F}^{\mathtt{nc}}$ for which $\mathtt{ncm}\text{-}\mathtt{FC}(\neg\neg(A \wedge B))$ does not hold and therefore (1.34) does not hold, see Lemma 1.39. Another (even earlier, in the induction course) possibility of a counterexample is for the case $C \equiv \overline{\forall} x A(x)$ which is isomorphic with $C \equiv \forall x A(x)$ but the extra condition $\mathtt{VC}_3(y, \mathcal{P})$ may not hold. Notice that $\mathtt{VC}_3(y, \mathcal{P}) \Leftrightarrow \mathtt{VC}_3(y, \mathcal{P}')$ and the latter may not hold because y may be free in some computationally relevant contraction formula of $\mathcal{P}'$. Such contraction is also due to an implication (1.34), which here involves formulas without $\overline{\forall}$ only. $\square$

Corollary 1.41 (Partial Stability for $\mathcal{L}^{\exists}$ and non-Stability for $\mathcal{L}^{\exists,\mathtt{nc}}$) The Stability schema $\neg\neg C \to C$ holds in $\mathtt{IL}$, for all C of $\mathcal{F}^{\exists}$ which do not contain the strong $\exists$ in a strictly positive position (see Definition 1.13 for this notion). No straightly similar statement can be made for formulas $C \in \mathcal{F}^{\exists,\mathtt{nc}}$.

Proof: For the positive statement, one simply exploits the fact that only the stability of B is required at the induction step for $C \equiv A \to B$ and thus there is no restriction on A, which can freely contain the strong $\exists$. On the contrary, if C contains $\mathtt{ncm}$ quantifiers then problems can appear in the inductive treatment of conjunction, due to the necessary use of (1.34), see also the counterexample in the proof of Lemma 1.40 above. See also the Remark below. $\square$

Remark 1.42 (Non-Stability in general for all the $\mathtt{ncm}$ languages) The schema $\neg\neg C \to C$ would hold in the extension of $\mathtt{IL}^{\mathtt{nc}}$ with the schema (1.34) for the full language $\mathcal{L}^{\exists,\mathtt{nc}}$, certainly for all formulas C of $\mathcal{F}^{\mathtt{nc}}$, but also for those $C \in \mathcal{F}^{\exists,\mathtt{nc}}$ which do not contain $\overline{\exists}$ or $\exists$ in a strictly positive position. Indeed, the addition of (1.34) as axiom would eliminate the contraction from its proof and thus also all restrictions involved by the use of $\mathtt{ncm}$ universal quantifiers in the formulas C. But this would only be artificial, since no Light Dialectica realizer can be directly provided for the $\mathtt{LD}$-interpretation of (1.34) such that the contraction over $\neg\neg(A \wedge B)$ to be avoided. Thus the only reasonable amount of stability which we can allow in view of our extractive purposes in the constructive systems is for formulas without $\mathtt{ncm}$ quantifiers and is given in Corollary 1.41 above.

Lemma 1.43 The following hold for all formulas $A(z)$ of $\mathcal{F}^{\mathtt{nc}}$ and B of $\mathcal{F}$:

$\mathtt{Lm}\exists^{\mathtt{cl}}_{-}$: $\quad \mathtt{IL} \vdash \exists^{\mathtt{cl}} z_1\, A(z_1) \wedge \forall z_2\,[\,A(z_2) \to B\,] \to B \qquad (\exists^{\mathtt{cl}}$ elimination)

$\mathtt{Lm}\exists^{\mathtt{cl}}_{+}$: $\quad \mathtt{ML_0} \vdash \forall z_1\,[\,A(z_1) \to \exists^{\mathtt{cl}} z_2\, A(z_2)\,] \qquad (\exists^{\mathtt{cl}}$ introduction)

$\mathtt{Lm}\overline{\exists^{\mathtt{cl}}}_{-}$: $\quad \mathtt{IL}^{\mathtt{nc}} \vdash \overline{\exists^{\mathtt{cl}}} z_1\, A(z_1) \wedge \overline{\forall} z_2\,[\,A(z_2) \to B\,] \to B \qquad (\overline{\exists^{\mathtt{cl}}}$ elimination)

$\mathtt{Lm}\overline{\exists^{\mathtt{cl}}}_{+}$: $\quad \mathtt{ML}^{\mathtt{nc}}_0 \vdash \overline{\forall} z_1\,[\,A(z_1) \to \overline{\exists^{\mathtt{cl}}} z_2\, A(z_2)\,] \qquad (\overline{\exists^{\mathtt{cl}}}$ introduction)

with the usual restriction that z_2 is not free in B.

Proof: Whereas $\mathtt{Lm}\exists^{\mathtt{cl}}_{+}$ and $\mathtt{Lm}\overline{\exists^{\mathtt{cl}}}_{+}$ are simply immediate after unwinding the definitions (1.5) and (1.9) of $\exists^{\mathtt{cl}}$, respectively $\overline{\exists^{\mathtt{cl}}}$, at $\mathtt{Lm}\exists^{\mathtt{cl}}_{-}$ and $\mathtt{Lm}\overline{\exists^{\mathtt{cl}}}_{-}$ we need to use Lemma 1.40. It is much easier to directly prove in $\mathtt{ML_0}$, respectively $\mathtt{ML}^{\mathtt{nc}}_0$ the variants of $\mathtt{Lm}\exists^{\mathtt{cl}}_{-}$ and $\mathtt{Lm}\overline{\exists^{\mathtt{cl}}}_{-}$ with the rightmost B replaced by $\neg\neg B$. One thereafter uses that $\neg\neg B \to B$ holds in $\mathtt{IL}$, provided that B contains only regular quantifiers (see Lemma 1.40). For this reason, the final complete proofs of $\mathtt{Lm}\exists^{\mathtt{cl}}_{-}$ and $\mathtt{Lm}\overline{\exists^{\mathtt{cl}}}_{-}$ are in $\mathtt{IL}$ and respectively $\mathtt{IL}^{\mathtt{nc}}$. $\Box$

1.3 Weakly extensional intuitionistic Arithmetics $\mathtt{WeZ}$, $\mathtt{WeZ}^{\exists}$, $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$

We now add the basic arithmetical apparatus to our logical systems. We begin by integrating Induction for natural numbers (type ι).

1.3.1 The "no-undischarged-assumptions" Induction rule for naturals

There are very simple realizing terms under Kreisel's Modified Realizability (see [10] and [111]) for the usual Induction Axiom

$\mathtt{IA}$: $\quad A(\mathtt{0}) \to \forall z\,(A(z) \to A(\mathtt{Suc}\, z)) \to \forall z\, A(z) \qquad .$

In contrast, the Dialectica realizing terms for $\mathtt{IA}$ are (far) more complicated since they include the Dialectica-translation of $A(z)$, see Section 1.3.5 below for an explanation of this fact[11]. The usual induction rule

$$\mathtt{IR}: \quad \frac{A(\mathtt{0}) \qquad \forall z\,(A(z) \to A(\mathtt{Suc}\, z))}{\forall z\, A(z)}$$

[11] More direct (but very complicated to display) Dialectica realizers of $\mathtt{IA}$ in a slightly different arithmetical system are presented in [103].

is fully equivalent to $\mathtt{IA}$ (see Section 1.3.5) hence it presents the same problems w.r.t. Dialectica realizability. Moreover, in the context of *light* Dialectica extractability, where input systems may include the $\mathtt{ncm}$ quantifiers, the induction formula $A(z)$ of $\mathtt{IA}$ or $\mathtt{IR}$ would actually be forced not to contain such $\mathtt{ncm}$ quantifiers, see Section 1.3.5 for an explanation.

We therefore choose to use the following variant of the simpler induction rule employed by Jørgensen in [61]:

$$\mathtt{IR}_{\mathrm{o}}: \quad \frac{\begin{array}{cc} \emptyset & \emptyset \\ \vdots & \vdots \\ A(\mathtt{0}) & \forall z\,(A(z) \to A(\mathtt{Suc}\,z)) \end{array}}{\forall z\,A(z)}$$

No restriction concerning the employment of $\mathtt{ncm}$ quantifiers in the induction formula $A(z)$ of $\mathtt{IR}_{\mathrm{o}}$ is needed. On the other hand, a full logical equivalence between $\mathtt{IR}/\mathtt{IA}$ and $\mathtt{IR}_{\mathrm{o}}$ can only be established if $A(z)$ is free of $\mathtt{ncm}$ quantifiers. This is because a $\mathtt{ncm}{-}\mathtt{FC}(\forall z(A(z) \to A(\mathtt{Suc}z)))$ restriction applies in the simulation of $\mathtt{IR}$ in terms of $\mathtt{IR}_{\mathrm{o}}$, see Section 1.3.5 for full details.

1.3.2 The rules of Equality for booleans, naturals and all complex types

We are now at the point of introducing the axioms which give the (usual) behaviour of (higher-order extensional) equality, stressing from the beginning that *extensionality* must[12] be restricted in the context of Dialectica interpretation. The extensionality *axiom* $E_{\sigma,\tau} : \forall z^{\sigma\tau}, x^{\sigma}, y^{\sigma}.\ x =_{\sigma} y \to zx =_{\tau} zy$ must not be derivable in our system. We here deviate from the system $\mathtt{Z}$ of Berger, Buchholz and Schwichtenberg which derives $E_{\sigma,\tau}$ and hence is fully extensional. But let us first present the more basic Reflexivity, Symmetry and Transitivity which we prefer to give directly at higher types and therefore we further prefer to introduce as rules:

[12] See, e.g., the chapter on Dialectica interpretation in [63] for detailed explanations. Howard's original counterexample to the Dialectica realizability of the extensionality axiom $E_{\iota\iota,\iota}$ by Gödel primitive recursive functionals is exposed in [58]. See also [114] for a counterexample to the Dialectica realizability of $E_{\iota\iota,\iota}$ by Van de Pol - Schwichtenberg monotone majorizable functionals (a class intersecting but independent of Gödel's $\mathbf{T}$).

$\mathtt{REF}_\tau$: $x =_\tau x$ (Reflexivity)

$\mathtt{SYM}_\tau$: $x =_\tau y \vdash y =_\tau x$ (Symmetry)

$\mathtt{TRZ}_\tau$: $x =_\tau y\,,\, y =_\tau z \vdash x =_\tau z$ (Transitivity)

Although without computational content under Modified Realizability, the axiom versions of $\mathtt{SYM}_\tau$ and $\mathtt{TRZ}_\tau$ would have required realizing terms under Dialectica interpretation for higher-order τ. We could have used only the axiom versions of $\mathtt{REF}_\iota$, $\mathtt{SYM}_\iota$ and $\mathtt{TRZ}_\iota$ since higher-type Reflexivity, Symmetry and Transitivity can be deduced in pure Minimal Logic from the Reflexivity, Symmetry and respectively Transitivity of natural numbers. The latter are quantifier-free and hence realizer-free under both Realizability and Dialectica interpretations. We however chose the above presentation for the practical reason that proofs are shorter and no Dialectica realizers are needed for the rule versions of higher-order Symmetry and Transitivity.

We do, however, have to deviate from system $\mathtt{Z}$ when it comes to the *Compatibility Axiom* (which implies $E_{\sigma,\tau}$) $x =_\sigma y \to B(x) \to B(y)$ which we replace by the following (strictly) weaker Compatibility Rule:

$$\mathtt{CMP}_\sigma : \quad \frac{\begin{array}{c} A_0 \\ \vdots \\ s =_\sigma t \end{array}}{B(s) \to B(t)}$$

with the restriction that all undischarged assumptions used in the proof of $s =_\sigma t$ (here denoted A_0) are quantifier-free

Had the above restriction[13] not been present, the Compatibility Axiom would be directly deducible by $\to^+$, hence full extensionality would be derivable and Dialectica interpretation would fail to interpret all proofs of our system[14].

Definition 1.44 (Safe term substitution) Let s and r be terms and x be a variable of the same type as r. We define the *safe substitution* of x with r in s, denoted $s[r/x]$, to be the new term obtained by substituting the lambda-free occurrences of x in s with r such that the free variables of r are prevented from getting bound in s by renaming the clashing lambda-bound variables of s to some completely fresh variables of corresponding type.

The last equality rule of our systems is the substitution rule (for terms s, t)

[13] This restriction is not only sufficient but also necessary. Already by allowing purely universal undischarged assumptions in the proof of $s =_\sigma t$ we can deduce the extensionality axiom $E_{\iota\iota,\iota}$ in our system and are therefore subject to Howard's counterexample [58].

[14] Details of the fact that Dialectica is valid for $\mathtt{CMP}$ are given in Chapter 2.

$\mathtt{SUB}_\sigma$: $\quad s[x] =_\sigma t[x] \;\vdash\; s[r/x] =_\sigma t[r/x]$

in which the variable x^τ is safely substituted by the term r^τ.

1.3.3 Equality axioms induced by the conversion relation $\hookrightarrow$

In the end we introduce the *equality axioms*, which are constructed by means of the (iterated) rewrite relation $\hookrightarrow^\star$, that is induced by the following:

Definition 1.45 (Conversion relation $\hookrightarrow$) We denote by $\hookrightarrow$ this conversion relation (and in extenso also its associated one-step reduction relation):

1. the usual rules of α, η and β reduction for simply typed lambda-calculus:

$$\begin{array}{lrcll} (\alpha) & \lambda x.\, t & \hookrightarrow & \lambda y.\, t[y/x] & , \text{ if } y \text{ does not occur free in } t \\ (\eta) & \lambda x.\, (tx) & \hookrightarrow & t & , \text{ if } x \text{ does not occur free in } t \\ (\beta) & (\lambda x.\, t)s & \hookrightarrow & t[s/x] & \end{array}$$

2. for $\mathtt{If}_\tau$ we have $\quad \mathtt{If}_\tau\ \mathtt{tt}\ x^\tau\ y^\tau \hookrightarrow x \quad$ and $\quad \mathtt{If}_\tau\ \mathtt{ff}\ x^\tau\ y^\tau \hookrightarrow y$

3. for the natural numbers we have, as expected,

$$\begin{array}{rclcrcl} \mathtt{Eq}(\mathtt{Suc}\, z, \mathtt{0}) & \hookrightarrow & \mathtt{ff} & \qquad & \mathtt{Eq}(\mathtt{Suc}\, x, \mathtt{Suc}\, y) & \hookrightarrow & \mathtt{Eq}(x, y) \\ \mathtt{R}_\tau\, x\, y\, \mathtt{0} & \hookrightarrow & x & & \mathtt{R}_\tau\, x\, y\, (\mathtt{Suc}\, z) & \hookrightarrow & y(z, \mathtt{R}_\tau\, x\, y\, z) \end{array}$$

We denote by $\hookrightarrow^\star$ the (possibly zero) iterate of the reduction relation $\hookrightarrow$.

Theorem 1.46 (Strong normalization of $\hookrightarrow^\star$) All terms of $\mathcal{T}$ are strongly normalizing w.r.t. the rewrite relation $\hookrightarrow^\star$. This is established in [111] by a proof that uses a variant of Tait's strong computability predicates method.

All sentences $s =_\tau t$ for which $s \hookrightarrow^\star \cdot\, {}^\star\!\hookleftarrow t$ will be added as *equality axioms*:

$\mathtt{AxEQL}$: $\quad s =_\tau t \quad$ for which $\quad s \hookrightarrow^\star \cdot\, {}^\star\!\hookleftarrow t \qquad$ (Equality Axioms)

1.3.4 The definition of weakly extensional Intuitionistic Arithmetics $\mathtt{WeZ}$, $\mathtt{WeZ}^\exists$, $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$

Definition 1.47 The weakly extensional intuitionistic arithmetical systems $\mathtt{WeZ}/\mathtt{WeZ}^{\mathtt{nc}}$ and the corresponding $\mathtt{WeZ}^\exists/\mathtt{WeZ}^{\exists,\mathtt{nc}}$ are obtained by adjoining to $\mathtt{IL}/\mathtt{IL}^{\mathtt{nc}}$ and respectively $\mathtt{IL}^\exists/\mathtt{IL}^{\exists,\mathtt{nc}}$ the following elements:

1. the induction rule $\mathtt{IR}_{\mathtt{o}}$
2. all the above equality axioms, hence $\mathtt{AxEQL}$
3. all the above equality rules, namely $\mathtt{REF}$, $\mathtt{SYM}$, $\mathtt{TRZ}$, $\mathtt{CMP}$ and $\mathtt{SUB}$.

The corresponding minimal arithmetical systems $\mathtt{WeZ}_{\mathtt{0}}/\mathtt{WeZ}_{\mathtt{0}}^{\mathtt{nc}}$ and respectively $\mathtt{WeZ}_{\mathtt{0}}^{\exists}/\mathtt{WeZ}_{\mathtt{0}}^{\exists,\mathtt{nc}}$ are obtained by eliminating the $\mathtt{AxEFQ}$ axiom $\bot \to \mathtt{F}$.

Remark 1.48 Systems $\mathtt{WeZ}_{\mathtt{0}}$ and $\mathtt{WeZ}$ are exactly the weakly extensional versions of Berger, Buchholz and Schwichtenberg's systems $\mathtt{Z}_{\mathtt{0}}$ and respectively $\mathtt{Z}$, see [10]. There as well, system $\mathtt{Z}$ can be obtained by simply adjoining $\bot \to \mathtt{F}$ to the corresponding system $\mathtt{Z}_{\mathtt{0}}$, see Section 4.1.1 for more on this parallel.

Remark 1.49 In particular, $\mathtt{Suc}\, z =_\iota \mathtt{0} \to \bot$ and $\mathtt{Suc}\, x =_\iota \mathtt{Suc}\, y \to x =_\iota y$ follow immediately (in $\mathtt{IL}_{\mathtt{0}}$ already) from the definition of $=_o$.

Remark 1.50 We stayed as close as possible to the axiomatic [111, 10] of system $\mathtt{Z}$ in order to render easier the task of implementing program-extraction by (light) Dialectica interpretation in $\mathtt{MinLog}$ [116].

Lemmas of $\mathtt{WeZ}_{\mathtt{0}}$ which necessarily require the Rewrite Relation $\hookrightarrow$

The $\mathtt{WeZ}_{\mathtt{0}}$ lemmas $\mathtt{LmO}\iota : \mathtt{O}^{\sigma\iota}\underline{z}^{\underline{\sigma}} =_\iota \mathtt{0}$ and $\mathtt{LmOo} : \mathtt{O}^{\sigma o}\underline{z}^{\underline{\sigma}} =_o \mathtt{ff}$ describe the behaviour of the zero terms, see Definition 1.3. Recall that $\rceil A \equiv A \to \mathtt{at}(\mathtt{ff})$. The expected behaviour of the multiple selector $\mathtt{If}^n_\tau$ (defined by (1.4)) is given by the following $n+1$ lemmas of $\mathtt{WeZ}_{\mathtt{0}}$ grouped under the name $\mathtt{LmIf}^n_\tau$:

$$\left.\begin{array}{rcl}
\rceil\mathtt{at}(p_1) & \to & \mathtt{If}^n_\tau(p_1,\dots,p_n,x_{n+1},x_n,\dots,x_1) =_\tau x_1 \\
\mathtt{at}(p_1)\wedge\rceil\mathtt{at}(p_2) & \to & \mathtt{If}^n_\tau(p_1,\dots,p_n,x_{n+1},x_n,\dots,x_1) =_\tau x_2 \\
 & \vdots & \\
\wedge_{j=1}^{i-1}\mathtt{at}(p_j)\wedge\rceil\mathtt{at}(p_i) & \to & \mathtt{If}^n_\tau(p_1,\dots,p_n,x_{n+1},x_n,\dots,x_1) =_\tau x_i \\
 & \vdots & \\
\wedge_{j=1}^{n-1}\mathtt{at}(p_j)\wedge\rceil\mathtt{at}(p_n) & \to & \mathtt{If}^n_\tau(p_1,\dots,p_n,x_{n+1},x_n,\dots,x_1) =_\tau x_n \\
\wedge_{i=1}^{n}\mathtt{at}(p_i) & \to & \mathtt{If}^n_\tau(p_1,\dots,p_n,x_{n+1},x_n,\dots,x_1) =_\tau x_{n+1}
\end{array}\right\} \quad (1.35)$$

Definition 1.51 (Schönfinkel-Curry combinators) For all types δ,ρ,τ, the combinators $\Sigma_{\delta,\rho,\tau}$ of type $(\delta\rho\tau)(\delta\rho)\delta$ and $\Pi_{\rho,\tau}$ of type $\rho\tau\rho$, are terms of $\mathcal{T}$ defined by means of lambda-abstraction as:

$$\Sigma_{\delta,\rho,\tau} \;:\equiv\; \lambda x^{\delta\rho\tau}, y^{\delta\rho}, z^{\delta}.\, xz(yz) \qquad \Pi_{\rho,\tau} \;:\equiv\; \lambda x^{\rho}, y^{\tau}.\, x$$

Definition 1.52 A *combinatorial term* is a λ-term which is built by application only starting from (free) variables, constants and Σ and Π combinators.

Proposition 1.53 (Lambda-terms to combinatorial terms) Every term $t[x_1,\ldots,x_k]$ with $x_1,\ldots,x_k$ all its free variables can be written as a combinatorial term $\widetilde{t}[x_1,\ldots,x_k]$, in the sense that $\mathtt{WeZ} \vdash t = \widetilde{t}$. We will denote by $\widetilde{\cdot}: t \mapsto \widetilde{t}$ this association which is given by a syntactic recursive algorithm.

Proof: The association algorithm proceeds by recursion on the structure of the input term t, from which it eliminates the lambda-abstractions in a bottom-up fashion, via the following transformation steps:

$$\left.\begin{array}{rcl} \lambda x.\, x & \mapsto & \Sigma\,\Pi\,\Pi \\ \lambda x.\, t & \mapsto & \Pi t\ , \quad \text{if } x \text{ not free in } t \\ \lambda x.\, ts & \mapsto & \Sigma(\lambda x.\, t)(\lambda x.\, s)\ , \quad \text{if } x \text{ is free in } ts \end{array}\right\} \tag{1.36}$$

In this translation, the combinators Σ and Π are treated as constants and not as lambda-terms! Hence the bottom-most original λ-abstractions are first eliminated via the translation (1.36) which proceeds in a top-down fashion. One then goes bottom-up in the original lambda-term. One thus ensures that each time (1.36) is applied, the abstracted term is combinatorial. $\Box$

1.3.5 Equivalence between formulations of Induction

Theorem 1.54 The induction axiom $\mathtt{IA}$ and the induction rule $\mathtt{IR}$ are fully equivalent over minimal logic $\mathtt{ML^{nc}}$. Equivalence in $\mathtt{ML^{nc}}$ between the full $\mathtt{IR}$ and the more restricted $\mathtt{IR_o}$ can be proved only if the open assumptions of the second premise $\forall z(A(z) \to A(\mathtt{Suc}\,z))$ of $\mathtt{IR}$ do respect the $\mathtt{ncm-FC}$ restriction.

Proof: We can produce the following proof of $\mathtt{IA}$ from $\mathtt{IR}$:

$$\dfrac{\dfrac{\dfrac{[2:A(\mathtt{0})]}{A(\mathtt{0})} \qquad \dfrac{[1:\forall z\,(\,A(z) \to A(\mathtt{Suc}\,z)\,)]}{\forall z\,(\,A(z) \to A(\mathtt{Suc}\,z)\,)}}{\forall z\,A(z)}\,\mathtt{IR}}{A(\mathtt{0}) \to \forall z\,(A(z) \to A(\mathtt{Suc}\,z)) \to \forall z\,A(z)}\,{\to}^{+}_{1,2}$$

We now show how $\mathtt{IR}$ can be simulated in terms of $\mathtt{IR_o}$ provided that the restriction on the open assumptions of $\forall z(A(z) \to A(\mathtt{Suc}z))$ is fulfilled. During this process it will become obvious that such restriction cannot be avoided in

any attempt of such simulation. Let $\mathcal{P}$ be a proof (in some system extending $\mathtt{WeZ}$) whose last rule is $\mathtt{IR}$. Let C be the set of all open assumptions of $\mathcal{P}$. Let $C = C_b \cup C_s$ be the split of C such that C_b is the set of assumptions of the $A(\mathtt{0})$ of $\mathtt{IR}$ and C_s is the set of assumptions of the $\forall z(A(z) \to A(\mathtt{Suc}\, z))$ of $\mathtt{IR}$. We assume that the elements of C_b and C_s are given in some arbitrary but fixed order from left to right such that C is associated their composed order with the elements of C_b at left followed by the elements of C_s at right. We can ensure (by, e.g., renaming it) that the bound variable z of the conclusion $\forall z A(z)$ of $\mathcal{P}$ is not free in C. We now transform $\mathcal{P}$ to $\mathcal{P}'$ such that $\mathcal{P}'$ employs the same open assumptions C of $\mathcal{P}$ but uses an $\mathtt{IR}_{\mathrm{o}}$ to simulate the last $\mathtt{IR}$ of $\mathcal{P}$. We also try to make this simulation as efficient as possible, in the sense of efficiency of the extracted programs, by employing only the strictly necessary contractions. Such contractions, if they exist, are only due to a number of $\to^+$ with 2-contraction[15], namely one for each of the elements of C_s, as we explain in the sequel. Let $\mathcal{P}_b$ be the proof of the $A(\mathtt{0})$ of $\mathtt{IR}$ and $\mathcal{P}_s$ be the proof of the $\forall z(A(z) \to A(\mathtt{Suc}\, z))$ of $\mathtt{IR}$. From $\mathcal{P}_b$ we can immediately produce the proof $\mathcal{P}_1 \colon\ \emptyset \vdash C_b \to (C_s \to A(\mathtt{0}))$ by using a number of $\to^+$ with 0-contraction over the elements of C_s, followed by a number of $\to^+$ with 1-contraction over the elements of C_b. We now build from $\mathcal{P}_s$ a proof $\mathcal{P}_2 \colon \vdash\ \forall z.\ (C_b \to (C_s \to A(z))) \to (C_b \to (C_s \to A(\mathtt{Suc} z)))$ which will require contractions over the elements of C_s only. We first build the proof

$$\dfrac{\dfrac{\boxed{C_s}}{A(z)} \qquad \dfrac{\dfrac{C_b \quad C_b \to (C_s \to A(z))}{C_s \to A(z)}}{A(z)}}{A(\mathtt{Suc} z)} \qquad \dfrac{\left.\begin{array}{c}\boxed{C_s}\\ \vdots \\ \forall z.\, A(z) \to A(\mathtt{Suc} z)\end{array}\right\} \mathcal{P}_s}{A(z) \to A(\mathtt{Suc} z)}$$

Since (by hypothesis) the $\mathtt{ncm{-}FC}$ restriction holds for all formulas of C_s, we can discharge all open assumptions from C_s by a sequence of $|C_s|$ $\to^+$ with 2-contraction. Subsequently, we also cancel all assumptions from C_b, this time with 1-contraction (hence without contraction, recall Definition 1.17). We thus obtain a proof $C_b \to (C_s \to A(z)) \vdash C_b \to (C_s \to A(\mathtt{Suc} z))$ to which we apply a last $\to^+$ with 1-contraction over $C_b \to (C_s \to A(z))$ and finally obtain,

[15] These contractions are unavoidable in the simulation of $\mathtt{IR}$ in terms of $\mathtt{IR}_{\mathrm{o}}$. On the other hand, contractions over the assumptions in C_b would be redundant, reason why splitting C into C_b. and C_s.

after a $\forall_z^+$ – since $\mathtt{VC}_1(z)$ is obviously satisfied – the following proof:

$$\mathcal{P}_2\left\{\begin{array}{c}\emptyset\\ \vdots\\ \dfrac{(C_b \to (C_s \to A(z))) \to (C_b \to (C_s \to A(\mathtt{Suc}z)))}{\forall z.\,(C_b \to (C_s \to A(z))) \to (C_b \to (C_s \to A(\mathtt{Suc}z)))}\ \forall_z^+\end{array}\right.$$

We now apply $\mathtt{IR}_0$ to $\mathcal{P}_1$ and $\mathcal{P}_2$ and obtain a proof $\mathcal{P}_3\colon\ \forall z(C_b \to C_s \to A(z))$. To $\mathcal{P}_3$ we apply a $\forall_z^-$ and then create new open assumptions C in the reversed order and apply a number of $|C| \to^-$ in order to obtain a proof $\mathcal{P}_4\colon\ C \vdash A(z)$. Since we ensured that z is not free in C we can apply a final $\forall_z^+$ and obtain the proof $\mathcal{P}'\colon\ C \vdash \forall z\, A(z)$.

For closing the equivalences we mention that $\mathtt{IR}$ is immediate from $\mathtt{IA}$ by means of two $\to^-$ and $\mathtt{IR}_0$ is just a particular case of $\mathtt{IR}$. □

1.4 Immediate extension of $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$

We present below the extension of our arithmetics $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ with a number of axioms which have simple realizers *directly* under the light functional "Dialectica" interpretation (defined in Chapter 2). Originally (i.e., for the pure Dialectica interpretation, see, e.g., [68, 63, 57]) three such principles are considered for inclusion in the systems directly interpreted by the Gödel $\mathtt{D}$-interpretation. They all contain the strong $\exists$ and are not provable in $\mathtt{WeZ}^{\exists}$. The first two principles are logically deducible in the fully classical version of $\mathtt{WeZ}^{\exists}$ (namely $\mathtt{WeZ}^{\exists,\mathtt{cl}}$, obtained by adding *full* stability $\neg\neg A \to A$ for formulas A which may contain $\exists$, see Section 1.6) but not in $\mathtt{WeZ}^{\exists}$. These "semi-classical" logical axioms are Markov's Principle[16] $\exists^{\mathtt{cl}} z\, A_0(z) \to \exists z\, A_0(z)$ (here A_0 is quantifier-free as usual) and Independence of premises for purely universal premises $[\forall x\, A_0(x) \to \exists y\, B(y)] \to \exists y\, [\forall x\, A_0(x) \to B(y)]$ (here B is an unrestricted formula). The third principle is the non-logical (i.e., not logically deducible in $\mathtt{WeZ}^{\exists,\mathtt{cl}}$) Axiom of Choice (below B is unrestricted as well):

$$\mathtt{AxAC}: \qquad \forall x\, \exists y\, B(x,y) \to \exists Y\, \forall x\, B(x, Y(x)) \qquad .$$

[16] The usual formulation of Markov's principle as $\mathtt{AxMK}_1: \neg\neg\exists z\, A_0(z) \to \exists z\, A_0(z)$ is equivalent to $\mathtt{AxMK}$ over $\mathtt{WeZ}^{\exists}$. The same statement holds for the form $\mathtt{AxMK}_2: \neg\neg\exists z\, A_0(z) \to \exists z \neg\neg A_0(z)$ which was preferred by the authors of [57] because of complexity considerations. Our choice of $\mathtt{AxMK}$ is motivated by the particularity of $\exists^{\mathtt{d}}$ in the development of the $\mathtt{BBS}$ refined $\mathtt{A}$-translation from [10, 111].

Whereas for AxAC the addition of the ncm quantifiers $\overline{\forall}$ and $\overline{\exists}$ would not bring a real change, in the sense that the light Dialectica interpretability of AxAC can be proved in the same lines as its pure D-interpretability (since B used to be unrestricted anyway in AxAC), for the Markov's principle and the Independence of premises concrete extensions are possible, as we show in the sequel. For Markov's principle, this holds as far as one accepts that the final verifying proofs are not necessarily constructive. Since this thesis is far more concerned with program extraction rather than the translation of classical to constructive systems, we content ourselves with the obtaining of fully classical verifying proofs. We therefore use the following variant of Markov's principle:

$\mathtt{AxMK}$: $\quad \exists^{\mathtt{cl}} z\, A^{\mathtt{nc}}(z) \to \exists z\, A^{\mathtt{nc}}(z)$

where $A^{\mathtt{nc}}$ is a purely ncm formula, in the sense that all its quantifiers are ncm (but there is no restriction on their number or position in the formula, hence an unrestricted formula, but in the ncm realm)[17]. It is not difficult to guess that in the verifying proofs the principle $\neg\neg A \to A$ will be necessary, where A is the literal translation of $A^{\mathtt{nc}}$ to its regular-quantifier form. Since $A^{\mathtt{nc}}$ was ncm-unrestricted, this forces that the verifying system is fully classical! See the proof of Theorem 2.10 from Section 2.1 for details. Once we accept full stability in the verifying proofs we can also include "for free" in the input systems the following pure-ncm variant of Stability:

$\mathtt{AxSTAB}^{\mathtt{nc}}$: $\quad \neg\neg\, A^{\mathtt{nc}} \to A^{\mathtt{nc}}$ (Pure-ncm Stability)

Its LD-interpretation will simply be the full Stability $\neg\neg A \to A$ (in the system without ncm), no realizing terms: $\mathtt{AxSTAB}^{\mathtt{nc}}$ is a purely logical construct, which will be necessary in the proof of soundness of Kuroda's negative translation.

Whereas the full "Independence of premises" axiom scheme, obtained by replacing $\forall x A_0(x)$ with unrestricted formulas A, although provable in $\mathtt{WeZ}^{\exists,\mathtt{cl}}$, fails to get any direct (light) Dialectica interpretation, it can be immediately seen that the following more general "Independence of premises for quasi-universal premises" scheme does have such an interpretation:

$\mathtt{AxIP}^{-\exists}_{+\forall}$: $\quad [\, A^{-\exists}_{+\forall} \to \exists y\, B(y)\,] \to \exists y\, [\, A^{-\exists}_{+\forall} \to B(y)\,]$,

where $A^{-\exists}_{+\forall}$ does not contain any positive $\exists$ or negative $\forall$, see the proof of

[17] We will here-on use notations like $A^{\mathtt{nc}}$, $B^{\mathtt{nc}}$, etc. for this kind of formulas only. In the same context, A, B, etc. will denote the literal translations of $A^{\mathtt{nc}}$, $B^{\mathtt{nc}}$, etc. to their regular-quantifier form, obtained by transforming each ncm quantifier to its corresponding regular quantifier.

Theorem 2.10 in the sequel for its direct light Dialectica interpretation.

Now the question is how these $\exists$ principles might be simulated in the systems without $\exists$, hence $\mathtt{WeZ}$ and $\mathtt{WeZ}^{\mathtt{nc}}$. In the absence of $\exists$, the only logical possibility appears to be the replacement of $\exists$ with $\exists^{\mathtt{cl}}$ and similarly of $\overline{\exists}$ with $\overline{\exists^{\mathtt{cl}}}$ in the case of $\mathtt{WeZ}^{\mathtt{nc}}$. This seems not to function for the general $\mathtt{AxAC}$, but it certainly works for its restriction that B does not contain any regular quantifier, see also Section 2.3. We can thus extend $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ with

$$\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}} : \quad \forall x\, \exists^{\mathtt{cl}} y\, B^{\mathtt{nc}}(x,y) \to \exists^{\mathtt{cl}} Y\, \forall x\, B^{\mathtt{nc}}(x, Y(x)) \quad ,$$

where $B^{\mathtt{nc}}$ may contain $\mathtt{ncm}$ quantifiers only. $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ is not provable in $\mathtt{WeZ}^{\exists,\mathtt{nc}}$.

Remark 1.55 Also the somewhat weaker $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{o}}$ is not provable in $\mathtt{WeZ}$, where

$$\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{o}} : \quad \forall x\, \exists^{\mathtt{cl}} y\, B_0(x,y) \to \exists^{\mathtt{cl}} Y\, \forall x\, B_0(x, Y(x))$$

(above B_0 is quantifier-free, as usual). Notice that $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{o}}$ is the restriction of $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ to the language $\mathcal{L}$.

Of course, the transformation of $\exists$ into $\exists^{\mathtt{cl}}$ would make $\mathtt{AxMK}$ hold in a trivial way. Whereas $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{o}}$ and $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ are not provable in $\mathtt{WeZ}$, respectively $\mathtt{WeZ}^{\mathtt{nc}}$, the situation is different for

$$\mathtt{AxIP}^{\mathtt{cl}} : \quad [\, A \to \exists^{\mathtt{cl}} y\, B(y)\,] \to \exists^{\mathtt{cl}} y\, [\, A \to B(y)\,] \quad ,$$

which can be proved to hold in $\mathtt{WeZ}$ for unrestricted $A, B \in \mathcal{F}^{\exists}$. Indeed, a proof of $\mathtt{AxIP}^{\mathtt{cl}}$ amounts to proving $\bot$ from the assumptions $\forall y.[A \to B(y)] \to \bot$ and $[\, A \to (\forall y.B(y) \to \bot) \to \bot\,]$, which can be established if one proves $[\, \neg\neg A \wedge (\forall y.B(y) \to \bot)\,]$ from the assumption $\forall y.[A \to B(y)] \to \bot$. The proof of $(\forall y.B(y) \to \bot)$ is simply straightforward, due to the easy theorem $B(y) \to (A \to B(y))$. Also the formula $\neg\neg A$ is immediately provable from $\forall y.[A \to B(y)] \to \bot$ by using the ex-falso-quodlibet scheme $\bot \to B(y)$. We thus obtain proofs of $\neg\neg A$ and of $\neg A$, hence of $\bot$. Notice that the assumption $\forall y.[A \to B(y)] \to \bot$ was used twice in the proof above. The global proof of $\mathtt{AxIP}^{\mathtt{cl}}$ cannot avoid contraction over this assumption, which unfortunately is computationally $\mathtt{LD}$-relevant, since $\forall y$ appears in a positive position. Due to the $\mathtt{ncm}-\mathtt{FC}$ restriction (see Definition 1.15), both formulas A and B may not contain $\mathtt{ncm}$ quantifiers. Hence $\mathtt{AxIP}^{\mathtt{cl}}$ cannot be proved in $\mathtt{WeZ}^{\mathtt{nc}}$ or $\mathtt{WeZ}^{\exists,\mathtt{nc}}$.

However, direct light Dialectica realizers can be given for the following variant of $\mathtt{AxIP}^{\mathtt{cl}}$ in which A and B can freely use the $\mathtt{ncm}$ quantifiers, but A is restricted to be quasi-universal and B is restricted to be quasi-existential:

$$\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}} : \quad [\, A^{-\exists}_{+\forall} \to \exists^{\mathtt{cl}} y\, B^{+\exists}_{-\forall}(y)\,] \to \exists^{\mathtt{cl}} y\, [\, A^{-\exists}_{+\forall} \to B^{+\exists}_{-\forall}(y)\,] \quad ,$$

where $B^{+\exists}_{-\forall}$ does not contain any positive $\forall$ or negative $\exists$, and $A^{-\exists}_{+\forall}$ may only contain positive $\forall$ and negative $\exists$, equivalent definition as at $\mathtt{AxIP}^{-\exists}_{+\forall}$ above.

In the end we consider a straightforward extension of the nc-systems $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ which will induce an isomorphic extension of their correspondents without ncm quantifiers WeZ and respectively $\mathtt{WeZ}^{\exists}$. This is with an arbitrary but fixed set of sentences Π of $\mathcal{F}^{\exists,\mathtt{nc}}$ of shape $\forall \underline{b} B^{\mathtt{nc}}(\underline{b})$, where $B^{\mathtt{nc}}(\underline{b})$ may contain ncm quantifiers only (see Footnote 17 above). These sentences Π are added as axioms to $\mathtt{WeZ}^{\mathtt{nc}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}$. Let $\widetilde{\Pi}$ denote the set of formulas of shape $\forall \underline{b} B(\underline{b})$ which correspond bijectively to the formulas $\forall \underline{b} B^{\mathtt{nc}}(\underline{b})$ of Π, where B is obtained from $B^{\mathtt{nc}}$ by transforming each ncm quantifier to its corresponding regular quantifier. Then sentences $\widetilde{\Pi}$ are isomorphically added as axioms to WeZ and respectively $\mathtt{WeZ}^{\exists}$. Notice that $\widetilde{\Pi}$ is exactly the image of Π under the Light Dialectica translation of formulas (LD-translation defined in Section 2.1, see Definition 2.1).

Remark 1.56 Axioms Π do not interfere with the Light Dialectica extraction process. Their LD-translations are bijectively gathered in the corresponding set $\widetilde{\Pi}$ and are isomorphically used in the Light Dialectica verifying proofs.

Convention 1.57 From here on we consider that systems $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ and $\mathtt{WeZ}^{\mathtt{nc}}$ are automatically extended with an arbitrary but fixed set Π like described above. When used in the same context, also the systems $\mathtt{WeZ}^{\exists}$ and respectively WeZ will include the set $\widetilde{\Pi}$ of axioms which correspond to the Π included in $\mathtt{WeZ}^{\exists,\mathtt{nc}}$, respectively $\mathtt{WeZ}^{\mathtt{nc}}$. This situation will extend to the monotonic and classical systems based on $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ which are to be introduced is Sections 2.2 and 2.3 in the sequel.

Definition 1.58 (The extended constr. systems $\mathtt{WeZ}^{\mathtt{nc}+}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$) We denote by $\mathtt{WeZ}^{\mathtt{nc}+}$ the extension of $\mathtt{WeZ}^{\mathtt{nc}}$ with $\mathtt{AxSTAB}^{\mathtt{nc}}$, $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$ and $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ (all principles restricted to $\mathcal{L}^{\mathtt{nc}}$) and by $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ the extension of $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ with $\mathtt{AxSTAB}^{\mathtt{nc}}$, AxMK, $\mathtt{AxIP}^{-\exists}_{+\forall}$, $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$, AxAC and $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$.

1.5 The monotonic intuitionistic Arithmetics $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$, $\mathtt{WeZ}^{\mathtt{nc}}_{\mathtt{m}}$, $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$, $\mathtt{WeZ}^{\mathtt{nc}+}_{\mathtt{m}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}$

Definition 1.59 (Monotonic languages, terms and formulas) The monotonic languages $\mathcal{L}_{\mathtt{m}}$, $\mathcal{L}^{\exists}_{\mathtt{m}}$, $\mathcal{L}^{\mathtt{nc}}_{\mathtt{m}}$ and the most complete $\mathcal{L}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ are obtained by adding to $\mathcal{L}$, $\mathcal{L}^{\exists}$, $\mathcal{L}^{\mathtt{nc}}$ and respectively $\mathcal{L}^{\exists,\mathtt{nc}}$ the following two new constants:

1. $\mathtt{Geq}$ – a functional *inequality* constant for naturals, of type $\iota\iota o$, and
2. $\mathtt{Max}$ – a *maximum* constant for naturals, of type $\iota\iota\iota$.

We will also denote by $\mathcal{T}_{\mathtt{m}}$ the new set of terms, built using also $\mathtt{Geq}$ and $\mathtt{Max}$ and by $\mathcal{F}_{\mathtt{m}}$, $\mathcal{F}_{\mathtt{m}}^{\exists}$, $\mathcal{F}_{\mathtt{m}}^{\mathtt{nc}}$ and $\mathcal{F}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ the respectively corresponding sets of formulas.

Definition 1.60 (Predicate inequality at all types)
We define predicate inequality at type ι, denoted $\geq_\iota$, as an abbreviation

$$s^\iota \geq_\iota t^\iota \quad :\equiv \quad \mathtt{at}(\mathtt{Geq}\, s\, t)$$

We define boolean inequality $\geq_o$ between terms s, t of type o by

$$s \geq_o t \quad :\equiv \quad \mathtt{at}(t) \to \mathtt{at}(s)$$

Similar to equality, predicate inequality at higher types $\tau \equiv \underline{\sigma}\rho$ with $\rho \in \{\iota, o\}$ is extensionally defined as

$$s \geq_\tau t \quad :\equiv \quad \forall x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\, (s\, x_1 \ldots x_n \geq_\rho t\, x_1 \ldots x_n) \tag{1.37}$$

Definition 1.61 (Howard's majorization relation) We define Howard's *majorization relation*, which is an open formula of $\mathcal{F}_{\mathtt{m}}$, by

$$\begin{aligned} x^\rho \succeq_\rho y^\rho \quad &:\equiv \quad x^\rho \geq_\rho y^\rho \ \text{ for } \ \rho \in \{\iota, o\} \quad \text{and} \\ x^{\sigma\tau} \succeq_{\sigma\tau} y^{\sigma\tau} \quad &:\equiv \quad \forall z_1^\sigma, z_2^\sigma\, (z_1 \succeq_\sigma z_2 \ \to \ x z_1 \succeq_\tau y z_2) \end{aligned} \tag{1.38}$$

Remark 1.62 Although the majorization results we are interested in concern mainly the natural numbers, it appears technically useful to have an inequality relation for booleans as well. The following hold in $\mathtt{WeZ}^{\exists}$ for any term s^o

$$s \geq_o \mathtt{ff} \quad \text{and} \quad \mathtt{tt} \geq_o s \tag{1.39}$$

Definition 1.63 We say that a type is *arithmetic* if it decomposes as $\underline{\sigma}\iota$ and *boolean* if it writes as $\underline{\sigma}o$. The definition extends to variables, constants and terms in the obvious way.

Definition 1.64 (Systems $\mathtt{WeZ}_{\mathtt{m}}$,$\mathtt{WeZ}_{\mathtt{m}}^{\exists}$,$\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$,$\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$,$\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$ and $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+}$)
The monotonic arithmetics $\mathtt{WeZ}_{\mathtt{m}}$, $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$, $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$ and $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$, the immediate extensions (in the sense of Section 1.4) $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$ and (the most complete) $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+}$ are obtained by adding to systems $\mathtt{WeZ}$, $\mathtt{WeZ}^{\exists}$, $\mathtt{WeZ}^{\mathtt{nc}}$, $\mathtt{WeZ}^{\exists,\mathtt{nc}}$, $\mathtt{WeZ}^{\mathtt{nc}+}$ and respectively $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ the following axioms and rules:

1. In the ncm systems, the ncm$-$FC restriction is withdrawn, hence the (computationally LD-relevant) contraction formulas may (freely, arbitrarily) contain ncm quantifiers. In consequence, also the variable condition $\mathtt{VC}_3(z,\mathcal{P})$ set on the ncm-ForAll introduction rule $\dfrac{\mathcal{P}\colon A(z)}{\overline{\forall} z\, A(z)}\ \overline{\forall}^{+}_{z}$ is relaxed to just (our formulation of) Berger's restriction[18], see also [9].

2. To the non-ncm systems $\mathtt{WeZ}_{\mathtt{m}}$ and $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$, the following axiom schema of full comprehension is added (for pure syntactic verification purposes):
   $\mathtt{AxCA}: \quad \exists \Phi^{\tau\to o}\, \forall x^{\tau}\, [\, \mathtt{at}(\Phi x) \leftrightarrow B(x)\,]$
   for arbitrary formulas $B(x^{\tau})$ in the corresponding language $\mathcal{L}_{\mathtt{m}}$, respectively $\mathcal{L}^{\exists}_{\mathtt{m}}$ and arbitrary tuples x of variables of suitable arbitrary types τ.

3. As a consequence of 1. above, a number of principles which were not provable in $\mathtt{WeZ}^{\mathtt{nc}}$ become provable in $\mathtt{WeZ}^{\mathtt{nc}}_{\mathtt{m}}$, hence it would be redundant to include them in $\mathtt{WeZ}^{\mathtt{nc}+}_{\mathtt{m}}$. These are $\mathtt{AxSTAB}^{\mathtt{nc}}$ and $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$. In fact the full stability AxSTAB and the full classical Independence of premises $\mathtt{AxIP}^{\mathtt{cl}}$ become provable in $\mathtt{WeZ}^{\mathtt{nc}}_{\mathtt{m}}$. Beware that $\mathtt{WeZ}^{\mathtt{nc}}_{\mathtt{m}}$ does not contain $\exists$ nor $\overline{\exists}$.

4. As a consequence of 1., $\mathtt{AxIP}^{\mathtt{cl}}$ is provable also in $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$. Therefore $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}$ will not include the now redundant $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$. On the other hand it still makes sense to retain $\mathtt{AxSTAB}^{\mathtt{nc}}$ into $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}$. This is because $\mathtt{AxSTAB}^{\mathtt{nc}}$ is provable in $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ only for formulas which do not contain $\overline{\exists}$ in a strictly positive position, see Corollary 1.41 and Lemma 1.40.

5. $\mathtt{WeZ}^{\mathtt{nc}+}_{\mathtt{m}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}$ still include $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$, which is not provable in $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$.

6. One adds the axioms defining the *maximum* constant $\mathtt{Max}^{\iota\iota\iota}$:
   $$\mathtt{Max}\, x\, y \ \geq_{\iota}\ x \qquad \mathtt{Max}\, x\, y \ \geq_{\iota}\ y \qquad \mathtt{Max} \ \succeq_{\iota\iota\iota}\ \mathtt{Max} \quad .$$

7. Like in Section 1.3.2 for $=_{\tau}$, we here provide similar rules for the higher-type *reflexivity* $\vdash x \geq_{\tau} x$ and *transitivity* $x \geq_{\tau} y, y \geq_{\tau} z \vdash x \geq_{\tau} z$ of "predicate" inequality [19]. We also add the rule $x =_{\tau} y \vdash x \geq_{\tau} y$ which establishes the compatibility between equality and "greater-than-or-equal"

[18] Namely that z does not occur free in any of the instantiating terms t involved by a $\forall^{-}_{\bullet,t}$ in the proof $\mathcal{P}$, except for the cases when at some point in $\mathcal{P}$ subsequent to such a $\forall^{-}$, the variable z gets to no longer be free in any of the uncancelled assumption or conclusion formulas of such a sub-proof of $\mathcal{P}$.

[19] An anti-symmetry rule $x \geq_{\tau} y, y \geq_{\tau} x \vdash x =_{\tau} x$ may also be added, but that's not necessary for the proof of Theorem 2.12 below.

8. Also the following rewrite rules are added to the relation $\hookrightarrow$ defined in Section 1.3 in order to attain the usual behaviour of inequality and $\mathtt{Max}$:

$$\begin{array}{lllllllllll}
\mathtt{Geq}(z,\mathtt{0}) & \hookrightarrow & \mathtt{tt} & \quad & \mathtt{Geq}(\mathtt{0},\mathtt{Suc}z) & \hookrightarrow & \mathtt{ff} & \quad & \mathtt{Geq}(\mathtt{Suc}x,\mathtt{Suc}y) & \hookrightarrow & \mathtt{Geq}(x,y) \\
\mathtt{Max}(z,\mathtt{0}) & \hookrightarrow & z & \quad & \mathtt{Max}(\mathtt{0},z) & \hookrightarrow & z & \quad & \mathtt{Max}(\mathtt{Suc}x,\mathtt{Suc}y) & \hookrightarrow & \mathtt{Max}(x,y)
\end{array}$$

The addition of full comprehension $\mathtt{AxCA}$ to the verifying systems $\mathtt{WeZ}_{\mathtt{m}}$ and $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ is meant to compensate the elimination of the $\mathtt{ncm}-\mathtt{FC}$ restriction (see Definition 1.15) from the input $\mathtt{ncm}$ monotone systems. We will see in the proof of Theorem 2.12 that arbitrary formulas of $\mathtt{WeZ}_{\mathtt{m}}/\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ must be uniformly equivalent (via $\mathtt{at}$) to corresponding boolean terms in order to prove the (possibly non-effective!) existence of realizers which are majorized by the monotone extracted terms. Even though the radical/root of the $\mathtt{LD}$-translation of a (relevant, see Definition 1.18) contraction formula is no longer included in the extracted terms (see Remark 1.16, Remark 2.2, Definition 2.6 and the proof of Theorem 2.10), one still needs its comprehension property in order to prove the (possibly weak) existence of terms which are majorized. In the absence of $\mathtt{ncm}-\mathtt{FC}$, these radical/root formulas are no longer guaranteed to be quantifier-free, which was a sufficient condition for having the uniform boolean term association of Proposition 1.35, see Definition 2.6. Hence a strong-hand external enforcement via the powerful axiom $\mathtt{AxCA}$ of the comprehension property for all formulas in the verifying (translated) system appears to be necessary. In consequence, not only $\mathtt{WeZ}_{\mathtt{m}}$ is a fully classical system (due to Lemma 1.40), but also $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ can prove full stability $\neg\neg A \to A$ for unrestricted formulas $A \in \mathcal{F}_{\mathtt{m}}^{\exists}$, using Corollary 1.28 applied to the boolean comprehension term $\Phi(\underline{a})$ associated to $A(\underline{a})$ by means of $\mathtt{AxCA}$. Thus $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ subsumes full classical logic.

Proposition 1.65 (Useful nice properties of $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$, $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$ and $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$) System $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ features full stability $\neg\neg A \to A$ for all $A \in \mathcal{F}_{\mathtt{m}}^{\exists}$, and also $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$ enjoys full stability for all $A \in \mathcal{F}_{\mathtt{m}}^{\mathtt{nc}}$. System $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ can prove stability for all those $A \in \mathcal{F}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ which do not contain $\exists$ or $\overline{\exists}$ in a strictly positive position. Note that also (1.31) and (1.32) hold freely in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$ for all formulas in $\mathcal{F}_{\mathtt{m}}^{\exists,\mathtt{nc}}$.

Proof: As explained in Definition 1.64, all this is natural once the $\mathtt{ncm}-\mathtt{FC}$ restriction is eliminated. The situations involving the $\mathtt{ncm}$ quantifiers are thus identical now to the cases of the corresponding regular quantifiers. The here abbreviated $\mathtt{VC}_3(\cdot)$ restriction does not intervene in these proofs. $\square$

Convention 1.66 Notice that the nc-systems $\mathtt{WeZ}^{\mathtt{nc}}_{\mathtt{m}}$ and $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ do also include the set of axiom sentences Π which they inherit from $\mathtt{WeZ}^{\mathtt{nc}}$ and respectively $\mathtt{WeZ}^{\exists,\mathtt{nc}}$. In the same context, also $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ comprises the corresponding set of axiom sentences $\widetilde{\Pi}$, which is inherited through $\mathtt{WeZ}^{\exists}$. See Convention 1.57.

Definition 1.67 (Δ axiom extension of monotonic systems) Moreover, to the monotonic system $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ one further adds as axioms all elements of an arbitrary but fixed set Δ of sentences of shape

$$\forall x^{\rho}\ \exists y \leq_{\sigma} rx\ \forall z^{\tau}\ B^{\mathtt{nc}}(x,y,z)$$

where $B^{\mathtt{nc}}$ is a formula whose quantifiers (if any) are all ncm, whose free variables are all exactly x, y, z and such that $r^{\rho\sigma}$ is a closed term. One also adds as axioms to $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ the elements of the isomorphic set $\widetilde{\Delta}$ of sentences

$$\exists Y \leq_{\rho\sigma} r\ \forall x^{\rho}\ \forall z^{\tau}\ B(x, Yx, z)$$

which correspond bijectively to the elements of Δ in the sense that formula B is the LD-translation of $B^{\mathtt{nc}}$ [20]. Hence when used in the same context, $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ includes exactly the axiom set $\widetilde{\Delta}$ which LD-corresponds to the axiom set Δ of $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$, and hence in extenso also of $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}$.

Remark 1.68 The following are immediate consequences (in $\mathtt{IL}_0$ already) of the newly introduced axioms at Definition 1.64.3 above:

$$\vdash\ z \geq_{\iota} \mathtt{0} \qquad \vdash\ \mathtt{0} \geq_{\iota} \mathtt{Suc}z \to \mathtt{at}(\mathtt{ff}) \qquad \vdash\ \mathtt{Suc}x \geq_{\iota} \mathtt{Suc}y \to x \geq_{\iota} y$$

Remark 1.69 Max is in fact only a common upper bound which commutes with $\succeq_{\iota\iota\iota}$. This suffices for the proof of the soundness (of light monotone Dialectica extraction) Theorem 2.12. More axioms can and must be added to ensure that Max truly denotes the integer maximum of its arguments.

Definition 1.70 (Maximum functionals at higher arithmetic types) Maximum functionals at higher arithmetic types $\tau \equiv \underline{\sigma}\iota$ are defined by

$$\mathtt{Max}_{\tau} \quad :\equiv \quad \lambda x^{\tau}, y^{\tau}, \underline{z}^{\underline{\sigma}}\ .\ \mathtt{Max}\,(x\,\underline{z})\,(y\,\underline{z}) \tag{1.40}$$

We will also use $\mathtt{Max}_{\iota} :\equiv \mathtt{Max}$ as an extension of the notation (1.40) above.

[20] Hence B is obtained from $B^{\mathtt{nc}}$ by translating each ncm quantifier to its corresponding regular quantifier, just like the formulas B from the set $\widetilde{\Pi}$ of Remark 1.56.

Definition 1.71 (n-maxima) Multiple maxima for every type τ and arbitrary $n \geq 2$ are defined inductively on n as follows: $\mathtt{Max}^2_\tau :\equiv \mathtt{Max}_\tau$ and

$$\mathtt{Max}^{n+1}_\tau \quad :\equiv \quad \lambda x_{n+1}, x_n, \ldots, x_1.\, \mathtt{Max}_\tau\, x_{n+1}\, (\mathtt{Max}^n_\tau\, x_n \ldots x_1) \quad \text{for } n \geq 2\ .$$

Lemma 1.72 Let $\rho \equiv \sigma_1 \ldots \sigma_n \tau$, with $n \geq 0$. The following hold in $\mathtt{WeZ}_\mathtt{m}$:

$$\vdash \quad x^\star \ \succeq_\rho\ x \ \leftrightarrow\ \forall y^\star_1, y_1, \ldots, y^\star_n, y_n\, [\, y^\star_1 \ \succeq_{\sigma_1}\ y_1 \ \rightarrow\ \ldots \\ \ldots \ \rightarrow\ y^\star_n \ \succeq_{\sigma_n}\ y_n \ \rightarrow\ (x^\star\, y^\star_1 \ldots y^\star_n)\ \succeq_\tau\ (x\, y_1 \ldots y_n)\,] \tag{1.41}$$

$$\vdash \quad y^\star \geq_\rho x^\star \ \wedge\ x^\star\ \succeq_\rho\ x \ \wedge\ x \geq_\rho y\ \rightarrow\ y^\star\ \succeq_\rho\ y \tag{1.42}$$

$$\vdash \quad x^\star\ \succeq_\rho\ x \ \wedge\ y^\star\ \succeq_\rho\ y\ \rightarrow\ (\mathtt{Max}_\rho\, x^\star\, y^\star)\ \succeq_\rho\ (\mathtt{Max}_\rho\, x\, y) \tag{1.43}$$

Proof: (1.41) is proved by induction on $n \geq 1$ using the definition of $\succeq$ at both base and step cases; (1.42) is proved by decomposing ρ as $\underline{\sigma}\iota$ or $\underline{\sigma}o$ and using (1.41) together with (1.37); (1.43) is proved by decomposing ρ as $\underline{\sigma}\iota$ and using (1.41) together with (1.40). $\square$

Remark 1.73 Notice that (1.43) simply implies that $\mathtt{Max}_\rho\ \succeq_{\rho\rho\rho}\ \mathtt{Max}_\rho$.

Definition 1.74 To each Gödel recursor $\mathtt{R}_\tau$ one associates the term $\mathtt{R}^\star_\tau$ of the same type, namely of type $\tau(\iota\tau\tau)\iota\tau$, as follows:

$$\mathtt{R}^\star_\tau \quad :\equiv \quad \lambda x^\tau, y^{\iota\tau\tau}, z^\iota.\, \mathtt{R}_\tau x (\lambda z, x.\, \mathtt{Max}_\tau\, x\, (yzx)) z$$

Lemma 1.75 For every type τ the following holds: $\mathtt{WeZ}_\mathtt{m} \vdash \mathtt{R}^\star_\tau\ \succeq\ \mathtt{R}_\tau$.

Proof: Let $t[y] :\equiv \lambda z, x.\, \mathtt{Max}_\tau\, x\, (yzx)$. We can prove by induction on z that

$$\mathtt{WeZ}_\mathtt{m}\ \vdash\ \forall z\,(\, x^\star\ \succeq_\tau\ x \rightarrow y^\star\ \succeq_{\iota\tau\tau}\ y \rightarrow z^\star \geq_\iota z \rightarrow \mathtt{R}_\tau x^\star t[y^\star] z^\star\ \succeq_\tau\ \mathtt{R}_\tau x y z\,)$$

For this we use the following: $\mathtt{R}_\tau x^\star t[y^\star](\mathtt{Suc} z) \hookrightarrow t[y^\star] z (\mathtt{R}_\tau x^\star t[y^\star] z)$, hence

$$\mathtt{WeZ}_\mathtt{m}\ \vdash\ \forall z\,.\ \mathtt{R}_\tau x^\star t[y^\star](\mathtt{Suc} z)\ =_\tau\ \mathtt{Max}_\tau\, (\mathtt{R}_\tau x^\star t[y^\star] z)\, [y^\star z (\mathtt{R}_\tau x^\star t[y^\star] z)] \tag{1.44}$$

For the base case $z = 0$ one uses then the following easy consequence of (1.44):

$$\mathtt{WeZ}_\mathtt{m}\ \vdash\ \forall z\,.\ \mathtt{R}_\tau x^\star t[y^\star](\mathtt{Suc} z)\ \geq_\tau\ \mathtt{R}_\tau x^\star t[y^\star] z$$

which immediately implies

$$\mathtt{WeZ}_\mathtt{m}\ \vdash\ \forall z\,.\ \mathtt{R}_\tau x^\star t[y^\star] z\ \geq_\tau\ \mathtt{R}_\tau x^\star t[y^\star] 0\ =_\tau\ x^\star \tag{1.45}$$

Now the induction base follows from (1.45) and $x^\star \succeq_\tau x$ using (1.42). For the induction step, let $z^\star \geq \mathtt{Suc}z$, hence one can write $z^\star \equiv \mathtt{Suc}\widetilde{z}$, where $\widetilde{z} \geq z$. From the induction hypothesis we then have, assuming $x^\star \succeq x$ and $y^\star \succeq y$, that $\mathtt{R}_\tau x^\star t[y^\star]\widetilde{z} \succeq \mathtt{R}_\tau xyz$, from which the following follows by Lemma 1.72:

$$\mathtt{R}_\tau x^\star t[y^\star](\mathtt{Suc}\widetilde{z}) \geq_\tau y^\star\widetilde{z}(\mathtt{R}_\tau x^\star t[y^\star]\widetilde{z}) \succeq_\tau yz(\mathtt{R}_\tau xyz) =_\tau \mathtt{R}_\tau xy(\mathtt{Suc}z)$$

Above we also used (1.44) for $\geq_\tau$ and the definition for $=_\tau$. By (1.42) this implies $\mathtt{R}_\tau x^\star t[y^\star]z^\star \succeq_\tau \mathtt{R}_\tau xy(\mathtt{Suc}z)$ and the whole proof is now finished. □

Remark 1.76 From (1.39) and (1.41) it follows that all boolean terms t of type $\tau \equiv \underline{\sigma}o$ are simply majorized by terms $\mathtt{True}^\tau :\equiv \lambda x_1^{\sigma_1}, \ldots, x_k^{\sigma_k}.\,\mathtt{tt}$.

Lemma 1.77 It can be immediately established from definitions, in $\mathtt{WeZ}_{\mathtt{m}}$, using (1.41), that $\Sigma \succeq \Sigma$ and $\Pi \succeq \Pi$ for all combinators Σ and Π.

Proposition 1.78 (Construction of majorants for all lambda-terms) Let $t[x_1, \ldots, x_k]^\tau$ be a $\mathtt{WeZ}_{\mathtt{m}}$ arithmetic term such that $x_1^{\sigma_1}, \ldots, x_k^{\sigma_k}$ are all its free arithmetic variables. There exists a generic syntactic procedure $\cdot^\star : t \mapsto t^\star$ which associates to it an arithmetic term $t^\star[x_1, \ldots, x_k]^\tau$ such that $x_1, \ldots, x_k$ are all the free variables of $t^\star$ and the following holds in $\mathtt{WeZ}_{\mathtt{m}}$:

$$\vdash \; x_1^\star \succeq_{\sigma_1} x_1 \to \ldots x_k^\star \succeq_{\sigma_k} x_k \to t^\star[x_1^\star, \ldots, x_k^\star] \succeq_\tau t[x_1, \ldots, x_k] \quad (1.46)$$

As a consequence, $\lambda x_1, \ldots, x_k.\, t^\star[x_1, \ldots, x_k] \succeq \lambda x_1, \ldots, x_k.\, t[x_1, \ldots, x_k]$ and also simply $t^\star \succeq t$ whenever t is a closed term.

Proof: The construction of $t^\star$ from t is by recursion on the arithmetic structure of t, starting from (free) arithmetic variables and constants of $\mathtt{WeZ}_{\mathtt{m}}$ and maximal boolean subterms of t. The latter are simply majorized by closed terms $\mathtt{True}$, see Remark 1.76. On the other hand, all arithmetic constants of $\mathtt{WeZ}_{\mathtt{m}}$ are majorized by closed arithmetic terms. For example, the selector $\mathtt{If}_\tau$ for arithmetic τ is majorized by $\mathtt{If}_\tau^\star :\equiv \lambda p^o, y^\tau, z^\tau.\, \mathtt{Max}_\tau\, y\, z$. Also Lemma 1.75 gives that $\mathtt{R}^\star \succeq \mathtt{R}$. The proof of (1.46) is now by induction on the associated combinatorial structure $\widetilde{t}$ of t (see Proposition 1.53) using Lemmas 1.72 and 1.77. In fact one proves that $\widetilde{t^\star}[x_1^\star, \ldots, x_k^\star] \succeq \widetilde{t}[x_1, \ldots, x_k]$, where, relative to the $\cdot \mapsto \widetilde{\cdot}$ translation, one moreover considers that $\mathtt{If}^\star$, $\mathtt{R}^\star$ and all other majorants for constants are (like) constants (e.g., they are "seen" as constants). □

Remark 1.79 (Possibly simpler majorants by normalization) From definitions it follows immediately that if $t \hookrightarrow^\star s$ then $t = s$, hence $s^\star \succeq t$ follows by compatibility CMP. If moreover $s^\star \hookrightarrow^\star s_m$ then also by compatibility we get $s_m \succeq t$. Thus if the normalization by evaluation (NbE) simplifies the term $s^\star$ into s_m, this is a way to produce a simpler majorant for an initial term t. Such a situation will appear concretely in the light monotone Dialectica extraction of terms in (NbE-)normal form, see Remark 2.15.

1.6 The classical (monotonic) Arithmetics $\mathtt{WeZ}^{\exists,\mathtt{cl}}$, $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}/\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ and $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{cl}}/\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$

The systems $\mathtt{WeZ}^{\exists,\mathtt{cl}}$, $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$ and $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{cl}}$ of weakly extensional (monotonic) Classical Arithmetic (with strong $\exists$/ ncm quantifiers) are obtained by adding to $\mathtt{WeZ}^{\exists}$, $\mathtt{WeZ}^{\mathtt{nc}}$ and respectively $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$, the full Stability:

$$\mathtt{AxSTAB}: \quad \neg\neg A \to A \qquad \text{(Full Stability)}$$

where the formula $A \in \mathcal{F}^{\exists}$, $A \in \mathcal{F}^{\mathtt{nc}}$ and respectively $A \in \mathcal{F}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ is unrestricted. It would be in fact sufficient that the axiom AxSTAB replaces AxEFQ/AxFLS, since the latter are deducible from AxSTAB in Minimal Logic. However we prefer to keep AxEFQ/AxFLS as axioms of our systems since they have simple Light Dialectica realizers[21].

Recall that AxSTAB is fully provable in $\mathtt{WeZ}_{\mathtt{m}}/\mathtt{WeZ}_{\mathtt{m}}^{\exists}$, hence it would not make sense to add it to these (non-ncm) systems. Also notice that no system $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc},\mathtt{cl}}$ is defined. This is because such a system would not make sense, since AxSTAB is already fully provable in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$, see Proposition 1.65. We also chose to define no system $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}$. The reason here is rather different: we just do not find any suitable double negation translation for $\mathtt{Ax}\exists^{-}$ (and also $\mathtt{Ax}\overline{\exists}^{-}$) in the presence of the ncm quantifiers, see Section 2.3 for more on this.

We nevertheless choose to consider the system $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$, since on one hand AxSTAB is not fully provable in $\mathtt{WeZ}^{\mathtt{nc}}$, see Remark 1.42 and Lemma 1.40. On the other hand, despite some technical difficulties, a certain negative translation can nevertheless be designed from $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$ to $\mathtt{WeZ}^{\mathtt{nc}}$, due to the lack of strong existentials. Hence for $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$ we are also constrained to further strengthen the restriction set on the rule $\to^+$ of Implication introduction, see Definitions 1.18 and 1.15. The reinforced $\to^+$ condition restricts not only the introduction

[21] Input proofs to the (light) Dialectica-extraction algorithm may thus become shorter.

formula A, but also the conclusion formula B, in fact a certain combination of the two. Hence the new $\rightarrow^{+}$ restriction is $\mathtt{ncm}-\mathtt{FC}(\neg(A \rightarrow B))$, or equivalently $\mathtt{ncm}-\mathtt{FC}(A \wedge \neg B)$. Notice that this cannot be written in the meta-language as "$\mathtt{ncm}-\mathtt{FC}(A)$ and $\mathtt{ncm}-\mathtt{FC}(\neg B)$". This addition is triggered by the concern about the soundness of our adaptation of Kuroda's double negation translation (the $\mathtt{KN}$-translation) to the new $\mathtt{ncm}$-classical (non-monotonic) system, see Definition 2.24 and Theorem 2.27 in Section 2.3. Hence the $\mathtt{KN}$-translation of $\rightarrow^{+}$ can be simulated only by means of an extra contraction over the formula $\neg(A \rightarrow B)$, for which we have to consider its $\mathtt{ncm}-\mathtt{FC}$ restriction, see the proof of Theorem 2.27 for full details on this issue. In consequence also the variable condition $\mathtt{VC}_3(z, \mathcal{P})$ which is set on the $\mathtt{ncm}$-ForAll introduction rule $\overline{\forall}^{+}$ must be strengthened in the sense that z is required to not be free also in the conclusions of the computationally $\mathtt{LD}$-relevant $\rightarrow^{+}$ from $\mathcal{P}$ and not only in the corresponding contraction formulas of such $\rightarrow^{+}$.

Nonetheless, for the *monotonic* classical $\mathtt{ncm}$-system $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{cl}}$, no such strengthening is needed at all, since on one hand no restriction is set on $\rightarrow^{+}$ already in the corresponding constructive system $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ anyway, recall Definition 1.64. On the other hand, the design of both Kuroda and Gödel-Gentzen double-negation translations is no longer problematic in the presence of $\exists/\overline{\exists}$.

Remark 1.80 (Portability of axioms sets Π and Δ to clas. systems) Recall from Convention 1.57 that $\mathtt{WeZ}^{\mathtt{nc}}/\,\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ include an arbitrary but fixed set of axioms $\Pi \equiv \{\forall z B^{\mathtt{nc}}(z)\}$, which induces the inclusion of the corresponding set $\widetilde{\Pi} \equiv \{\forall z B(z)\}$ into $\mathtt{WeZ}/\,\mathtt{WeZ}_{\mathtt{m}}^{\exists}$. This implies that Π is further inherited into $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}/\,\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{cl}}$ and also that its corresponding $\widetilde{\Pi}$ gets included into the verifying systems $\mathtt{WeZ}/\,\mathtt{WeZ}_{\mathtt{m}}^{\exists}$. Also recall from Definition 1.67 that $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ includes the arbitrary but fixed set of axioms

$$\Delta \;\equiv\; \{\, \forall x^{\rho}\, \exists y \leq_{\sigma} rx\, \forall z^{\tau}\; B^{\mathtt{nc}}(x,y,z) \,\}$$

and that correspondingly, the verifying system $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ contains the associated axiom set

$$\widetilde{\Delta} \;\equiv\; \{\, \exists Y \leq_{\rho\sigma} r\, \forall x^{\rho}\, \forall z^{\tau}\; B(x, Yx, z) \,\} \qquad .$$

In consequence, the axiom set Δ is further inherited into $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{cl}}$, and correspondingly $\widetilde{\Delta}$ gets included into $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$.

We now consider other possible extensions of the fully classical systems. Notice that the axioms $\mathtt{AxMK}$ and $\mathtt{AxSTAB}^{\mathtt{nc}}$ would become redundant in these

systems. For $\mathtt{AxMK}$ it can be immediately be established, *without contraction*, that $\exists^{\mathtt{cl}} z B(z) \leftrightarrow \neg\neg\exists z B(z)$ for every formula B (and hence in particular for $B \equiv A^{\mathtt{nc}}$). We are thus brought to consider the following (now) equivalent formulation of $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ (recall it from Section 1.4, also see it below):

$$\mathtt{AxAC}_{\mathtt{nc}}: \qquad \forall x\, \exists y\, B^{\mathtt{nc}}(x,y) \;\to\; \exists Y\, \forall x\, B^{\mathtt{nc}}(x, Y(x)) \qquad ,$$

which is (more directly seen to be) the pure-$\mathtt{ncm}$ version of $\mathtt{AxAC}$ (which still uses the strong $\exists$). Nonetheless, because of the $\mathtt{ncm}-\mathtt{FC}$ restriction, the $\mathtt{ncm}$-classical arithmetic $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$ does not necessarily include all theorems of $\mathtt{WeZ}$ extended to the $\mathtt{ncm}$ language. Some of these theorems no longer hold if the formulas involved contain at least one $\mathtt{ncm}$ quantifier. It is the case of Independence of premises, $[\, A \to \exists y\, B(y)\,] \to \exists y\, [\, A \to B(y)\,]$, whose proof makes a necessary use of contraction over $\forall y.[A \to B(y)] \to \bot$, see Section 1.4. We are thus again brought to consider the addition to $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$ of

$$\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}: \qquad [\, A^{-\exists}_{+\forall} \to \exists^{\mathtt{cl}} y\, B^{+\exists}_{-\forall}(y)\,] \to \exists^{\mathtt{cl}} y\, [\, A^{-\exists}_{+\forall} \to B^{+\exists}_{-\forall}(y)\,] \qquad ,$$

where A and B can freely contain $\overline{\forall}$, but A is restricted to contain $\forall$ in positive positions only and B is restricted to contain $\forall$ in negative positions only (A quasi-universal and B quasi-existential, notions first introduced in Section 1.4).

Definition 1.81 We denote by $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ the extension of system $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{cl}}_{\mathtt{m}}$ with $\mathtt{AxAC}_{\mathtt{nc}}$, but also with the already presented pure-$\mathtt{ncm}$ $\exists^{\mathtt{cl}}$-variant (which is though logically equivalent to $\mathtt{AxAC}_{\mathtt{nc}}$):

$$\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}: \qquad \forall x\, \exists^{\mathtt{cl}} y\; B^{\mathtt{nc}}(x,y) \;\to\; \exists^{\mathtt{cl}} Y\, \forall x\; B^{\mathtt{nc}}(x, Y(x)) \qquad .$$

Whereas for the system with strong $\exists$ the addition of $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ is redundant, for $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$ this appears to be the only possibility. We thus obtain the system $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ as the extension of $\mathtt{WeZ}^{\mathtt{nc},\mathtt{cl}}$ with $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$, but also with $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$.

Discussion

We have presented all the arithmetical systems which are to be used in this thesis, except for the "feasible" systems of Chapter 3. The latter will nevertheless be built on the top of the systems developped so far. This progressive and comparative exposition is meant to prepare the reader to the truly effective part of the thesis, namely the presentation of the extraction techniques in the following chapter. In fact the reading of the present chapter is inseparable from the one of Chapter 2, particularly concerning the specific

ncm restrictions. Also the discharging of most of these restrictions in the monotonic systems can only make sense while reading the corresponding proofs of the soundness-and-extraction theorems.

Chapter 2

The light (monotone) functional interpretation

By *LD-interpretation* (a short for "Light Dialectica Interpretation") we call below our adaptation of Gödel's functional "Dialectica" interpretation [45] to the extraction of (more) efficient programs from (classical) proofs. The LD-interpretation is a recursive syntactic translation from proofs in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ (or in $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$, modulo the Kuroda double-negation translation) to proofs in $\mathtt{WeZ}^{\exists}$ or even WeZ, such that the positive occurrences of $\exists$ and the negative occurrences of $\forall$ in the proof's conclusion formula get effectively realized by terms in Gödel's $\mathbf{T}$. These *realizing terms* are also called the *programs extracted* by the LD-interpretation and (if only the extracted programs are wanted) the translation process is also referred to as *program-extraction*. The translated proof is also called the *verifying proof* since it verifies the fact that the extracted programs truly realize the existential side of the LD-interpretation (LD-translation) of the conclusion formula of the proof at input.

Gödel's Dialectica interpretation (abbreviated *D-interpretation*) is relatively (far) more complicated when it has to face (computationally relevant[1]) *contraction*, which in Natural Deduction amounts to discharging more than one copy of an assumption formula in an Implication Introduction $\rightarrow^{+}$. Kohlenbach introduces in [68] an elegant method for simplifying the treatment of contraction when the goal is to extract Howard majorizing functionals for

[1] In the sense of Definition 1.18. The pure Gödel D-interpretation has to deal with a larger class of so-called "computationally D-relevant contractions". The key optimizing feature of the LD-interpretation is that there may be some D-relevant contractions which turn out to not be LD-relevant (i.e., to be LD-irrelevant, and thus D-redundant, see Remark 1.19).

the Dialectica realizers. He named "monotone functional interpretation" this variant of the D-interpretation which we here abbreviate by *MD-interpretation*.

The MD-interpretation has been used with great success over the last years for producing proofs to important new theorems in numerical functional analysis ([81] and [63] contain comprehensive surveys of such to-day applications of MD-interpretation to concrete mathematical proofs from the literature - for direct references see, e.g., [68, 62, 64, 66, 67, 75, 76, 78, 79, 80, 82, 41, 77]). Such theorems are completely new in the sense that they were not established previously via the usual mathematical means from inside the theory (despite attempts). However, whenever the extraction of *exact* realizers is concerned, the MD-interpretation does not necessarily bring efficient answers. A different kind of optimization of Gödel's D-interpretation appears to be necessary. We here propose the LD-interpretation as a refinement of Gödel's technique which allows for the extraction of more efficient exact realizers. Moreover, the same refinement equally applies to the extraction of more efficient majorants and bounds via what might be called the *light monotone* (or *light bounded*, following [33] rather than [68]) functional interpretation (abbreviated LMD-interpretation).

The D-interpretation was first introduced in [45] for a Hilbert-style formulation of Arithmetic – see also [123, 90, 25, 4, 63] for other (more modern) formulations within Hilbert-style systems. Natural Deduction formulations of the Diller-Nahm [25] variant of D-interpretation were provided by Diller's students Rath [107] and Stein [121]. Only in the year 2001 Jørgensen [61] provided a first Natural Deduction formulation of the original Gödel's functional interpretation. In the Diller-Nahm setting all choices between the potential realizers of a contraction are postponed to the very end by collecting all candidates and making a single final global choice. In contrast, Jørgensen's formulation respects Gödel's original treatment of contraction by immediate (local) choices. Jørgensen devises a so-called "Contraction Lemma" in order to handle (in the given Natural Deduction context) the discharging of more than one copy of an assumption in an Implication Introduction $\rightarrow^+$. If $n+1$ undischarged occurrences of an assumption are to be cancelled in an $\rightarrow^+$ then Jørgensen uses his Contraction Lemma n times, shifting partial results n times back and forth over the "proof gate" $\vdash$. We find this not only inefficient from the applied program-extraction perspective but also inelegant since the soundness proof for the D-interpretation complicates unnecessarily w.r.t. contraction. We will instead use the n-selector $\mathtt{If}^n_\tau$ for equalizing in one

single (composed) step all the LD-interpretations of the $n+1$ undischarged occurrences (see the proof of Theorem 2.10 below). While it is technically impossible to have a direct n-selector available for all $n \in \mathbb{N}$, in actual optimizing implementations $\mathtt{If}^n_\tau$ could be given a (more) direct definition for $n \leq N$ for a certain convenient upper margin N instead of being simulated in terms of n times $\mathtt{If}^1_\tau$ (only a certain limited number of n-selectors is needed for most practical applications). The practical gain w.r.t. Jørgensen's solution is that the handling of contraction is directly moved from the proof level to the term level: back-and-forth shifting over $\vdash$ is no longer required when building the verifying proof.

We also modify Jørgensen's variant of D-interpretation by allowing free variables in the extracted terms. This corresponds to the formulation of Gödel's **T** with λ-abstraction as primitive and is more natural in a Natural Deduction setting. In addition we include the treatment of our adaptation of Berger's uniform existential and universal quantifiers from [9] to the Dialectica-extraction context. In fact, it is exactly these ncm quantifiers that bring the so-called "light" optimization of Dialectica, since they can be used to isolate some computationally redundant contractions from the extracted exact realizers. In the monotonic context, the ncm quantifiers are exactly like Berger's, without generating any further restrictions on Implication Introduction or ncm-ForAll Introduction, see Definition 1.64. Even though the treatment of Contraction is simpler by the Monotone Dialectica, still the "light" optimization can bring a serious further improvement, see the comment in the last paragraph of Section 2.2. Therefore the Light Monotone Dialectica interpretation is a very successful combination of the two optimizations of Dialectica, which join together in a very smooth way. In consequence, the LMD-interpretation also extends very easily to the extraction from fully classical proofs, in contrast to the LD-interpretation, which has very big difficulties in this sense, due to the heavier ncm restrictions, see Section 2.3 for details.

2.1 The light Gödel Dialectica interpretation

The light functional (or "Light Dialectica") interpretation/translation (on short "LD-interpretation" or "LD-translation") subsumes the following unique assignment of formulas. To each instance[2] of the formula $A(\underline{a})$ (with $\underline{a}$ all

[2] Here by *instance* of a formula A we understand on one hand an individual occurrence of A as label for a node in a proof-tree $\mathcal{P}$, but on the other hand also a formula A^α

the free variables of A) one associates a unique (α-isomorphism comprised) formula $A^{\mathtt{D}}(\underline{a}) \equiv \exists \underline{x}\, \forall \underline{y}\, A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a})$ with $A_{\mathtt{D}}$ not necessarily quantifier-free[3] and $\underline{x}, \underline{y}$ unique tuples of fresh variables of finite type (such that $\{\underline{x}, \underline{y}, \underline{a}\}$ are all free variables of $A_{\mathtt{D}}$). The types of $\underline{x}, \underline{y}$ depend only on the types of the regularly bound variables of A and on the logical structure of A.

Notational Convention. Whenever we will use below the formula notation "A", its free variables will be denoted "$\underline{a}$" and its LD-translation will be denoted "$A^{\mathtt{D}}(\underline{a}) \equiv \exists \underline{x}\, \forall \underline{y}\, A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a})$". Similarly, when the formula notation "B" will be used, the free variables of B will be denoted "$\underline{b}$" and its LD-interpretation will be denoted "$B^{\mathtt{D}}(\underline{b}) :\equiv \exists \underline{u}\, \forall \underline{v}\, B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{b})$".

Notational Convention. If non-ambiguous, we sometimes omit to display some of the free variables of the LD-translated formulas.

The LD-interpretation/translation of formulas is then given by the following:

Definition 2.1 (The light Dialectica interpretation of formulas) Recall the definition 1.4 of quantifier-free formulas (which may contain $\vee$) and the mapping of qfr formulas A_0 to boolean terms $\mathtt{t}_{A_0}$ from Proposition 1.35.

$$
\begin{array}{rcl}
A^{\mathtt{D}} & :\equiv & A_{\mathtt{D}} :\equiv \mathtt{at}(\mathtt{t}_A) \text{ for maximal quantifier-free formulas } A \\
(A \wedge B)^{\mathtt{D}} & :\equiv & \exists \underline{x}, \underline{u}\, \forall \underline{y}, \underline{v}\, [\, (A \wedge B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) \wedge B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{b})\,] \\
(\exists z A(z, \underline{a}))^{\mathtt{D}} & :\equiv & \exists z^{\dagger}, \underline{x}\, \forall \underline{y}\, [\, (\exists z A(z, \underline{a}))_{\mathtt{D}}(z^{\dagger}, \underline{x}; \underline{y}; \underline{a}) :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}; z^{\dagger}, \underline{a})\,] \\
(\forall z A(z, \underline{a}))^{\mathtt{D}} & :\equiv & \exists \underline{X}\, \forall z^{\dagger}, \underline{y}\, [\, (\forall z A(z, \underline{a}))_{\mathtt{D}}(\underline{X}; z^{\dagger}, \underline{y}; \underline{a}) :\equiv A_{\mathtt{D}}(\underline{X}(z^{\dagger}); \underline{y}; z^{\dagger}, \underline{a})\,] \\
(\overline{\exists} z A(z, \underline{a}))^{\mathtt{D}} & :\equiv & \exists \underline{x}\, \forall \underline{y}\, [\, (\overline{\exists} z A(z, \underline{a}))_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) :\equiv \exists z\, A_{\mathtt{D}}(\underline{x}; \underline{y}; z, \underline{a})\,] \\
(\overline{\forall} z A(z, \underline{a}))^{\mathtt{D}} & :\equiv & \exists \underline{x}\, \forall \underline{y}\, [\, (\overline{\forall} z A(z, \underline{a}))_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) :\equiv \forall z\, A_{\mathtt{D}}(\underline{x}; \underline{y}; z, \underline{a})\,] \\
(A \to B)^{\mathtt{D}} & :\equiv & \exists \underline{Y}, \underline{U}\, \forall \underline{x}, \underline{v}\, [\, (A \to B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v})\,]
\end{array}
$$

Here $\cdot \mapsto \cdot^{\dagger}$ is a mapping which assigns to every given variable z a completely new variable $z^{\dagger}$ which has the same type of z. Different variables $z^{\dagger}$ are returned for different applications on the same argument variable z when processing a given formula A. This ensures that two nested quantifications of the same

which is α-isomorphic to A. Here α-isomorphism of formulas is meant in the sense that the corresponding terms of A and A^{α} are α-equivalent (the usual notion from typed λ-calculus).

[3] Here is one difference w.r.t. the pure Gödel Dialectica, for which $A_{\mathtt{D}}$ is always quantifier-free. On the other hand, this situation is quite similar to the approach from Ferreira and Oliva's bounded functional interpretation, see [33, 34].

variable in A are correctly distinguished in $A_{\mathtt{D}}$. The variables $\underline{X}, \underline{Y}, \underline{U}$ produced in the treatment of $\rightarrow$ and $\forall$ are also completely new but in contrast to the variables yielded by $\cdot^{\dagger}$, their type is strictly more complex than the type of the original variable. The free variables of $A^{\mathtt{D}}$ are exactly the free variables of A. If A is quantifier-free then $\mathtt{IL}/\mathtt{IL}^{\exists} \vdash A \leftrightarrow A^{\mathtt{D}} \equiv A_{\mathtt{D}}$.

Remark 2.2 (Non-quantifier-free radical/root associated formula $A_{\mathtt{D}}$) For the light Dialectica interpretation, the *radical* (or "root") formula $A_{\mathtt{D}}$ (which is LD-interpretation associated to A) is not necessarily quantifier-free, like it is for the pure Gödel's functional interpretation. It actually contains the translation of all ncm quantifiers to the corresponding regular quantifiers.

Convention 2.3 The LD-associated formula $A^{\mathtt{D}}$ is unique (alpha-isomorphism comprised) for all instances of the formula A. See also Footnote 2.

Remark 2.4 The use of $\cdot^{\dagger}$ is mandatory: just consider the interpretation of

$$\forall z.\, (z = \mathtt{0} \rightarrow \forall z.\, z = \mathtt{0}) \tag{2.1}$$

The sentence (2.1) is obviously false in our systems. An uncarefully formalized (L)D-interpretation would translate it to the universally true sentence

$$\forall z. \forall z.\, z = \mathtt{0} \rightarrow z = \mathtt{0} \qquad .$$

Remark 2.5 By abuse of notation from now on we use non-underlined letters also for tuples of objects (variables, terms, ...) and identify an individual object with the tuple containing only that object.

Definition 2.6 (Dialectica terms) For every ncm-quantifier free formula $A(a)$ we denote by $\mathtt{t}^{\mathtt{D}}_A[x; y; a]$ the boolean term associated to (the quantifier-free formula) $A_{\mathtt{D}}(x; y; a)$ by the mapping from Proposition 1.35. We call it "the Dialectica term associated to" $A(a)$. The following holds:

$$\mathtt{IL} \vdash A_{\mathtt{D}}(x; y; a) \leftrightarrow \mathtt{at}(\mathtt{t}^{\mathtt{D}}_A[x; y; a]) \tag{2.2}$$

Remark 2.7 Terms $\mathtt{t}^{\mathtt{D}}_A$ can also be defined directly, in parallel with the definition of $A^{\mathtt{D}}$ as follows (association by induction on the structure of the formula

A, starting from quantifier-free components):

$$\begin{array}{rcl}
\mathtt{t}^{\mathtt{D}}_{\mathtt{at}(\mathtt{t}_{A_0}[a])}[;;a] & :\equiv & \mathtt{t}_{A_0}[a] \\
\mathtt{t}^{\mathtt{D}}_{A\wedge B}[x,u;y,v;a,b] & :\equiv & \mathtt{And}\,\mathtt{t}^{\mathtt{D}}_{A}[x;y;a]\,\mathtt{t}^{\mathtt{D}}_{B}[u;v;b] \\
\mathtt{t}^{\mathtt{D}}_{\exists z A(z,a)}[z^\dagger,x;y;a] & :\equiv & \mathtt{t}^{\mathtt{D}}_{A(z,a)}[x;y;z^\dagger,a] \\
\mathtt{t}^{\mathtt{D}}_{\forall z A(z,a)}[X;z^\dagger,y;a] & :\equiv & \mathtt{t}^{\mathtt{D}}_{A(z,a)}[X(z^\dagger);y;z^\dagger,a] \\
\mathtt{t}^{\mathtt{D}}_{A\to B}[Y,U;x,v;a,b] & :\equiv & \mathtt{Imp}\,\mathtt{t}^{\mathtt{D}}_{A}[x;Y(x,v);a]\,\mathtt{t}^{\mathtt{D}}_{B}[U(x);v;b]
\end{array}$$

The proof of (2.2) would then be immediate by structural induction. Anyway, even with such a definition, $\mathtt{t}^{\mathtt{D}}_{A}$ would be literally, syntactically equal with the term $\mathtt{t}_{A_{\mathtt{D}}}$ associated to the quantifier-free formula $A_{\mathtt{D}}$ by Proposition 1.35.

Below we present the LD-interpretation of some (generic) formulas of interest:

$$\left.\begin{array}{rcl}
(\neg A)^{\mathtt{D}}(a) & \equiv & \exists Y\,\forall x\,\neg A_{\mathtt{D}}(x;Y(x);a) \\
(\neg\neg A)^{\mathtt{D}}(a) & \equiv & \exists X\,\forall Y\,\neg\neg A_{\mathtt{D}}(X(Y);Y(X(Y));a) \\
(\exists^{\mathtt{cl}} z\,A(z))^{\mathtt{D}}(a) & \equiv & \exists Z,X\,\forall Y\,\neg\neg A_{\mathtt{D}}(X(Y);Y(Z(Y),X(Y));Z(Y),a) \\
(\neg\neg\exists z A(z))^{\mathtt{D}}(a) & \equiv & \exists Z,X\,\forall Y\,\neg\neg A_{\mathtt{D}}(X(Y);Y(Z(Y),X(Y));Z(Y),a)
\end{array}\right\} \quad (2.3)$$

Remark 2.8 Formula $\exists^{\mathtt{cl}} z\,A(z)$ has the same LD-interpretation as $\neg\neg\exists z A(z)$.

Definition 2.9 (ncm-stable pure-ncm formulas) A pure-ncm formula $A^{\mathtt{nc}}$ is defined to be "ncm-stable" if it does not contain the strong ncm-existential quantifier $\overline{\exists}$ in a strictly positive position (in the sense of Definition 1.13). Hence for the LD-interpretation A of such formula $A^{\mathtt{nc}}$ one can prove stability $\neg\neg A \to A$ already in $\mathtt{WeZ}^{\exists}$, see Corollary 1.41.

Theorem 2.10 (Extraction and its soundness for Light Dialectica [51])
There exists an algorithm which, given at input a proof $\mathcal{P} : \{C^i\}_{i=1}^n \vdash A$ in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ or $\mathtt{WeZ}^{\mathtt{nc}+}$, will produce at output the following:

1. the tuples of terms $\{T_i\}_{i=1}^n$ and T,
2. the tuples of variables $\{x_i\}_{i=1}^n$ and y, all together with
3. the verifying proof $\mathcal{P}_{\mathtt{D}} : \{C^i_{\mathtt{D}}(x_i;T_i(\underline{x},y))\}_{i=1}^n \vdash A_{\mathtt{D}}(T(\underline{x});y)$ in $\mathtt{WeZ}^{\exists,\mathtt{cl}}$, or respectively WeZ – where $\underline{x} :\equiv x_1,\ldots,x_n$ above.

Moreover,

- the variables $\underline{x}$ and y do not occur in $\mathcal{P}$ (they are all completely new)
- the free variables of T and $\{T_i\}_{i=1}^n$ are among the free variables of A and $\{C^i\}_{i=1}^n$ – we call this "the *free variable condition* (FVC) for programs extracted by the LD-interpretation".

hence $\underline{x}$ and y also do not occur free in the *extracted* terms $\{T_i\}_{i=1}^n$ and T. If the input proof in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ does not employ any $\mathtt{AxSTAB}^{\mathtt{nc}}$ or AxMK in which the pure-ncm main formula is not ncm-stable then the verifying proof is in $\mathtt{WeZ}^{\exists}$ rather than $\mathtt{WeZ}^{\exists,\mathtt{cl}}$ (since Stability can be proved in $\mathtt{WeZ}^{\exists}$ for cases of interest).

Proof: The algorithm proceeds by recursion on the structure of the input proof $\mathcal{P}$. Realizing terms must be presented for all the axioms and then realizing terms for the conclusion of a rule must be deduced from terms which realize the premise of that rule. Since $\underline{x}, y$ are produced by the LD-interpretation of formulas (see Definition 2.1) it is immediate that they do not occur in $\mathcal{P}$. We present below all cases. Notice that the (sub)case of Implication Introduction $\rightarrow^+$ in which at least two copies of the implicative assumption get cancelled is far more difficult than all the other axioms and rules[4] because it involves contraction, which may be computationally LD-relevant. In such a case, the whole LD-translation of the discharged assumption enters into the realizing terms, heavily increasing their complexity. The ncm−FC restriction is here essential in allowing such necessary translation from formulas to terms.

The rules of Intuitionistic Logic

$\boxed{A \vdash A}$ Just take $T_1 :\equiv \lambda x, y.\, y$ and $T :\equiv \lambda x.\, x$. The FVC is obviously satisfied.

$\boxed{\dfrac{A \wedge B}{A} \wedge_l^-}$ We are given that (with $\underline{x} \equiv x_1, \ldots, x_n$)

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, y', y''))\}_{i=1}^n \;\vdash\; A_{\mathtt{D}}(S'(\underline{x}); y') \wedge B_{\mathtt{D}}(S''(\underline{x}); y'')$$

in which we substitute the variables from $\mathcal{V}_{\mathrm{f}}(B) \setminus [\cup_{i=1}^n \mathcal{V}_{\mathrm{f}}(C^i) \cup \mathcal{V}_{\mathrm{f}}(A)]$ and those from y'' with terms 0 of corresponding type. We denote by T' and $\{S_i^\star\}_{i=1}^n$ the (newly obtained after substitution) terms corresponding to S' and $\{S_i\}_{i=1}^n$ respectively. For all $i \in \overline{1,n}$ let $T_i :\equiv \lambda \underline{x}, y'.\, S_i^\star(\underline{x}, y', 0)$. We finally obtain after a $\wedge_l^-$, with the FVC obviously satisfied, that

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y'))\}_{i=1}^n \;\vdash\; A_{\mathtt{D}}(T'(\underline{x}); y') \qquad .$$

[4] See also the comments from the preamble of Chapter 2.

$\boxed{\dfrac{A \wedge B}{B}\wedge_r^-}$ Similar to the previous case.

$\boxed{\dfrac{A\,,\,B}{A \wedge B}\wedge^+}$ We are given, with $\underline{x'} \equiv x'_1, \ldots, x'_n$ and $\underline{x''} \equiv x''_{n+1}, \ldots, x''_m$, that :

$$\{C^i_{\mathtt{D}}(x'_i; T_i(\underline{x'}, y'))\}^n_{i=1} \;\vdash\; A_{\mathtt{D}}(T'(\underline{x'}); y')$$
$$\{C^i_{\mathtt{D}}(x''_i; T_i(\underline{x''}, y''))\}^m_{i=1} \;\vdash\; B_{\mathtt{D}}(T''(\underline{x''}); y'')$$

It has been assumed that $n \leq m$. By one application of $\wedge^+$ we get

$$\frac{\{C^i_{\mathtt{D}}(x'_i; T_i(\underline{x'}, y'))\}^n_{i=1}\;,\;\{C^i_{\mathtt{D}}(x''_i; T_i(\underline{x''}, y''))\}^m_{i=n+1}}{A_{\mathtt{D}}(T'(\underline{x'}); y') \wedge B_{\mathtt{D}}(T''(\underline{x''}); y'')}\;.$$

Let $\underline{x} :\equiv \underline{x'}, \underline{x''}$, $S' :\equiv \lambda\underline{x}.\, T'(\underline{x'})$, $S'' :\equiv \lambda\underline{x}.\, T''(\underline{x''})$ and for $i \in \overline{1, m}$,

$$S_i :\equiv \begin{cases} \lambda\underline{x}, y', y''.\, T_i(\underline{x'}, y') & ,\ \text{if } 1 \leq i \leq n \\ \lambda\underline{x}, y', y''.\, T_i(\underline{x''}, y'') & ,\ \text{if } n < i \leq m \end{cases}$$

Then we obviously have (with the `FVC` immediately satisfied)

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, y', y''))\}^m_{i=1} \;\vdash\; (A \wedge B)_{\mathtt{D}}(\, S'(\underline{x}), S''(\underline{x})\,;\, y', y''\,) \quad .$$

$\boxed{\dfrac{A\,,\,A \to B}{B}\to^-}$ We are given, with $\underline{x'} \equiv x'_1, \ldots, x'_n$ and $\underline{x''} \equiv x''_{n+1}, \ldots, x''_m$, that :

$$\{C^i_{\mathtt{D}}(x'_i; T_i(\underline{x'}, y'))\}^n_{i=1} \;\vdash\; A_{\mathtt{D}}(T'(\underline{x'}); y') \tag{2.4}$$
$$\{C^i_{\mathtt{D}}(x''_i; T_i(\underline{x''}, y'', y))\}^m_{i=n+1} \;\vdash\; (A \to B)_{\mathtt{D}}(\, T''(\underline{x''}), T(\underline{x''})\,;\, y'', y\,) \tag{2.5}$$

It has been assumed that $n \leq m$. It is more convenient to redisplay (2.5) as

$$\{C^i_{\mathtt{D}}(x''_i; T_i(\underline{x''}, y'', y))\}^m_{i=n+1} \;\vdash\; A_{\mathtt{D}}(y''; T''(\underline{x''}, y'', y)) \;\to\; B_{\mathtt{D}}(T(\underline{x''}, y''); y) \tag{2.6}$$

We substitute $\{y' \leftarrow T''(\underline{x''}, T'(\underline{x'}), y)\}$ in (2.4) and $\{y'' \leftarrow T'(\underline{x'})\}$ in (2.6). Then

$$\frac{\{C^i_{\mathtt{D}}(x'_i; T_i(\underline{x'}, T''(\underline{x''}, T'(\underline{x'}), y)))\}^n_{i=1}\,,\;\{C^i_{\mathtt{D}}(x''_i; T_i(\underline{x''}, T'(\underline{x'}), y))\}^m_{i=n+1}}{B_{\mathtt{D}}(T(\underline{x''}, T'(\underline{x'})); y)} \tag{2.7}$$

is obtained by an $\rightarrow^-$ applied to (2.6) and substituted (2.4). In (2.7) we substitute the variables from $\mathcal{V}_{\mathrm{f}}(A) \setminus [\cup_{i=1}^m \mathcal{V}_{\mathrm{f}}(C^i) \cup \mathcal{V}_{\mathrm{f}}(B)]$ with terms $\mathtt{0}$ of corresponding type. We denote by $T^\star, T'_\star, T''_\star$ and $\{T_i^\star\}_{i=1}^n$ the (newly obtained after substitution) terms corresponding to T, T', T'' and $\{T_i\}_{i=1}^n$ respectively. Let $\underline{x} :\equiv \underline{x'}, \underline{x''}$ and then let $S :\equiv \lambda \underline{x}.\, T^\star(\underline{x''}, T'_\star(\underline{x'}))$ and

$$S_i \quad :\equiv \quad \begin{cases} \lambda \underline{x}, y.\, T_i^\star(\underline{x'}, T''_\star(\underline{x''}, T'_\star(\underline{x'}), y)) \ , \text{ if } 1 \leq i \leq n \\ \lambda \underline{x}, y.\, T_i^\star(\underline{x''}, T'_\star(\underline{x'}), y) \qquad\quad \ , \text{ if } n < i \leq m \end{cases}$$

We then immediately obtain (with the $\mathtt{FVC}$ obviously satisfied)

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, y))\}_{i=1}^m \ \vdash \ B_{\mathtt{D}}(S(\underline{x}); y) \qquad .$$

$$\boxed{\dfrac{[u\colon A] \ \ldots \ / B}{A \rightarrow B} \ \rightarrow^+}$$

We are given, with $n \geq 1$ (for $n < 1$ skip to the end), $\underline{z} \equiv \overbrace{z, \ldots, z}^{n+1}$ and $\underline{x} \equiv x_{n+2}, \ldots, x_m$, that :

$$\{A_{\mathtt{D}}(z; T_i(\underline{z}, \underline{x}, y))\}_{i=1}^{n+1} \ , \ \{C^i_{\mathtt{D}}(x_i; T_i(\underline{z}, \underline{x}, y))\}_{i=n+2}^m \ \vdash \ B_{\mathtt{D}}(T(\underline{z}, \underline{x}); y) \quad (2.8)$$

It has been assumed that $n+1 \leq m$, where $n+1$ is the number of copies of the assumption formula A which get discharged in this $\rightarrow^+$. Each of these $n+1$ instances of A produces the same unique tuple z of existential variables under the $\mathtt{LD}$-interpretation, see the Convention 2.3 following the Definition 2.1. The $\mathtt{ncm{-}FC}(A)$ constraint ensures that the tuples $\{T_i\}_{i=1}^{n+1}$ are all of length zero (denoted $\sqcup$), i.e., A is computationally $\mathtt{LD}$-irrelevant (see Definition 1.18), or otherwise $A_{\mathtt{D}}$ is quantifier-free (pre-condition for the association to A of so-called "Dialectica terms", see Definition 2.6). Only in the former case can we directly discharge in a single $\rightarrow^+$ all $n+1 \geq 2$ copies of $A_{\mathtt{D}}(z; \sqcup)$ from the $\mathtt{LD}$-interpretation of the premise of this rule, see (2.8). In contrast, for the latter case we must first *equalize* the assumptions involving the $n+1$ terms $\{T_i\}_{i=1}^{n+1}$. This is because the terms $\{T_i\}_{i=1}^{n+1}$ can be mutually different since they are extracted from the various different sub-proofs which involve the different copies of A in the same parcel u. We hereafter treat this case. We achieve the *equalizing* of $\{T_i\}_{i=1}^{n+1}$ in one single (composed) step[5], using the n-selector $\mathtt{If}_\tau^n$. For all $i \in \overline{1,n}$ let T^i abbreviate $T_i(\underline{z}, \underline{x}, y)$ (recall that $\underline{z} \equiv z, \ldots, z$) and let

$$\widetilde{S} \ :\equiv \ \lambda \underline{x}, z, y.\, \mathtt{If}_\tau^n(\mathtt{t}^{\mathtt{D}}_A[z; T^1], \ldots, \mathtt{t}^{\mathtt{D}}_A[z; T^n], T_{n+1}(\underline{z}, \underline{x}, y), T^n, \ldots, T^1) \quad (2.9)$$

[5] Jørgensen's solution uses here n steps which correspond to the simulation of $\mathtt{If}_\tau^n$ in terms of n instances of $\mathtt{If}_\tau^1$, see also the comments in the preamble of Chapter 2.

By letting $\{p_i \leftarrow \mathtt{t}^{\mathtt{D}}_A[z;T^i]\}^n_{i=1}$ in $\mathtt{LmIf}^n_\tau$ (see (1.35)) and also using (2.2) we get

$$\vdash \ \wedge^{i-1}_{j=1} A_{\mathtt{D}}(z;T^j) \wedge \rceil A_{\mathtt{D}}(z;T^i) \ \rightarrow \ \widetilde{S}(\underline{x},z,y) = T_i(\underline{z},\underline{x},y)$$

for all $i \in \overline{1,n}$ and (for the above and below we used β-equality, hence $\mathtt{CMP}$)

$$\vdash \ \wedge^{n}_{i=1} A_{\mathtt{D}}(z;T^i) \ \rightarrow \ \widetilde{S}(\underline{x},z,y) = T_{n+1}(\underline{z},\underline{x},y)$$

from which we further obtain

$$\vdash \ [\,\wedge^{i-1}_{j=1} A_{\mathtt{D}}(z;T^j) \wedge \rceil A_{\mathtt{D}}(z;T^i)\,] \rightarrow [\,A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \rightarrow \wedge^{n+1}_{k=1} A_{\mathtt{D}}(z;T^k)\,]$$
$$\vdash \ [\,\wedge^{n}_{j=1} A_{\mathtt{D}}(z;T^j)\,] \ \rightarrow \ [\,A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \rightarrow \wedge^{n+1}_{k=1} A_{\mathtt{D}}(z;T^k)\,]$$

for all $i \in \overline{1,n}$. We used one $\mathtt{CMP}$[6] in each of the $n+1$ above deductions and an $\mathtt{AxEFQ}$ in each of the the first n of these. Since $A_{\mathtt{D}}$ is $\mathtt{qfr}$[7] (due to the $\mathtt{ncm-FC}$ restriction on A), it follows by multiple Case Distinction (1.23) that

$$\vdash \ A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \rightarrow \wedge^{n+1}_{i=1} A_{\mathtt{D}}(z;T^i) \tag{2.10}$$

hence for all $i \in \overline{1,n+1}$ we obtain

$$A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \ \vdash \ A_{\mathtt{D}}(z;T_i(\underline{z},\underline{x},y)) \tag{2.11}$$

We now sequentially discharge all $n+1$ assumptions $A_{\mathtt{D}}$ of (2.8) in $n+1$ sequential applications of $\rightarrow^+$ (with 1-contraction) and thus obtain

$$\{C^i_{\mathtt{D}}(x_i;T_i(\underline{z},\underline{x},y))\}^m_{i=n+2} \ \vdash \ \{A_{\mathtt{D}}(z;T_i(\underline{z},\underline{x},y))\}^{n+1}_{i=1} \ \rightarrow \ B_{\mathtt{D}}(T(\underline{z},\underline{x});y) \tag{2.12}$$

We combine the proof (2.12) with the $n+1$ proofs of $\{A_{\mathtt{D}}(z;T_i(\underline{z},\underline{x},y))\}^{n+1}_{i=1}$ which are obtained above at (2.11) in $n+1$ applications of $\rightarrow^-$ to conclude, with $\{S_i :\equiv \lambda\underline{x},z.\,T_i(\underline{z},\underline{x})\}^m_{i=n+2}$ and $S :\equiv \lambda\underline{x},z.\,T(\underline{z},\underline{x})$, that

$$\{A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y))\}^{n+1}_{i=1}\,,\ \{C^i_{\mathtt{D}}(x_i;S_i(\underline{x},z,y))\}^m_{i=n+2} \ \vdash \ B_{\mathtt{D}}(S(\underline{x},z);y) \quad .$$

[6] Since $A_{\mathtt{D}}$ is quantifier-free, the restriction on undischarged assumptions is respected.

[7] Because of the multiple Decidability for quantifier-free formulas (see (1.24)) we have

$$\vdash \ \vee^n_{i=1}[\,\wedge^{i-1}_{j=1} A_{\mathtt{D}}(z;T^j) \wedge \rceil A_{\mathtt{D}}(z;T^i)\,] \vee [\,\wedge^n_{j=1} A_{\mathtt{D}}(z;T^j)\,] \qquad ,$$

from which (2.10) would follow immediately using (1.19). However, this method requires (because of our definition of $\vee$) the inclusion of $\exists$ in the verifying system, which we want to avoid (since we can) whenever the proof $\mathcal{P}$ at input does not make use of $\exists$.

Since they are all equal, we are now able to cancel all $n+1 \geq 2$ assumptions $\{A_{\mathtt{D}}(z; \widetilde{S}(\underline{x}, z, y))\}_{i=1}^{n+1}$ in a single $\rightarrow^+$ with $(n+1)$-contraction and thus get

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, z, y))\}_{i=n+2}^{m} \;\vdash\; A_{\mathtt{D}}(z; \widetilde{S}(\underline{x}, z, y)) \rightarrow B_{\mathtt{D}}(S(\underline{x}, z); y) \quad .$$

Notice that $\mathtt{t}^{\mathtt{D}}_A$ introduces in $\widetilde{S}$ new occurrences of the free variables of A. We finally obtain, with the $\mathtt{FVC}$ obviously satisfied, that

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, z, y))\}_{i=n+2}^{m} \;\vdash\; (A \rightarrow B)_{\mathtt{D}}(\, \widetilde{S}(\underline{x}), S(\underline{x})\,; z, y) \;.$$

In the case when exactly one copy of A gets cancelled we can use the same extracted terms except that the equation which defines $\widetilde{S}$ is replaced by $\widetilde{S} :\equiv \lambda \underline{x}, z, y.\, T_1(\underline{z}, \underline{x}, y)$[in fact $\underline{z} \equiv z$]. We now consider the case when no copy of A gets cancelled. We are given that $\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^{m} \vdash B_{\mathtt{D}}(T(\underline{x}); y)$ to which we can apply an $\rightarrow^+$ in which no copy of the assumption gets cancelled to obtain

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^{m} \;\vdash\; A_{\mathtt{D}}(z; \mathtt{0}) \rightarrow B_{\mathtt{D}}(T(\underline{x}); y) \quad .$$

We then simply define $\{S_i :\equiv \lambda \underline{x}, z, y.\, T_i(\underline{x}, y)\}_{i=1}^{m}$, $\widetilde{S} :\equiv \lambda \underline{x}, z, y.\, \mathtt{0} \equiv \mathtt{0}$ and $S :\equiv \lambda \underline{x}, z.\, T(\underline{x})$ to finally obtain, with the $\mathtt{FVC}$ obviously satisfied, that

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, z, y))\}_{i=1}^{m} \;\vdash\; A_{\mathtt{D}}(z; \widetilde{S}(\underline{x}, z, y)) \rightarrow B_{\mathtt{D}}(S(\underline{x}, z); y) \quad .$$

$\boxed{\dfrac{\forall z\, A(z)}{A(t)} \forall^-_{z,t}}$ We are given that

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, z^\dagger, y))\}_{i=1}^{n} \;\vdash\; A_{\mathtt{D}}(S(\underline{x}, z^\dagger); y; z^\dagger) \qquad ,$$

where $\underline{x} \equiv x_1, \ldots, x_n$ and (by its construction) $z^\dagger$ is not free in $\{C^i_{\mathtt{D}}\}_{i=1}^{n}$ and (also due to the $\mathtt{FVC}$) $z^\dagger$ is not free in S and/or $\{S_i\}_{i=1}^{n}$. By substituting $\{z^\dagger \leftarrow t\}$ we therefore obtain that $\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, t, y))\}_{i=1}^{n} \vdash A_{\mathtt{D}}(S(\underline{x}, t); y; t)$. Then by defining $\{T_i :\equiv \lambda \underline{x}.\, S_i(\underline{x}, t)\}_{i=1}^{n}$ and $T :\equiv \lambda \underline{x}.\, S(\underline{x}, t)$ we conclude, also using $\mathtt{VC}_2$, modulo some applications of $\mathtt{CMP}$, with $\mathtt{FVC}$ obviously satisfied, that:

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^{n} \;\vdash\; A_{\mathtt{D}}(T(\underline{x}); y; t) \qquad .$$

$\boxed{\dfrac{A(z)}{\forall z\, A(z)} \forall^+_z}$ We are given that

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^{n} \;\vdash\; A_{\mathtt{D}}(T(\underline{x}); y; z) \qquad ,$$

where $\underline{x} \equiv x_1, \ldots, x_n$ and (by $\mathtt{VC}_1$) z is not free in the assumptions $\{C^i\}_{i=1}^n$. Due to the rules for variable naming in the LD-interpretation of a formula, z is also not among $\underline{x}, y$ (see Definition 2.1). On the other hand, z may be free in T and/or $\{T_i\}_{i=1}^n$. Let $S :\equiv \lambda \underline{x}, z.\, T(\underline{x})$ and $\{S_i :\equiv \lambda \underline{x}, z.\, T_i(\underline{x})\}_{i=1}^n$. We then obtain, after some applications of the CMP rule and the substitution $\{z \leftarrow z^\dagger\}$, that (with the FVC obviously satisfied):

$$\{C^i_{\mathtt{D}}(x_i; S_i(\underline{x}, z^\dagger, y))\}_{i=1}^n \;\vdash\; (\forall z A(z))_{\mathtt{D}}(S(\underline{x}); z^\dagger, y) \qquad .$$

$\boxed{\dfrac{\overline{\forall} z\, A(z)}{A(t)}\,\overline{\forall}^{-}_{z,t}}$ We are given that

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^n \;\vdash\; \forall z A_{\mathtt{D}}(T(\underline{x}); y; z) \qquad ,$$

to which we can immediately apply $\forall^-_{z,t}$ since $\mathtt{VC}_2(z,t)$ is obviously respected (due to the policy of variable naming in the LD-interpretation of a formula, see Definition 2.1). We then obtain, with the FVC obviously satisfied, that

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^n \;\vdash\; A_{\mathtt{D}}(T(\underline{x}); y; t) \qquad .$$

We here also used that, due to the FVC, z is not free in any of T or $\{T_i\}_{i=1}^n$.

$\boxed{\dfrac{\mathcal{P}\colon A(z)}{\overline{\forall} z\, A(z)}\,\overline{\forall}^{+}_{z}}$ We are given that

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^n \;\vdash\; A_{\mathtt{D}}(T(\underline{x}); y; z) \qquad ,$$

where z is neither in $\cup_{i=1}^n \mathcal{V}_{\mathrm{f}}(C^i)$ (because of $\mathtt{VC}_1(z)$) nor among $\underline{x}, y$ (due to the rules for variable naming in the LD-interpretation of a formula, see Definition 2.1). The fact that z is not free in any of $T, \{T_i\}_{i=1}^n$ is ensured mainly by $\mathtt{VC}_3(z, \mathcal{P})$: the exception in $\mathtt{VC}_3(z, \mathcal{P})$ is taken care of by the FVC applied to all sub-proofs of $\mathcal{P}$. Thus, even in the exceptional cases, z still is not free in any of $T, \{T_i\}_{i=1}^n$, because of the FVC. Since the pre-condition $\mathtt{VC}_1(z)$ is established, we can apply a $\forall^+_z$ in the verifying proof to obtain, with the FVC clearly satisfied also due to $\mathtt{VC}_3(z, \mathcal{P})$, that

$$\{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^n \;\vdash\; \forall z\, A_{\mathtt{D}}(T(\underline{x}); y; z) \qquad .$$

The axioms of Intuitionistic Logic

$\boxed{\mathtt{Ax}\exists^{-}:\ \exists z_1\, A(z_1) \wedge \forall z_2\,[\,A(z_2) \to B\,] \to B}$ The LD-interpretation of $\mathtt{Ax}\exists^{-}$ is (for legibility we abbreviate below by $\cdot :\equiv z, x, y, u, v$)

$$\exists\, Y, Z, X, V, U\ \forall\, z, x, y, u, v$$
$$\left[\begin{array}{c} A_{\mathtt{D}}(x; Y(x,y,u,v); z) \\ \wedge \\ [A_{\mathtt{D}}(X(\cdot); y(Z(\cdot), X(\cdot), V(\cdot)); Z(\cdot)) \to B_{\mathtt{D}}(u(Z(\cdot), X(\cdot)); V(\cdot))] \\ \to \\ B_{\mathtt{D}}(U(z,x,y,u); v) \end{array}\right]$$

for which a realizing tuple is T_Y, T_Z, T_X, T_V, T_U where

$$\begin{array}{lcl} T_Y & :\equiv & \lambda z, x, y, u, v.\, y(z, x, v) \\ T_Z & :\equiv & \lambda z, x, y, u, v.\, z \\ T_X & :\equiv & \lambda z, x, y, u, v.\, x \\ T_V & :\equiv & \lambda z, x, y, u, v.\, v \\ T_U & :\equiv & \lambda z, x, y, u.\, u(z, x) \end{array}$$

$\boxed{\mathtt{Ax}\exists^{+}:\ \forall z_1\,[\,A(z_1) \to \exists z_2\, A(z_2)\,]}$ The LD-interpretation of $\mathtt{Ax}\exists^{+}$ is

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

for which a realizing tuple is T_Y, T_Z, T_X where

$$\begin{array}{lcl} T_Y & :\equiv & \lambda z, x, y.\, y \\ T_Z & :\equiv & \lambda z, x.\, z \\ T_X & :\equiv & \lambda z, x.\, x \end{array}$$

$\boxed{\mathtt{Ax}\overline{\exists}^{-}:\ \overline{\exists} z_1\, A(z_1) \wedge \overline{\forall} z_2\,[\,A(z_2) \to B\,] \to B}$ The LD-interpretation of $\mathtt{Ax}\overline{\exists}^{-}$is (for legibility we abbreviate below by $\cdot :\equiv x, y, u, v$)

$$\exists Y, X, V, U\ \forall x, y, u, v$$
$$\left[\begin{array}{c} \exists z_1\, A_{\mathtt{D}}(x; Y(x,y,u,v); z_1, a) \\ \wedge \\ \forall z_2\,[A_{\mathtt{D}}(X(\cdot); y(X(\cdot), V(\cdot)); z_2, a) \to B_{\mathtt{D}}(u(X(\cdot)); V(\cdot); b)] \\ \to \\ B_{\mathtt{D}}(U(x,y,u); v; b) \end{array}\right]$$

for which a realizing tuple is T_Y, T_X, T_V, T_U where

$$\begin{array}{lcl} T_Y & :\equiv & \lambda x, y, u, v.\, y(x, v) \\ T_X & :\equiv & \lambda x, y, u, v.\, x \\ T_V & :\equiv & \lambda x, y, u, v.\, v \\ T_U & :\equiv & \lambda x, y, u.\, u(x) \end{array}$$

The verification is ensured by an instance of $\mathtt{Ax}\exists^{-}$.

$\boxed{\mathtt{Ax}\overline{\exists}^{+}:\ \overline{\forall} z_1\,[\,A(z_1)\ \rightarrow\ \overline{\exists} z_2\,A(z_2)\,]}$ The LD-interpretation of $\mathtt{Ax}\overline{\exists}^{+}$ is

$$\exists Y, X \forall x, y \forall z_1 [A_{\mathtt{D}}(x; Y(x, y); z_1, a) \rightarrow \exists z_2 A_{\mathtt{D}}(X(x); y; z_2, a)]$$

for which a realizing tuple is T_Y, T_X where

$$\begin{array}{lcl} T_Y & :\equiv & \lambda x, y.\, y \\ T_X & :\equiv & \lambda x.\, x \end{array}$$

The verification is ensured by an instance of $\mathtt{Ax}\exists^{+}$.

The boolean axioms

The axiom AxTRH, is simply quantifier-free. For the others we have:

$\boxed{\mathtt{AxFLS}:\ \mathtt{at}(\mathtt{ff})\ \rightarrow\ A}$ The LD-interpretation of AxFLS is

$$\exists x\, \forall y\, (\mathtt{at}(\mathtt{ff})\ \rightarrow\ A_{\mathtt{D}}(x; y))$$

for which a realizing tuple is $T_x :\equiv \mathtt{0}$.

$\boxed{\mathtt{AxBIA}:\ A(\mathtt{tt}) \wedge A(\mathtt{ff})\ \rightarrow\ \forall p^{o} A(p)}$ The LD-interpretation of AxBIA is

$$\exists Y_1, Y_2, X\ \forall p^{\dagger}, x_1, x_2, y$$
$$\left(\begin{array}{c} A_{\mathtt{D}}(x_1; Y_1(p^{\dagger}, x_1, x_2, y); \mathtt{tt})\ \wedge\ A_{\mathtt{D}}(x_2; Y_2(p^{\dagger}, x_1, x_2, y); \mathtt{ff}) \\ \longrightarrow \\ A_{\mathtt{D}}(X(p^{\dagger}, x_1, x_2); y; p^{\dagger}) \end{array} \right)$$

for which a realizing tuple is T_{Y_1}, T_{Y_2}, T_X where

$$\begin{array}{lcl} T_{Y_1} & :\equiv & \lambda p^{\dagger}, x_1, x_2, y.\, y \\ T_{Y_2} & :\equiv & \lambda p^{\dagger}, x_1, x_2, y.\, y \\ T_Z & :\equiv & \mathtt{If}_{\tau} \end{array}$$

and the verifying proof uses an instance of AxBIA for a qfr formula and $\mathtt{AxIf}_{\tau}$.

Arithmetical axioms and rules

The axiom AxEFQ is quantifier-free and the axioms REF and AxEQL are either quantifier-free (at ground type) or purely universal (at higher-order type). Therefore none of them produces any realizing term. The quantifier-free axioms verify themselves. For the purely universal axioms the verifying proof is given by simple applications of $\forall^-$ to themselves. The soundness of LD-interpretation for the rules SYM, TRZ and SUB at higher-order type is immediate since the realizing terms involved are just simple projections of shape $\lambda x_1, \ldots, x_n.\, x_i$ (with $i \in \overline{1,n}$) and the verifying proofs only involve AxEQL and some $\forall^+$ followed by $\forall^-$. We now present the treatment of the remaining cases:

$$\frac{A_0 \ \ldots \ / s =_\tau t}{B(s) \to B(t)}\,\mathtt{CMP}$$

We have that $B(z^\tau, a)^{\mathtt{D}} \equiv \exists u \forall v B_{\mathtt{D}}(u; v; z^\tau, a)$, hence

$$(B(x) \to B(y))^{\mathtt{D}} \ \equiv \ \exists U, V \forall u, v\, B_{\mathtt{D}}(u; V(u,v); x, a) \to B_{\mathtt{D}}(U(u); v; y, a)\ .$$

Since $A_0^{\mathtt{D}} = (A_0)_{\mathtt{D}} = A_0$ we are given that $A_0 \vdash s(z_1, \ldots, z_n) =_\iota t(z_1, \ldots, z_n)$ to which we can (since none of $z_1, \ldots, z_n$ appears in A_0) apply n times $\forall^+$ to (re)obtain $A_0 \vdash s =_\tau t$. To this we apply a CMP for $B'(z) \equiv B_{\mathtt{D}}(u; v; z, a)$ and thus obtain

$$A_0 \ \vdash \ B_{\mathtt{D}}(u; v; s, a) \to B_{\mathtt{D}}(u; v; t, a)\ .$$

Let $T_V :\equiv \lambda u, v.\, v$ and $T_U :\equiv \lambda u.\, u$. It immediately follows (also using some other CMP) that

$$A_0 \ \vdash \ B_{\mathtt{D}}(u; T_V(u,v); s, a) \ \to \ B_{\mathtt{D}}(T_U(u); v; t, a)$$

which rewrites as

$$(A_0)_{\mathtt{D}} \ \vdash \ (B(s) \to B(t))_{\mathtt{D}}(T_U, T_V; u, v)\ .$$

$$\frac{\begin{array}{cc}\emptyset & \emptyset \\ \vdots & \vdots \\ A(\mathtt{0}) & \forall z\,(A(z) \to A(\mathtt{Suc}\,z))\end{array}}{\forall z\, A(z)}\,\mathtt{IR}_{\mathrm{o}}$$

We are given that

(by abuse of notation we use the same z in the verifying proof)

$$\emptyset \ \vdash \ A_{\mathtt{D}}(T'; y'; \mathtt{0}) \tag{2.13}$$

$$\emptyset \ \vdash \ A_{\mathtt{D}}(y''; T_1''(z, y'', y); z) \to A_{\mathtt{D}}(T_2''(z, y''); y; \mathtt{Suc}\,z) \tag{2.14}$$

We define by simultaneous primitive recursion in higher types the terms T such that $T(\mathtt{0}) = T'$ and $T(\mathtt{Suc}z) = T''_2(z, T(z))$. By $\forall^+$, (2.13) becomes

$$\emptyset \;\vdash\; \forall y\, A_{\mathtt{D}}(T(\mathtt{0}); y; \mathtt{0}) \tag{2.15}$$

We substitute $\{y'' \leftarrow T(z)\}$ in (2.14) and obtain

$$\emptyset \;\vdash\; A_{\mathtt{D}}(T(z); T''_1(z, T(z), y); z) \to A_{\mathtt{D}}(T(\mathtt{Suc}z); y; \mathtt{Suc}z) \tag{2.16}$$

from which we can produce the following proof

$$\dfrac{\dfrac{\dfrac{\dfrac{[\,\forall y\, A_{\mathtt{D}}(T(z); y; z)\,]}{\forall y\, A_{\mathtt{D}}(T(z); y; z)}\,\forall^-}{A_{\mathtt{D}}(T(z); T''_1(z, T(z), y); z)} \quad (2.16)}{A_{\mathtt{D}}(T(\mathtt{Suc}z); y; \mathtt{Suc}z)}\,\to^-}{\forall y\, A_{\mathtt{D}}(T(\mathtt{Suc}z); y; \mathtt{Suc}z)}\,\forall^+}{\forall y\, A_{\mathtt{D}}(T(z); y; z) \to \forall y\, A_{\mathtt{D}}(T(\mathtt{Suc}z); y; \mathtt{Suc}z)}\,\to^+$$

where-from by a final $\forall^+_z$ we obtain

$$\emptyset \;\vdash\; \forall z\,(\,\forall y\, A_{\mathtt{D}}(T(z); y; z) \;\to\; \forall y\, A_{\mathtt{D}}(T(\mathtt{Suc}z); y; \mathtt{Suc}z)\,) \tag{2.17}$$

From (2.15) and (2.17) we obtain by $\mathtt{IR}_{\mathrm{o}}$ followed by $\forall^-$ w.r.t. z and y that $\emptyset \vdash A_{\mathtt{D}}(T(z^\dagger); y; z^\dagger)$, hence, with the FVC obviously satisfied,

$$\emptyset \;\vdash\; (\forall z A(z))_{\mathtt{D}}(T; z^\dagger, y)$$

Other axioms and rules

$\boxed{\mathtt{AxAC}:\ \forall x\,\exists y\, B(x,y) \;\to\; \exists Y\,\forall x\, B(x, Y(x))}$ Since $B^{\mathtt{D}} \equiv \exists u\,\forall v\, B_{\mathtt{D}}(u; v)$ we have

$$[\,\forall x\,\exists y\, B(x,y)\,]^{\mathtt{D}} \equiv \exists Y^\dagger, U\,\forall x^\dagger, v^\dagger\, B_{\mathtt{D}}(U(x^\dagger); v^\dagger; x^\dagger, Y^\dagger(x^\dagger)) \equiv [\,\exists Y\,\forall x\, B(x, Y(x))\,]^{\mathtt{D}}$$

hence the LD-interpretation of AxAC is

$$\exists X, V, Y^\dagger, U\;\forall y^\dagger, u^\dagger, x^\dagger, v^\dagger$$
$$\left(\begin{array}{c} B_{\mathtt{D}}(u^\dagger(X(y^\dagger, u^\dagger, x^\dagger, v^\dagger)); V(y^\dagger, u^\dagger, x^\dagger, v^\dagger); X(y^\dagger, u^\dagger, x^\dagger, v^\dagger), y^\dagger(X(y^\dagger, u^\dagger, x^\dagger, v^\dagger))) \\ \longrightarrow \\ B_{\mathtt{D}}(U(y^\dagger, u^\dagger, x^\dagger); v^\dagger; x, Y^\dagger(y^\dagger, u^\dagger, x^\dagger)) \end{array}\right)$$

for which a realizing tuple is

$$\begin{array}{rcl} T_X & :\equiv & \lambda y^\dagger, u^\dagger, x^\dagger, v^\dagger.\, x^\dagger \\ T_V & :\equiv & \lambda y^\dagger, u^\dagger, x^\dagger, v^\dagger.\, v^\dagger \\ T_{Y^\dagger} & :\equiv & \lambda y^\dagger, u^\dagger.\, y^\dagger \\ T_U & :\equiv & \lambda y^\dagger, u^\dagger.\, u^\dagger \end{array}$$

$$\boxed{\mathtt{AxMK}:\ \exists^{\mathtt{cl}} z\, A^{\mathtt{nc}}(z) \to \exists z\, A^{\mathtt{nc}}(z) \quad \text{and} \quad \mathtt{AxMK}_1:\ \neg\neg\, \exists z A^{\mathtt{nc}}(z) \to \exists z A^{\mathtt{nc}}(z)}$$

Since $[\,\exists^{\mathtt{cl}} z\, A^{\mathtt{nc}}(z)\,]^{\mathtt{D}} \equiv [\,\neg\neg\, \exists z A^{\mathtt{nc}}(z)\,]^{\mathtt{D}} \equiv \exists z^\dagger\, \neg\neg\, A(z^\dagger)$, where A is the regular-quantifier translation of $A^{\mathtt{nc}}$ in the sense of Footnote 17 (see also Remark 2.8), the LD-interpretation of both variants AxMK and $\mathtt{AxMK}_1$ of Markov's principle gets to be $\exists Z\, \forall z^\dagger\, [\,\neg\neg\, A(z^\dagger) \to A(Z(z^\dagger))\,]$, which is simply realized as usual by $T_Z :\equiv \lambda z^\dagger.\, z^\dagger$. Nonetheless, the verifying proof uses stability for all formulas, hence this is the unique place where AxSTAB is needed in the verifying proof. Notice that the variant $\mathtt{AxMK}_2:\ \neg\neg\, \exists z\, A^{\mathtt{nc}}(z) \to \exists z\, \neg\neg\, A^{\mathtt{nc}}(z)$ of Markov's principle, which was preferred by the authors of [57] because of complexity considerations (see also Footnote 16) is here much simpler because no stability is needed for the verification. Hence the use of $\mathtt{AxMK}_2$ instead of AxMK or $\mathtt{AxMK}_1$ would allow to avoid full classical logic in the verifying systems.

$$\boxed{\mathtt{AxIP}^{-\exists}_{+\forall}:\quad [\,A^{-\exists}_{+\forall} \to \exists y\, B(y)\,] \to \exists y\, [\,A^{-\exists}_{+\forall} \to B(y)\,]}$$

Recall the usual notation $B^{\mathtt{D}} \equiv \exists u\, \forall v\, B_{\mathtt{D}}(u; v)$. On the other hand the more special kind of formula $A^{-\exists}_{+\forall}$ (see Section 1.4 for its definition) has the structural property that its LD-interpretation does not have an existential side (i.e., its existential side is empty, by Definition 2.1), hence we here denote rather unusually[8] $(A^{-\exists}_{+\forall})^{\mathtt{D}} :\equiv \forall x\, A_{\mathtt{D}}(\sqcup; x)$, and even shorter, $(A^{-\exists}_{+\forall})^{\mathtt{D}} :\equiv \forall x A_{\mathtt{D}}(x)$. Thus:

$$\begin{array}{rcl} [\,A^{-\exists}_{+\forall} \to \exists y\, B(y)\,]^{\mathtt{D}} & \equiv & \exists X, y^\dagger, u^\dagger\, \forall v^\dagger\, [\,A_{\mathtt{D}}(X(v^\dagger)) \to B_{\mathtt{D}}(u^\dagger; v^\dagger; y^\dagger)\,] \\ (\,\exists y\, [\,A^{-\exists}_{+\forall} \to B(y)\,]\,)^{\mathtt{D}} & \equiv & \exists y^\dagger, X, u^\dagger\, \forall v^\dagger\, [\,A_{\mathtt{D}}(X(v^\dagger)) \to B_{\mathtt{D}}(u^\dagger; v^\dagger; y^\dagger)\,] \end{array}$$

hence the LD-interpretation of $\mathtt{AxIP}^{-\exists}_{+\forall}$ is

$$\exists V, Y, X, U\ \forall x^\dagger, y^\dagger, u^\dagger, v^\dagger$$
$$\left(\begin{array}{c} A_{\mathtt{D}}(x^\dagger(V(x^\dagger, y^\dagger, u^\dagger, v^\dagger))) \to B_{\mathtt{D}}(u^\dagger; V(x^\dagger, y^\dagger, u^\dagger, v^\dagger); y^\dagger) \\ \longrightarrow \\ A_{\mathtt{D}}(X(x^\dagger, y^\dagger, u^\dagger, v^\dagger)) \to B_{\mathtt{D}}(U(x^\dagger, y^\dagger, u^\dagger); v^\dagger; Y(x^\dagger, y^\dagger, u^\dagger)) \end{array}\right)$$

[8] Since x is normally used for the existential side of $A^{\mathtt{D}}$.

for which a realizing tuple is

$$\begin{aligned} T_V &:\equiv \lambda x^\dagger, y^\dagger, u^\dagger, v^\dagger.\, v^\dagger \\ T_Y &:\equiv \lambda x^\dagger, y^\dagger, u^\dagger.\, y^\dagger \\ T_X &:\equiv \lambda x^\dagger, y^\dagger, u^\dagger.\, x^\dagger \\ T_U &:\equiv \lambda x^\dagger, y^\dagger, u^\dagger.\, u^\dagger \end{aligned}$$

$$\boxed{\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}: \quad [\,A^{-\exists}_{+\forall} \to \exists^{\mathtt{cl}} y\, B^{+\exists}_{-\forall}(y)\,] \to \exists^{\mathtt{cl}} y\,[\,A^{-\exists}_{+\forall} \to B^{+\exists}_{-\forall}(y)\,]}$$

Just like above, let $(A^{-\exists}_{+\forall})^{\mathtt{D}} :\equiv \forall x A_{\mathtt{D}}(\sqcup; x)$, and even shorter, $(A^{-\exists}_{+\forall})^{\mathtt{D}} :\equiv \forall x A_{\mathtt{D}}(x)$. Similarly, let $(B^{+\exists}_{-\forall}(y))^{\mathtt{D}} :\equiv \exists u B_{\mathtt{D}}(u; \sqcup; y)$, which we choose to short, despite the ambiguity, as $(B^{+\exists}_{-\forall}(y))^{\mathtt{D}} :\equiv \exists u B_{\mathtt{D}}(u; y)$. Then by means of (2.3) we immediately obtain that $(\exists^{\mathtt{cl}} y B^{+\exists}_{-\forall}(y))^{\mathtt{D}} :\equiv \exists y, u \neg\neg B_{\mathtt{D}}(u; y)$ and further combined:

$$[\,A^{-\exists}_{+\forall} \to \exists^{\mathtt{cl}} y\, B^{+\exists}_{-\forall}(y)\,]^{\mathtt{D}} \equiv \exists x, y, u\,[A_{\mathtt{D}}(x) \to \neg\neg B_{\mathtt{D}}(u; y)] \tag{2.18}$$

$$[\,A^{-\exists}_{+\forall} \to B^{+\exists}_{-\forall}(y)\,]^{\mathtt{D}} \equiv \exists x, u\,[A_{\mathtt{D}}(x) \to B_{\mathtt{D}}(u; y)] \tag{2.19}$$

Now from (2.19) we immediately obtain, again by using (2.3), that

$$(\exists^{\mathtt{cl}} y\,[\,A^{-\exists}_{+\forall} \to B^{+\exists}_{-\forall}(y)\,])^{\mathtt{D}} \equiv \exists x, y, u\, \neg\neg[A_{\mathtt{D}}(x) \to B_{\mathtt{D}}(u; y)] \tag{2.20}$$

Since $A_{\mathtt{D}}, B_{\mathtt{D}} \in \mathcal{F}^{\exists}$, due to (1.32) we can write (in `IL` already)

$$[A_{\mathtt{D}}(x) \to \neg\neg B_{\mathtt{D}}(u; y)] \leftrightarrow \neg\neg[A_{\mathtt{D}}(x) \to B_{\mathtt{D}}(u; y)] \tag{2.21}$$

Finally, from (2.18), (2.20) and (2.21) we immediately figure out that the `LD`-interpretation of $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$ is realized by simple projection functionals.

$$\boxed{\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}: \quad \forall x\, \exists^{\mathtt{cl}} y\; B^{\mathtt{nc}}(x, y) \to \exists^{\mathtt{cl}} Y\, \forall x\; B^{\mathtt{nc}}(x, Y(x))}$$

By (2.3) we have $(\exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x, y))^{\mathtt{D}} \equiv \exists y \neg\neg B(x, y)$, and further

$$[\forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x, y)]^{\mathtt{D}} \equiv \exists y \forall x \neg\neg B(x, y(x)) \tag{2.22}$$

Also by (2.3) we have (above and below B isomorphically corresponds to $B^{\mathtt{nc}}$)

$$[\exists^{\mathtt{cl}} Y \forall x B^{\mathtt{nc}}(x, Y(x))]^{\mathtt{D}} \equiv \exists Y \forall x \neg\neg B(x(Yx), Y(x, x(Yx))) \tag{2.23}$$

Let y realize (2.22) and define $Y :\equiv \lambda z, x.\, y(x)$. From (2.22) we thus have $\forall x', z \neg\neg B(x', Y(z, x'))$, from which we immediately obtain (2.23) by setting

$z :\equiv x$ and $x' :\equiv x(Yx)$. We have thus constructed a purely intuitionistic proof of (2.23) from (2.22) and it should now be clear that the $\mathtt{LD}$-interpretation of $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ does have realizers. In fact, $[\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}]^{\mathtt{D}}$ is now by pure Definition 2.1 (we here also used some convenient renaming of variables in (2.22) and (2.23)):

$$\exists Y, U \forall x, v [\neg\neg B(Y(x,v), x(Y(x,v))) \to \neg\neg B(v(Uxv), U(x,v,v(Uxv)))]$$

for which we can directly construct $Y :\equiv \lambda x, v.\, v(x)$ and $U :\equiv \lambda x, v.\, x$. $\square$

Corollary 2.11 (Exact realizer synthesis by the Light Dialectica [51]) There exists an algorithm which from a proof $\emptyset \vdash A(a)$ in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ or respectively $\mathtt{WeZ}^{\mathtt{nc}+}$ produces exact realizing terms $T[a]$ with a verifying proof $\emptyset \vdash \forall a, y\, A_{\mathtt{D}}(T; y; a)$ in the corresponding system $\mathtt{WeZ}^{\exists,\mathtt{cl}}$ or respectively $\mathtt{WeZ}$. If the input proof in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ does not employ any $\mathtt{AxSTAB}^{\mathtt{nc}}$ or $\mathtt{AxMK}$ in which the pure-$\mathtt{ncm}$ main formula is not $\mathtt{ncm}$-stable then the verifying proof is in $\mathtt{WeZ}^{\exists}$ rather than $\mathtt{WeZ}^{\exists,\mathtt{cl}}$. Stability can be proved in $\mathtt{WeZ}^{\exists}$ already for cases of need.

2.2 The light monotone functional (light monotone Dialectica) interpretation

We describe below the combination of our refinement of Gödel's Dialectica interpretation [61, 4, 45] with its optimization for the extraction of bounds[9] due to Kohlenbach [68].

Theorem 2.12 (Extraction and its soundness for $\mathtt{LMD}$-interpretation) There exists an algorithm which, given at input a proof $\mathcal{P} : \{C^i(a_i)\}_{i=1}^n \vdash A(a')$ in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}$ will produce at output the tuples of terms $\{T_i[\underline{a}]\}_{i=1}^n$ and $T[\underline{a}]$ (where $\underline{a} :\equiv a_1, \ldots, a_n, a'$) together with the following verifying proof in $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$:

$$\vdash \quad \exists X_1 \ldots X_n, X\, [\, \wedge_{i=1}^n (\lambda \underline{a}.\, T_i[\underline{a}]) \;\succeq\; X_i \;\wedge\; (\lambda \underline{a}.\, T[\underline{a}]) \;\succeq\; X \;\wedge$$
$$\forall \underline{a}, \underline{x}, y\, (\, \{C^i_{\mathtt{D}}(x_i; X_i(\underline{a}, \underline{x}, y); a_i)\}_{i=1}^n \;\to\; A_{\mathtt{D}}(X(\underline{a}, \underline{x}); y; a')\,)\,]$$

– where $\underline{x} :\equiv x_1, \ldots, x_n$. We denote this verifying proof which corresponds to $\mathcal{P}$ by $\mathcal{P}_{\mathtt{MD}}$. If the input proof $\mathcal{P}$ is in $\mathtt{WeZ}^{\mathtt{nc}+}_{\mathtt{m}}$ only, then $\mathcal{P}_{\mathtt{MD}}$ is in $\mathtt{WeZ}_{\mathtt{m}}$ plus $\mathtt{Ax}\exists^{+}$ and $\mathtt{Ax}\exists^{-}$ for the newly introduced existential variables $X_1 \ldots X_n, X$ and for the existential variables Φ of those $\mathtt{AxCA}$ necessarily involved by $\mathcal{P}_{\mathtt{MD}}$ only.

[9] See also [33] for a more recent adaptation of Dialectica to the extraction of bounds.

Proof: The wanted algorithm proceeds by recursion on $\mathcal{P}$ and it can be seen as the interleaving of two algorithms. On one side we have the extraction algorithm which is derived from that of Theorem 2.10 by carrying it out with a number of small (but important) modifications concerning:

1. The computationally LD-relevant contractions which are simpler treated by replacing (2.9) with

$$\widetilde{S} \quad :\equiv \quad \lambda \underline{x}, z, y.\, \mathtt{Max}^n_\tau \left(T_{n+1}(\underline{z}, \underline{x}, y)\right) T^n \ldots T^1 \tag{2.24}$$

2. The new Gödel recursor R introduced in the extracted terms by the treatment of the induction rule $\mathtt{IR}_{\mathrm{o}}$ is replaced with its associate $\mathtt{R}^\star$ given by Definition 1.74 / Proposition 1.78.

3. The term t of ForAll Elim $\forall^-_{z,t}$, which had gotten comprised into the extracted terms during the treatment of this rule, is here replaced with its correspondent $t^\star$ obtained from t via the algorithm of Proposition 1.78.

Notice that the issue of program synthesis is separate from that of program verification, hence this (much simpler, hence more efficient) algorithm can be carried out independently in itself. On the other hand we have the algorithm charged with the construction of the verifying proof. The monotone verifying proof not only mimics the verifying proof produced by Theorem 2.10, but acts more like a Hilbert-style counterpart of this Natural Deduction proof. Indeed, the abstractions for the concrete LD-realizing terms, denoted by the (strong) existentially quantified variables $X_1 \ldots X_n$ and X, actually force the verifying proof to keep the assumptions on the right-hand side of the proof-gate $\vdash$, as *implicative* assumptions rather than *open* assumptions. Thus the second algorithm ensures that during a recursion loop, the verifying proof is built from such Hilbert-style-like (i.e., assumption-less) proofs and at the end of the loop a true Hilbert-style-like proof is produced, even though the intermediate proofs of this construction may freely use undischarged assumptions. The second algorithm is thus concerned only with the construction of the verifying proof.

In the monotone context, the virtual existence of the exact realizer is still granted due to the addition of AxCA to the verifying systems. Since $A_{\mathtt{D}}$ is no longer necessarily quantifier-free, its comprehension term $\Phi_{A_{\mathtt{D}}}$ can and must be used instead of $\mathtt{t}^{\mathtt{D}}_A$ in (2.9). Then AxCA ensures that the construction from Theorem 2.10 still works for the Imp Intro rule $\rightarrow^+$, in the sense that a verifying Hilbert-style-like proof for the new exact realizer can be constructed in the new

monotone setting. If t denotes that exact realizer, then the construction of its "canonical" majorant $t^\star$ by means of Proposition 1.78 would actually replace the $\widetilde{S}$ of the modified (2.9) with the $\widetilde{S}$ defined above via (2.24). But with the new definition of the (monotone) extracted terms we actually get to avoid the application of the $\cdot^\star$ algorithm at the end of the whole extraction. The modified construction ensures that the lambda-closure of the monotone extracted terms already is a majorant of the lambda-closure of the exact realizers. This fact has a straightforward proof similar to that of Proposition 1.78, using Lemmas 1.72, 1.75 and 1.77 (hence also using Proposition 1.53)

We left at the end the treatment of axioms Δ, included into $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ via Definition 1.67. These axioms appear in the monotonic systems only, hence they were not given realizers in the proof of Theorem 2.10. Let

$$\forall x^\rho \, \exists y \leq_\sigma rx \, \forall z^\tau \; B^{\mathtt{nc}}(x, y, z)$$

be an axiom formula from Δ, and let

$$\exists Y \leq_{\rho\sigma} r \, \forall x^\rho \, \forall z^\tau \; B(x, Yx, z)$$

be its corresponding formula from the associated isomorphic set $\widetilde{\Delta}$, included as axiom in the verifying system $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$. Notice that this associated formula is nothing but the $\mathtt{LD}$-translation of the original Δ-axiom. Since an exact realizer for such translation is only guaranteed by the strong existence of Y, but it cannot be concretely given as $\mathbf{T}$-term, we can understand why the axioms Δ are characteristic to the monotonic systems. In these arithmetics the merely qualitative information that $Y \leq r$ is sufficient to construct a majorant for Y, which is $r^\star$, via the algorithm of Proposition 1.78, also due to (1.42) and to the fact that r is a closed term. Thus the monotone realizer for such a Δ-axiom is taken to be its associated $r^\star$. $\square$

Remark 2.13 In the formulation of Theorem 2.12 it appears to be technically unavoidable that verifying proofs are without open assumptions. Therefore the Hilbert-style formulation of logic seems to be more suitable for proving soundness of the (light) $\mathtt{MD}$-interpretation. On the other hand Natural Deduction rules are fewer and therefore less inductive-step cases are to be treated. Hence moving back-and-forth over the proof gate in the treatment of Natural Deduction rules appears to be an acceptable compromise.

Corollary 2.14 (Majorant synthesis by the Light $\mathtt{MD}$-interpretation) There exists an algorithm which from a proof $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+} \vdash A(a)$ produces terms

$T[a]$ and a verifying proof $\mathtt{WeZ}^{\exists}_{\mathtt{m}} \vdash \exists x\,[(\lambda a.\,T) \succeq x \wedge \forall a, y\, A_{\mathtt{D}}(x(a); y; a)]$. If the input proof is in $\mathtt{WeZ}^{\mathtt{nc}+}_{\mathtt{m}}$, then the verifying proof is in $\mathtt{WeZ}_{\mathtt{m}}$ plus $\mathtt{Ax}\exists^{+}$ and $\mathtt{Ax}\exists^{-}$ for the newly introduced existential variables x and for the existential variables Φ of those $\mathtt{AxCA}$ necessarily involved by the verifying proof only.

Remark 2.15 The difference between the above extraction by the light MD-interpretation and the sequence of first doing a Light Dialectica extraction followed by the majorizability construction of Proposition 1.78 only pops up when we further require that extracted terms should be in (NbE-)normal form and is mostly about run-time performance. It should be obvious that the normalization of $\widetilde{S}$ from (2.24) is much simpler than the one of $\widetilde{S}$ from (2.9) because the boolean terms are eliminated (hence also the n comparisons between them!). The complexity gain can get quite huge in concrete practical applications like the one described in Chapter 5 below (based on [55]). This big optimization is due to the partially repeated (during extraction) normalization of the extracted term, i.e., the so-called "Normalization during Extraction" (NdE), see [53] (also in combination with [55]). In the end one gets a correct (NbE-)normalized (pre-)majorant by either of the two methods[10], see Remark 1.79 for an explanation of this fact.

Convention 2.16 Only the arithmetic (pre-)majorants $t[a] \in \{T[a]\}$ are of interest, since boolean majorants can always be ensured by terms True (see Remark 1.76). In view of this we consider that all explicitly displayed types, variables and terms in the sequel are arithmetic, unless otherwise specified.

Corollary 2.17 There exists an algorithm which from a given proof

$$\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}} \quad \vdash \quad \forall u \exists v\, B(u, v, a)$$

produces (pre-)majorants $T[a]$ with a verifying proof

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}} \quad \vdash \quad \exists V\,[\,(\lambda a.\,T) \;\succeq\; V \;\wedge\; \forall a, u\, B^{\mathtt{D}}(u, V a u, a)\,] \tag{2.25}$$

If the input proof is in $\mathtt{WeZ}^{\mathtt{nc}+}_{\mathtt{m}}$, then the verifying proof is in $\mathtt{WeZ}_{\mathtt{m}}$ plus $\mathtt{Ax}\exists^{+}$ and $\mathtt{Ax}\exists^{-}$ for the newly introduced existential variables V and for the existential variables Φ of those $\mathtt{AxCA}$ necessarily involved by the verifying proof only.

[10] The two methods are the direct extraction of the normalized pre-majorant vs. the extraction of a normalized exact realizer followed by the majorizability construction of Proposition 1.78, followed by a NbE-normalization. Both extractions use the Normalization during Extraction. Even though the final result may be different, both methods produce correct pre-majorants in NbE-normal form.

Proof: Let $[B(u,v,a)]^{\mathtt{D}} \equiv \exists x \forall y\, B_{\mathtt{D}}(x;y;u,v,a)$. Then

$$[\forall u \exists v B(u,v,a)]^{\mathtt{D}} \;\equiv\; \exists V, X\, \forall u, y\; B_{\mathtt{D}}(Xu;y;u,Vu,a)$$

and the application of Corollary 2.14 yields

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}} \;\vdash\; \exists V, X\, [\, (\lambda a.\, T) \;\succeq\; V \;\wedge\; \forall a, u, y\, B_{\mathtt{D}}(Xau;y;u,Vau,a)\,]$$

hence a fortiori, by setting $x :\equiv Xau$ for given a, u, one obtains

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}} \;\vdash\; \exists V\, [\, (\lambda a.\, T) \;\succeq\; V \;\wedge\; \forall a, u\, \exists x\, \forall y\; B_{\mathtt{D}}(x;y;u,Vau,a)\,]$$

which is exactly a rewrite of (2.25). □

Proposition 2.18 (Equivalence of non-`ncm` fmlas with their `D`-interp.) For all $\mathcal{F}^{\exists}_{\mathtt{m}}$ formulas B there is a proof $\mathtt{WeZ}^{\exists,+}_{\mathtt{m}} \vdash B \leftrightarrow B^{\mathtt{D}}$, where $\mathtt{WeZ}^{\exists,+}_{\mathtt{m}}$ is the extension of $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ with the full (in the language $\mathcal{L}^{\exists}$) axiom of choice `AxAC`, and the comprehension axiom `AxCA` inherited from $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ is not actually used.

Proof: Straightforward by induction on the structure of B, also using that B contains no `ncm` quantifier. In consequence, the "radical" of its `LD`-translation, namely $B_{\mathtt{D}}$, is quantifier-free, hence decidable (stable). Thus one does not need to use the decidability of all $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ formulas as consequence of `AxCA`. The importance of the restriction on B actually lies in the fact that one cannot possibly prove the logical equivalence of a formula containing at least an `ncm` quantifier to its `LD`-translation which has no `ncm` quantifiers. E.g., a pure-`ncm` formula cannot be proved equivalent to its regular-quantifier translation. The logical equivalence can also not be proved in general between $B^{\mathtt{D}}$ and $\widetilde{B}$, where the latter formula is the regular-quantifier translation of B. This can be immediately (dis)proved by simple counterexamples involving formulas of shape $B \equiv \overline{\exists} z A(z)$ or $B \equiv \overline{\forall} z A(z)$, where $A \in \mathcal{F}^{\exists}_{\mathtt{m}}$ is such that $A^{\mathtt{D}}$ contains both universal and existential quantifiers.

The proof is obvious for quantifier-free B and immediate for conjunctions $A \wedge B$ and for existentially quantified $B \equiv \exists z A(z)$. For universally quantified $B \equiv \forall z A(z)$ one has to apply `AxAC`, but the proof is immediate as well.

For implications $A \to B$ we proceed step by step in showing the logical equivalence between $\exists x \forall y A_{\mathtt{D}}(x;y) \to \exists u \forall v B_{\mathtt{D}}(u;v)$ and $(A \to B)^{\mathtt{D}}$. The first step to $\forall x[\forall y A_{\mathtt{D}}(x;y) \to \exists u \forall v B_{\mathtt{D}}(u;v)]$ is purely intuitionistic. Then, by $\mathtt{AxIP}^{-\exists}_{+\forall}$ one obtains $\forall x \exists u[\forall y A_{\mathtt{D}}(x;y) \to \forall v B_{\mathtt{D}}(u;v)]$, from which the step to

$\forall x\exists u\forall v[\forall y A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)]$ is again purely intuitionistic. Now the step to $\forall x\exists u\forall v\exists y[A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)]$ requires an $\mathtt{AxMK}$. In fact the implication

$$\exists y[A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)] \;\to\; [\forall y A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)]$$

is purely intuitionistic and only its converse requires a passing through

$$\exists^{\mathtt{cl}} y[A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)] \qquad .$$

Hence the first half of the converse, namely

$$[\forall y A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)] \;\to\; \exists^{\mathtt{cl}} y[A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)]$$

is purely intuitionistic, its proof is similar to that of $\mathtt{Lm}\exists^{\mathtt{cl}}_{-}$. Hence we want a proof of $\bot$ from $[\forall y A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)]$ and $\forall y\neg(A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v))$. We use contraction over $\neg(A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v))$, from which both $\neg B_{\mathtt{D}}(u;v)$ and $\neg\neg A_{\mathtt{D}}(x;y)$ can be intuitionistically obtained. Since $A_{\mathtt{D}}(x;y)$ is quantifier-free, we can use its stability (see Lemma 1.36) to actually get $A_{\mathtt{D}}(x;y)$ and further $\forall y\, A_{\mathtt{D}}(x;y)$. Hence, using the first assumption we finally obtain $B_{\mathtt{D}}(u;v)$, which is in contradiction with $\neg B_{\mathtt{D}}(u;v)$. The second half of this converse, namely

$$\exists^{\mathtt{cl}} y[A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)] \;\to\; \exists y[A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)]$$

is just an instance of $\mathtt{AxMK}$, since both $A_{\mathtt{D}}(x;y)$ and $B_{\mathtt{D}}(u;v)$ are quantifier-free.

We now arrive at the last step of our equivalence simulation, namely the passing from $\forall x\exists u\forall v\exists y[A_{\mathtt{D}}(x;y) \to B_{\mathtt{D}}(u;v)]$, which is the "least non-intuitionistic"[11] prenex normal form of $\exists x\forall y A_{\mathtt{D}}(x;y) \to \exists u\forall v B_{\mathtt{D}}(u;v)$, to

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

As is now easy to guess, one here applies twice the Axiom of Choice $\mathtt{AxAC}$. □

Proposition 2.19 There exists a $\mathtt{WeZ}_{\mathtt{m}}$ functional term $\cdot^M$ of type $(\iota\iota)\iota\iota$ such that $\mathtt{WeZ}_{\mathtt{m}} \vdash \forall x^{\iota\iota}(x^M \succeq x)$.

Proof: The term $x^M :\equiv \lambda n.\, max(x\mathtt{0}, \ldots, xn)$ is definable in $\mathtt{WeZ}_{\mathtt{m}}$. □

Theorem 2.20 (Uniform bound synthesis by the LMD-interpretation) Let $A(x^{\iota\iota}, k^{\iota}, y^{\delta}, z^{\gamma})$ be a formula of $\mathcal{F}^{\exists}_{\mathtt{m}}$, hence without $\mathtt{ncm}$ quantifiers and

[11] See the "Motivation of the functional interpretation" section in the preamble of the "Dialectica" chapter in [63] for an explanation of this characterization.

moreover such that x, k, y, z are all its free variables, with $dg(\gamma) \leq 2$. Let $s^{(\iota\iota)\iota\delta}$ be a closed term of $\mathcal{T}_{\mathtt{m}}$. There exists an algorithm which from a given proof

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+} \ \vdash \ \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_{\delta} sxk\, \exists z^{\gamma}\, A(x,k,y,z) \tag{2.26}$$

which does not use any $\mathtt{AxSTAB}^{\mathtt{nc}}$ or $\mathtt{AxMK}$ in which the pure-$\mathtt{ncm}$ main formula is not $\mathtt{ncm}$-stable, produces the closed term $\mathtt{t}$ of $\mathcal{T}_{\mathtt{m}}$ such that

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,+} \ \vdash \ \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_{\delta} sxk\, \exists z \leq_{\gamma} \mathtt{t}xk\, A(x,k,y,z) \tag{2.27}$$

Hence $\mathtt{t}$ is a bound for z which is uniform w.r.t. y. If the formula A is such that $\mathtt{WeZ}_{\mathtt{m}}^{\exists} \vdash A^{\mathtt{D}} \to A$ then the verifying proof (2.27) is in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ instead of $\mathtt{WeZ}_{\mathtt{m}}^{\exists,+}$. If $A \in \mathcal{F}_{\mathtt{m}}^{\exists,\mathtt{nc}}$ is s.t. $\mathtt{WeZ}_{\mathtt{m}}^{\exists,+} \vdash A^{\mathtt{D}} \to \widetilde{A}$ (or $\mathtt{WeZ}_{\mathtt{m}}^{\exists} \vdash A^{\mathtt{D}} \to \widetilde{A}$), where $\widetilde{A}$ is the regular-quantifier translation of A, then (2.27) holds (in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$) with $\widetilde{A}$ instead of A.

Proof: Let $\delta \equiv \underline{\sigma}\iota$. Then (2.26) rewrites as

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+} \ \vdash \ \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y\, [\, \forall w^{\underline{\sigma}}\, (yw \leq_{\iota} sxkw) \ \to \ \exists z^{\gamma}\, A(x,k,y,z)\,]$$

and using $\mathtt{AxIP}_{+\forall}^{-\exists}$ this becomes

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+} \ \vdash \ \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y\, \exists z^{\gamma}\, [\, \forall w^{\underline{\sigma}}\, (yw \leq_{\iota} sxkw) \ \to \ A(x,k,y,z)\,]$$

to which we apply Corollary 2.17 and obtain effectively closed terms T s.t.

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists} \ \vdash \ \exists Z\, [\, T \ \succeq \ Z \ \wedge \ \forall x,k,y\, (y \leq_{\delta} sxk \to A(x,k,y,Zxky))^{\mathtt{D}}\,] \tag{2.28}$$

To (2.28) we can apply Proposition 2.18 and thus obtain

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,+} \ \vdash \ \exists Z\, [\, T \ \succeq \ Z \ \wedge \ \forall x^{\iota\iota} \forall k^{\iota} \forall y \leq_{\delta} sxk\ A(x,k,y,Zxky)\,] \tag{2.29}$$

Let $s^{\star}$ be the closed term of $\mathtt{WeZ}_{\mathtt{m}}$ associated to s by means of Proposition 1.78 and let $x^{\iota\iota}, k^{\iota}$. Since $\mathtt{WeZ}_{\mathtt{m}} \vdash s^{\star} \ \succeq \ s$ and, by Proposition 2.19, $\mathtt{WeZ}_{\mathtt{m}} \vdash x^{M} \ \succeq \ x$ it follows by the definition of $\succeq$ that $\mathtt{WeZ}_{\mathtt{m}} \vdash s^{\star}x^{M}k \ \succeq \ sxk$. Thus, for $y \leq_{\delta} sxk$ it follows from (1.42) that $s^{\star}x^{M}k \ \succeq_{\delta} \ y$. Since $T \ \succeq \ Z$, by the definition of $\succeq$ we obtain that $Tx^{M}k(s^{\star}x^{M}k) \ \succeq_{\tau} \ Zxky$. If $\tau \equiv \iota$ then $Zxky \leq_{\tau} Tx^{M}k(s^{\star}x^{M}k)$ and we take $\mathtt{t} :\equiv \lambda x,k.\, Tx^{M}k(s^{\star}x^{M}k)$. If $dg(\tau) = 1,$, hence $\tau \equiv \iota \ldots \iota$ for v^{ι} we have $v \ \succeq_{\iota} \ v$ and thus we obtain $Zxky\underline{v} \leq_{\iota} Tx^{M}k(s^{\star}x^{M}k)\underline{v}$ which rewrites as $Zxky \leq_{\tau} Tx^{M}k(s^{\star}x^{M}k)$. Again, we can take $\mathtt{t} :\equiv \lambda x,k.\, Tx^{M}k(s^{\star}x^{M}k)$. The situation is slightly different when $dg(\tau) = 2$, hence $\tau \equiv \ldots(\iota\iota)\ldots\iota$. By Proposition 2.19 we have $Zxky\underline{v} \leq_{\iota} Tx^{M}k(s^{\star}x^{M}k)\underline{v^{M}}$ and in this case we can take

$$\mathtt{t} \ :\equiv \ \lambda x,k,\underline{v}.\, Tx^{M}k(s^{\star}x^{M}k)\underline{v^{M}} \qquad .$$

In all three cases determined by $dg(\tau)$, from (2.29) we obtain

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,+} \;\vdash\; \exists Z\,\forall x^{\iota\iota}\forall k^{\iota}\forall y \leq_{\delta} s\,x\,k\;[\,Zxky \leq_{\tau} \mathtt{t}xk \wedge A(x,k,y,Zxky)]$$

from which the conclusion (2.27) follows immediately. □

Definition 2.21 We denote by `smaj` Bezem's [16] *strong* majorization variant of Howard's [58] $\succeq$, defined by replacing (1.38) with

$$x\;\mathtt{smaj}_{\sigma\tau}\,y \;:\equiv\; \forall z_1^{\sigma}, z_2^{\sigma}\;(z_1\;\mathtt{smaj}_{\sigma}\,z_2 \;\rightarrow\; xz_1\;\mathtt{smaj}_{\tau}\,xz_2, yz_2) \quad (2.30)$$

Remark 2.22 In fact Kohlenbach's original proofs from [68] of the results from which our theorems and corollaries above are inspired were given with `smaj` instead of $\succeq$, but in [63] the same proofs are re-taken using $\succeq$. See Remark 2.4 of [68] for a comment on the trade-offs in terms of style of proofs (on the meta level) when using one or the other of the two variants of the majorizability relation. Nonetheless, for the results exposed in our thesis both variants can be used equally, as stated by the following theorem.

Theorem 2.23 All theorems and corollaries above hold as well if `smaj` is used instead of $\succeq$.

Notice that the optimization brought by the light `MD`-interpretation w.r.t. the pure `D`-interpretation concerns as much the diminishing of the maximal type degree of the extracted bounds as the elimination of a number of redundant contractions. The computation of maxima can turn out to be a rather costly operation in practice, even though the pure `MD`-interpretation already brings an important optimization w.r.t. the pure Dialectica Interpretation.

2.3 Extensions of the `L(M)D`-interpretation to extractions from fully classical proofs

The problem of full `AxSTAB` is that it has no (direct) realizer under (light) Dialectica interpretation. Preprocessing double negation translations will be necessary to interpret fully classical systems. There is a large choice in the literature of so-called *negative* or *double-negation* translations, initially introduced by Gödel [44], Gentzen, Kolmogorov, Glivenko. However, they all have the common feature that the image of a formula is intuitionistically equivalent to a negative formula. This immediately implies the elimination of Stability in

the translated proof. Thus, the translation of formulas A induces a translation from classical proofs of A to intuitionistic proofs of the image of A.

Kuroda's $\mathtt{N}$-translation (abbreviated "the $\mathtt{KN}$-translation") appears as a natural choice in the context of Dialectica interpretations because of its ability to uniformly handle blocks of universal quantifiers (see, e.g., [57, 63, 81]). We will also use the Gödel-Gentzen negative translation (abbreviated "the $\mathtt{GN}$-translation", for important historical references see [123] and [90]) which simply replaces $\exists$ by $\exists^{\mathtt{cl}}$ (and also $\overline{\exists}$ by $\overline{\exists^{\mathtt{cl}}}$ in our $\mathtt{ncm}$-setting). The usual double negation of every atomic formula would here be redundant because our systems enjoy quantifier-free stability, see Lemma 1.36. Hence the $\mathtt{GN}$-translation appears to be structurally much simpler than the $\mathtt{KN}$-translation and therefore appears to be most suitable in a Natural Deduction context (see [111] as an example, also for an important optimization relative to the program extraction by the $\mathtt{BBS}$ refined $\mathtt{A}$-translation). Also the strong $\exists$ (and similarly $\overline{\exists}$ in our $\mathtt{ncm}$-setting) disappears from the translated system which therefore seems to be simpler than the $\mathtt{KN}$-translated system (which conserves the strong existentials).

But surprisingly, we quickly find out that none of these negative translations extends well to the non-monotonic $\mathtt{ncm}$ systems with strong existentials. For the $\mathtt{KN}$-translation this is mainly because the interpretations of the elimination axioms $\mathtt{Ax}\exists^{-}$ and $\mathtt{Ax}\overline{\exists}^{-}$ cannot be proved just in $\mathtt{WeZ}^{\mathtt{nc}+}$ for all instances of such axioms with formulas of $\mathcal{F}^{\mathtt{nc}}$. The reason is that, e.g., the proof of $[\mathtt{Ax}\exists^{-}]^{\mathtt{KN}}$ makes a necessary use of contraction over

$$\neg[\exists z_1 A(z_1) \wedge \forall z_2 \neg\neg(A(z_2) \to B) \to B]$$

which would clearly violate the $\mathtt{ncm}-\mathtt{FC}$ restriction if A or B would contain just an $\mathtt{ncm}$ quantifier. In the case of the $\mathtt{GN}$-translation, the interpretation of $\mathtt{Ax}\exists^{-}$ appears to not be provable without full stability for B in $\mathtt{WeZ}^{\mathtt{nc}+}$, see Lemma 1.43. But this full stability is generally not provable in our $\mathtt{ncm}$-system $\mathtt{WeZ}^{\mathtt{nc}+}$, (even though it is provable in its monotonic counterpart $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$), see Remark 1.42 and Lemma 1.40.

Hence we choose not to design any (non-monotonic) classical variant of $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$. In the lack of existential quantifiers, we can nevertheless still design a Kuroda translation for the system $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$, defined in Section 1.6. Some further restrictions on the fully classical proof rules of $\to^{+}$ and $\overline{\forall}^{+}$ were necessary in order to ensure that the $\mathtt{KN}$-translated proofs fit into $\mathtt{WeZ}^{\mathtt{nc}+}$, see the proof of Theorem 2.27 below. This appears as a quite hard price to pay in order to use stability for formulas containing $\overline{\forall}$ in the input systems. Moreover, the

addition of such stability does not ensure that the extension of all principles provable in $\mathtt{WeZ}$ to the language with $\overline{\forall}$ would become provable in such $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ system. The classical Independence of premises $\mathtt{AxIP}^{\mathtt{cl}}$ is a counter-example to this, see Sections 1.4 and 1.6. Despite all the above, we chose to design $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ as a pedagogical example of the difficulties incurred by such an attempt which thus (we think) do justify the worthlessness of going further and adding more restrictions in order to also have the strong existentials.

On the other hand no technical problem appears in the application of our $\mathtt{ncm}$ negative translations to the monotonic systems, which have no restriction on $\rightarrow^+$ and only have a simpler, harmless restriction on $\overline{\forall}^+$.

Definition 2.24 (Kuroda's negative translation adapted to $\mathtt{ncm}$ qfs.) To a formula A one associates $A^{\mathtt{KN}} \equiv \neg\neg A^*$, where A^* is defined by structural induction on A as follows:

- $A^* :\equiv A$, if A is a pure-$\mathtt{ncm}$ formula (prime formulas included)
- $[A \Box B]^* :\equiv A^* \Box B^*$, where $\Box \in \{\wedge, \vee, \rightarrow\}$
- $[\exists x A(x)]^* :\equiv \exists x [A(x)]^*$ and $[\overline{\exists} x A(x)]^* :\equiv \overline{\exists} x [A(x)]^*$
- $[\forall x A(x)]^* :\equiv \forall x \neg\neg [A(x)]^*$ and $[\overline{\forall} x A(x)]^* :\equiv \overline{\forall} x \neg\neg [A(x)]^*$

Remark 2.25 (Preservation of the $\mathtt{ncm}-\mathtt{FC}$ cnd. under $\mathtt{KN}$-translation) Since the insertion of double negations $\neg\neg$ into a formula A does not change the polarity of the (original) positive/negative positions for quantifiers, it follows that the restriction $\mathtt{ncm}-\mathtt{FC}(A)$ holds if and only if $\mathtt{ncm}-\mathtt{FC}(A^{\mathtt{KN}})$ holds.

Lemma 2.26 If A is a formula of shape $\forall \underline{x} B^{\mathtt{nc}}(\underline{x}, a)$ or $\forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x, y, a)$ then $\mathtt{WeZ}^{\mathtt{nc}+} \vdash A^{\mathtt{KN}} \leftrightarrow A$. If $A \equiv \forall x \exists y B^{\mathtt{nc}}(x, y, a)$ then $\mathtt{WeZ}^{\exists,\mathtt{nc}+} \vdash A^{\mathtt{KN}} \leftrightarrow A$.

Proof: Let $A :\equiv \forall \underline{x} B^{\mathtt{nc}}(\underline{x})$ be such a quasi - purely universal formula (for simplicity we can ignore its free variables). By the definition of $\mathtt{KN}$-translation, then multiple (possibly none) use of (1.27) and finally $\mathtt{AxSTAB}^{\mathtt{nc}}$:

$$\mathtt{WeZ}^{\mathtt{nc}+} \vdash [\forall \underline{x} B^{\mathtt{nc}}(\underline{x})]^{\mathtt{KN}} \leftrightarrow \neg\neg \forall \underline{x} \neg\neg B^{\mathtt{nc}}(\underline{x}) \overset{(1.27)}{\leftrightarrow} \forall \underline{x} \neg\neg B^{\mathtt{nc}}(\underline{x}) \leftrightarrow \forall \underline{x} B^{\mathtt{nc}}(\underline{x})$$

Now let $A :\equiv \forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x, y, a)$. The wanted conclusion follows in this case by (1.27) and the easy $\mathtt{WeZ}^{\mathtt{nc}}$ theorem $\neg\neg\neg A \leftrightarrow \neg A$. Similarly, but in this case using $\mathtt{AxMK}$ instead of pure-$\mathtt{ncm}$ Stability, formulas $\forall x \exists y B^{\mathtt{nc}}(x, y)$ are also preserved by the $\mathtt{KN}$-translation:

$$\mathtt{WeZ}^{\exists,\mathtt{nc}+} \vdash [\forall x \exists y B^{\mathtt{nc}}(x,y)]^{\mathtt{KN}} \leftrightarrow \neg\neg\forall x \neg\neg\exists y B^{\mathtt{nc}}(x,y) \overset{(1.27)}{\leftrightarrow} \forall x \neg\neg\exists y B^{\mathtt{nc}}(x,y) \leftrightarrow \forall x \exists y B^{\mathtt{nc}}(x,y)$$

□

Recall from Section 1.6 that in the classical non-monotonic ncm-system $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$, the ncm-restriction on Implication introduction $\rightarrow^+$ is extended to (the negation of) the conclusion formula as well. Related to this, also the variable condition $\mathtt{VC}_3(\cdot)$ set on ncm-ForAll introduction $\overline{\forall}^+$ is adapted to this reenforcement of the restriction on $\rightarrow^+$.

Theorem 2.27 (Soundness of KN-translation - non-monotonic case) Let Π be the arbitrary but fixed set of quasi-purely-universal axioms included in $\mathtt{WeZ}^{\mathtt{nc}}$. There exists an algorithm which transforms a given input proof $\mathcal{P} : \{C_i\}_{i=1}^n \vdash A$ in $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ into a proof $\mathcal{P}^{\mathtt{KN}} : \{C_i^{\mathtt{KN}}\}_{i=1}^n \vdash A^{\mathtt{KN}}$ in $\mathtt{WeZ}^{\mathtt{nc}+}$ such that Π is preserved by the KN-translation .

Proof: The algorithm proceeds by recursion on the input proof $\mathcal{P}$. The translated proof $\mathcal{P}^{\mathtt{KN}}$ mimics the structure of $\mathcal{P}$, whose leaves (assumptions and axiom instances) and root (proof conclusion) are replaced by their $\cdot^{\mathtt{KN}}$ translations. We first establish that the KN-translations of the input axioms are provable theorems of the verifying systems. We then demonstrate that the KN-translations of the rules can be simulated in the verifying systems as the deductions of the KN-translation of the conclusion from the KN-translations of the premises. Whereas for the non-ncm subsystems such translations had already been established and actively used in concrete proof mining (see, e.g., [57, 63, 81]), we need to put some special attention to the situations in which the ncm quantifiers are involved either directly or indirectly.

Whereas Remark 2.25 ensures that the translation proceeds smoothly relative to the original computationally LD-relevant contractions, hence the KN-translations of $\rightarrow^+$ and $\overline{\forall}^+$ are sound in this respect, we need to trace the occurrence of new such computationally relevant contractions, which may be introduced by the verifying simulations. We will see that only the KN-translation of $\rightarrow^+$ introduces a new contraction which involves the (negation of the) conclusion formula of $\rightarrow^+$. This will also impact on the verification of the variable condition $\mathtt{VC}_3(\cdot)$ set on $\overline{\forall}^+$. Nonetheless, our restrictions set on the underlying $\rightarrow^+$ and $\overline{\forall}^+$ of the ncm-classical arithmetic $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ will ensure the correctness of our simulations of $[\rightarrow^+]^{\mathtt{KN}}$ and $[\overline{\forall}^+]^{\mathtt{KN}}$. We now proceed with the enumeration of the KN-translations of axioms and rules.

The Stability axiom $\neg\neg A \to A$ simply $\mathtt{KN}$-translates to $\neg\neg\neg\neg A^* \to \neg\neg A^*$ which has an immediate proof. Hence $\mathtt{AxSTAB}$ is immediately eliminated in the verifying proof. From Lemma 2.26 follows that all quantifier-free and purely universal axioms and rules (axioms: $\mathtt{AxTRH}$, $\mathtt{AxEFQ}$, $\mathtt{REF}$, $\mathtt{AxEQL}$ and rules: $\mathtt{SYM}$, $\mathtt{TRZ}$, $\mathtt{SUB}$) together with the Π axioms of the input systems are simply preserved by the $\mathtt{KN}$-translation. In roughly the same category enters the weak Compatibility rule $\mathtt{CMP}$, whose premise involves a quantifier-free and a purely universal formula and therefore its $\mathtt{KN}$-translation is equivalent to an(other) instance of the very same $\mathtt{CMP}$. The $\mathtt{KN}$-translation of $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ can more directly be proved to be equivalent in $\mathtt{WeZ}^{\mathtt{nc}+}$ to $\neg\neg[\forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x,y) \to \exists^{\mathtt{cl}} Y \forall x \neg\neg B^{\mathtt{nc}}(x, Yx)]$, which can be established in $\mathtt{WeZ}^{\mathtt{nc}+}$ by $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$, $\mathtt{AxSTAB}^{\mathtt{nc}}$ and (1.25). For such equivalence proof one uses that, due to (1.27), it can be established that $[\exists^{\mathtt{cl}} x A(x)]^* \leftrightarrow \exists^{\mathtt{cl}} x [A(x)]^*$ and also that $\neg\neg\exists^{\mathtt{cl}} x A(x) \leftrightarrow \exists^{\mathtt{cl}} x A(x)$. The $\mathtt{KN}$-translation of $\mathtt{AxBIA}$ is $\neg\neg([A(\mathtt{tt})]^* \wedge [A(\mathtt{ff})]^* \to \forall p^o \neg\neg [A(p)]^*)$, which can be proved with (1.25) and an(other) instance of $\mathtt{AxBIA}$. The $\mathtt{KN}$-translation of $\mathtt{AxFLS}$ is the double negation of an(other) instance of $\mathtt{AxFLS}$.

We now treat the induction and logical rules. The $\mathtt{KN}$-translation of $A \vdash A$ is $A^{\mathtt{KN}} \vdash A^{\mathtt{KN}}$, hence an(other) instance of the same rule. The $\mathtt{KN}$-translations of $\wedge^+$, $\wedge^-_l$ and $\wedge^-_r$ follow by simple Natural Deduction proofs from the instances of the same rules for A^* and B^*. No contraction is needed in the simulation of $[\wedge^-_l]^{\mathtt{KN}}$ or $[\wedge^-_r]^{\mathtt{KN}}$, since the necessary contraction from the proof of (1.34) intervenes only at the very end, when the two double negated conjuncts are joined together, step which is not present in these $\mathtt{KN}$-verifying proofs. The $\mathtt{KN}$-translation of $\to^-$ can also immediately be simulated without any use of extra contractions, see also the proof of (1.32), where this would be one of the "easy" implications. Similar to the treatment of $\to^-$, but here also using (1.27) (both implications!) is the simulation of the $\mathtt{KN}$-translation of the Induction rule $\mathtt{IR}_{\mathrm{o}}$, which thus uses the instance of $\mathtt{IR}_{\mathrm{o}}$ for $\neg\neg A^*(z) \equiv [A(z)]^{\mathtt{KN}}$. Here as well, no "alien" contraction is introduced in the verifying proof. Similarly, again using (1.27), the verification for $[\forall^-]^{\mathtt{KN}}$ and $[\forall^+]^{\mathtt{KN}}$ is immediate. We would be tempted to state the same for the verification of $[\overline{\forall}^{\,-}]^{\mathtt{KN}}$ and $[\overline{\forall}^{\,+}]^{\mathtt{KN}}$, this time by means of (1.29). Nonetheless we here must distinguish some more cases. At both $\overline{\forall}^{\,-}$ and $\overline{\forall}^{\,+}$ it may happen that A is a pure-$\mathtt{ncm}$ formula, situation when $\mathtt{AxSTAB}^{\mathtt{nc}}$ must be used in the verifying proofs (also (1.25)). For the soundness of $\overline{\forall}^{\,+}$ we also must comment that the $\mathtt{VC}_3(\cdot)$ is still satisfied because, on one hand, no new terms t get introduced via $\forall^-_{\bullet,t}$ in our $\mathtt{KN}$-verifying proofs. On the other hand, the case of the new computationally $\mathtt{LD}$-relevant contractions,

which involve the conclusion formulas of $\rightarrow^+$ was already considered in the extra restriction set on $\overline{\forall}^+$ in the $\mathtt{ncm}$-classical systems.

In the very end, we outline the $\mathtt{KN}$-translation of $\rightarrow^+$. We are given a deduction of the translated conclusion $B^{\mathtt{KN}} \equiv \neg\neg B^*$ from the translated undischarged assumptions, among which $A^{\mathtt{KN}} \equiv \neg\neg A^*$. From this we have to produce a proof of $(A \rightarrow B)^{\mathtt{KN}} \equiv \neg\neg(A^* \rightarrow B^*)$ in which the assumptions $A^{\mathtt{KN}}$ were cancelled. It is very easy to obtain a proof of $(\neg\neg A^* \rightarrow \neg\neg B^*)$ by an Implication introduction of $\neg\neg A^*$. But the obtaining of $(A \rightarrow B)^{\mathtt{KN}}$ appears to be impossible without an extra contraction over $\neg(A^* \rightarrow B^*)$. From such an assumption one can immediately get on one hand $\neg\neg A^*$, and thus also $\neg\neg B^*$, by a Modus Ponens and on the other hand $\neg B^*$. Now $\bot$ is obtained by Modus Ponens and, via a $\rightarrow^+$ with contraction over $\neg(A^* \rightarrow B^*)$, one finally obtains $(A \rightarrow B)^{\mathtt{KN}}$. This final extra contraction can take place because $\mathtt{ncm}\!-\!\mathtt{FC}(\neg(A^* \rightarrow B^*))$ holds due to the restrictions set on the $\mathtt{ncm}$-classical system at input. $\square$

Corollary 2.28 In the case when $A \equiv \forall x B^{\mathtt{nc}}(x,a)$ or $A \equiv \forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x,y,a)$, in Theorem 2.27 we can replace $A^{\mathtt{KN}}$ by A in the conclusion proof.

Proof: Follows immediately from Theorem 2.27 by using Lemma 2.26. $\square$

Theorem 2.29 (Exact realizer synthesis by the $\mathtt{KNLD}$-interpretation) There exists an algorithm which from a given input proof $\vdash A(a)$ in $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ produces exact realizing terms $T[a]$ together with a verifying proof in system $\mathtt{WeZ}$: $\vdash \forall a, y (A^{\mathtt{N}})_{\mathtt{D}}(T; y; a)$.

Proof: Immediate from Theorem 2.27 followed by Corollary 2.11. $\square$

Corollary 2.30 There exists an algorithm which from a given input proof $\vdash \forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x,y,a)$ in $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$ produces closed realizing terms T with a verifying proof $\vdash \forall a, x B(x, Tax, a)$ in $\mathtt{WeZ}$. The same holds also with $B^{\mathtt{nc}}$ replaced by a formula $\exists^{\mathtt{cl}} y' B^{\mathtt{nc}}(x,y,y',a)$.

Proof: Immediate from Corollary 2.28 followed by Corollary 2.11 and then expanding $[\forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x,y,a)]_{\mathtt{D}}$ according to Definition 2.1. One also uses the fact that full stability is available in the verifying system $\mathtt{WeZ}$, see Lemma 1.40. The variant with $\exists^{\mathtt{cl}} y' B^{\mathtt{nc}}(x,y,y',a)$ follows from the original version by ignoring the actual realizers for the weak existentially quantified variables y'. $\square$

We now proceed to the treatment of the monotonic classical $\mathtt{ncm}$-system $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$. Recall that all monotonic systems are without any restriction on

$\rightarrow^+$ and with only a half-restriction on $\overline{\forall}^+$, which shows to be harmless to our adaptation of negative translations to the ncm systems.

Theorem 2.31 (Soundness of KN-translation - the monotonic case) Let Π be the usual arbitrary but fixed set of quasi-purely-universal axioms. Also let Δ be the axiom set of Definition 1.67 and Remark 1.80. Both Π and Δ are included in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$. There exists an algorithm which transforms a given proof $\mathcal{P} : \{C_i\}_{i=1}^n \vdash A$ in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ into a proof $\mathcal{P}^{\mathtt{KN}} : \{C_i^{\mathtt{KN}}\}_{i=1}^n \vdash A^{\mathtt{KN}}$ in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+}$ such that Π is preserved by the KN-translation and $\Delta^{\mathtt{KN}}$ is proved in terms of Δ (i.e., $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+} \vdash A \rightarrow A^{\mathtt{KN}}$ for every formula $A \in \Delta$).

Proof: Similar to the proof of Theorem 2.27 above, with a simplified treatment of $[\rightarrow^+]^{\mathtt{KN}}$ and $[\overline{\forall}^+]^{\mathtt{KN}}$. We will here only give the treatment of strong existential axioms and $\mathtt{AxAC}_{\mathtt{nc}}$, which are the only principles not present in $\mathtt{WeZ}^{\mathtt{nc},\mathtt{c}+}$, because of the strong $\exists$. We begin with $[\mathtt{AxAC}_{\mathtt{nc}}]^{\mathtt{KN}}$. By Lemma 2.26, the premise of $\mathtt{AxAC}_{\mathtt{nc}}$ is preserved. The KN-translation of $\mathtt{AxAC}_{\mathtt{nc}}$ is thus equivalent in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ to $\neg\neg[\forall x \exists y B^{\mathtt{nc}}(x,y) \rightarrow \exists Y \forall x \neg\neg B^{\mathtt{nc}}(x, Yx)]$, which can be proved by $\mathtt{AxAC}_{\mathtt{nc}}$, pure-ncm Stability $\mathtt{AxSTAB}^{\mathtt{nc}}$ and (1.25).

We now treat the strong existential axioms. The KN-translation of $\mathtt{Ax}\exists^+$ is $\neg\neg\forall z_1 \neg\neg[A^*(z_1) \rightarrow \exists z_2 A^*(z_2)]$ which has a simple logical proof from the instance with A^* of $\mathtt{Ax}\exists^+$. The situation is identical for $[\mathtt{Ax}\overline{\exists}^+]^{\mathtt{KN}}$ if A is not purely-ncm, case which is even simpler, more direct. The KN-translation of $\mathtt{Ax}\exists^-$ is $\neg\neg[\exists z_1 A^*(z_1) \wedge \forall z_2 \neg\neg(A^*(z_2) \rightarrow B^*) \rightarrow B^*]$, which can be reduced to $\exists z_1 A^*(z_1) \wedge \forall z_2 \neg\neg(A^*(z_2) \rightarrow B^*) \rightarrow \neg\neg B^*$ via (1.32), since no $\mathtt{ncm}-\mathtt{FC}$ restrictions apply to the monotone systems. Again, via the simpler implication of (1.32), this reduces further to $\exists z_1 A^*(z_1) \wedge \forall z_2 (A^*(z_2) \rightarrow \neg\neg B^*) \rightarrow \neg\neg B^*$, which is another instance of $\mathtt{Ax}\exists^-$, with A^* for A and $\neg\neg B^*$ for B. The situation is similar for $[\mathtt{Ax}\overline{\exists}^-]^{\mathtt{KN}}$, and even simpler in its pure-ncm sub-case.

In the end, we prove the statement concerning the translation of axioms Δ. Let $A \equiv \forall x^\rho \, \exists y \leq_\sigma rx \, \forall z^\tau \; B^{\mathtt{nc}}(x,y,z)$ be such an axiom formula.Then

$$A^{\mathtt{KN}} \;\equiv\; \neg\neg\, \forall x^\rho \, \neg\neg \; \exists y \leq_\sigma rx \; \forall z^\tau \; \neg\neg \, B^{\mathtt{nc}}(x,y,z)$$

which is equivalent via $\mathtt{AxSTAB}^{\mathtt{nc}}$ and (1.27) to the simpler-looking formula $\forall x^\rho \, \neg\neg \; \exists y \leq_\sigma rx \; \forall z^\tau \; B^{\mathtt{nc}}(x,y,z)$, which is immediately seen to be provable from A via (1.25) and simple logical manipulations involving $\forall^-$ and $\forall^+$. $\square$

Corollary 2.32 In the case when $A \equiv \forall x \exists y B^{\mathtt{nc}}(x,y,a)$, in Theorem 2.31 we can replace $A^{\mathtt{KN}}$ by A in the conclusion proof.

Proof: Follows immediately from Theorem 2.31 by using Lemma 2.26. □

Theorem 2.33 (Main theorem on bound extraction by KNLMD-intrp.) There exists an algorithm which from a given input proof $\vdash A(a)$ in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ produces at output (pre-)majorizing terms $T[a]$ together with the verifying proof $\vdash \exists x[(\lambda a.\, T) \ \succeq \ x \wedge \forall a, y(A^{\mathtt{N}})_{\mathtt{D}}(x(a); y; a)]$ in system $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$. The same hold if Bezem's `smaj` is used instead of Howard's $\succeq$, see (2.30).

Proof: Immediate from Theorem 2.31 followed by Corollary 2.14. Theorem 2.23 is used for the `smaj` variant. □

Corollary 2.34 There exists an algorithm which from a given input proof $\vdash \forall x \exists y \exists z B^{\mathtt{nc}}(x, y, z)$ in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ produces closed realizing terms T together with the following verifying proof in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$:

$$\vdash \ \exists Y\, [T \ \succeq \ Y \wedge \forall x \exists z B(x, Yx, z)] \tag{2.31}$$

Proof: By applying Corollary 2.32 followed by Corollary 2.14 and expanding $[\forall x \exists y, z B^{\mathtt{nc}}(x, y, z)]_{\mathtt{D}}$ by Definition 2.1 we obtain closed terms T and T' s.t.

$$\vdash \ \exists Y \exists Z\, [T \ \succeq \ Y \wedge T' \ \succeq \ Z \wedge \forall x B(x, Yx, Zx)]$$

from which (2.31) follows by ignoring T' and simple logical manipulations. □

Theorem 2.35 (Uniform bound synthesis by KNLMD-interpretation) Let $A_1(x^{\iota\iota}, k^{\iota}, y^{\delta}, z^{\gamma})$ be a quasi-purely-existential formula with x, k, y, z all its free variables (i.e., $A_1 \equiv \exists u A^{\mathtt{nc}}(x, k, y, z, u)$) and $dg(\gamma) \leq 2$. Let $s^{(\iota\iota)\iota\delta}$ be a closed term of $\mathcal{T}_{\mathtt{m}}$. There exists an algorithm which from a given proof

$$\vdash \ \forall x^{\iota\iota} \ \forall k^{\iota} \ \forall y \leq_{\delta} sxk \ \exists z^{\gamma} \ A_1(x, k, y, z) \tag{2.32}$$

in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ produces the closed term $\mathtt{t}^{(\iota\iota)\iota\gamma}$ of $\mathcal{T}_{\mathtt{m}}$ together with the following verifying proof in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$:

$$\vdash \ \forall x^{\iota\iota} \ \forall k^{\iota} \ \forall y \leq_{\delta} sxk \ \exists z \leq_{\gamma} \mathtt{t}xk \ \widetilde{A_1}(x, k, y, z) \tag{2.33}$$

where $\widetilde{A_1}(x, k, y, z) \equiv \exists u A(x, k, y, z, u)$, i.e., the regular-quantifier translation of A_1. Hence $\mathtt{t}$ is a bound for z which is uniform relative to y.

Proof: Let $\delta \equiv \underline{\sigma}\iota$. Then (2.32) immediately yields in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ a proof

$$\vdash \ \forall x,k,y\exists w^{\underline{\sigma}},z,u^{\underline{\tau}}[yw \leq_\iota sxkw \to A^{\mathtt{nc}}(x,k,y,z,u^{\underline{\tau}})] \qquad (2.34)$$

The application of Corollary 2.34 to (2.34) gives closed terms T such that

$$\vdash \ \exists Z[T \succeq Z \wedge \forall x,k,y\exists w,u(yw \leq_\iota sxkw \to A(x,k,y,Zxky,u))]$$

is a proof in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$, which immediately yields in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$

$$\vdash \ \exists Z[T \succeq Z \wedge \forall x,k\forall y \leq_\delta sxk\,\widetilde{A_1}(x,k,y,Zxky)] \qquad (2.35)$$

By logical manipulations inside $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$, expanding the definition of majorizability $\succeq$, we obtain the following proof in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$:

$$\vdash \ \exists Z\forall x,k\forall y \leq_\delta sxk\,[Tx^Mk(s^\star x^Mk) \ \succeq_\gamma \ Zxky \wedge \widetilde{A_1}(x,k,y,Zxky)] \quad ,$$

where $s^\star$ is the majorant of s built by Proposition 1.78 and x^M is the majorant of $x^{\iota\iota}$ given by Proposition 2.19. This gives

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists} \vdash \ \forall x,k\forall y \leq_\delta sxk\exists z^\gamma\,[Tx^Mk(s^\star x^Mk) \ \succeq_\gamma \ z \wedge \widetilde{A_1}(x,k,y,z)] \qquad (2.36)$$

by very simple logical manipulations inside $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$. If $dg(\gamma) \leq 1$ then

$$Tx^Mk(s^\star x^Mk) \geq_\gamma z$$

and therefore by defining the term $\mathtt{t}$ to be $\lambda x,k.\,Tx^Mk(s^\star x^Mk)$, (2.36) immediately implies the conclusion (2.33). If $dg(\gamma) = 2$ then just like in the proof of Theorem 2.20 we can set $\mathtt{t}$ to be $\lambda x,k,\underline{v}.\,Tx^Mk(s^\star x^Mk)\underline{v^M}$ for which again, using $v^M \ \succeq_{\iota\iota} \ v$, (2.36) immediately implies the conclusion (2.33). $\square$

In the end we give the treatment of the $\mathtt{GN}$-translation, which is worth to be designed for the monotonic system $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ only, due to its simpler definition of $\to^+$ and $\overline{\forall}^+$, see the comments in the preamble of this section.

Definition 2.36 (The $\mathtt{ncm}$-Gödel-Gentzen negative translation) To a formula A one associates $A^{\mathtt{GN}}$, the so-called $\mathtt{GN}$-translation of A, which is defined by structural induction on A as follows:

- $A^{\mathtt{GN}} :\equiv A$, if A is a quantifier-free formula (prime formulas included)
- $[A\Box B]^{\mathtt{GN}} :\equiv A^{\mathtt{GN}}\Box B^{\mathtt{GN}}$, where $\Box \in \{\wedge,\to\}$

- $[\exists x A(x)]^{\mathtt{GN}} :\equiv \exists^{\mathtt{cl}} x [A(x)]^{\mathtt{GN}}$ and $[\overline{\exists} x A(x)]^{\mathtt{GN}} :\equiv \overline{\exists^{\mathtt{cl}}} x [A(x)]^{\mathtt{GN}}$
- $[\forall x A(x)]^{\mathtt{GN}} :\equiv \forall x [A(x)]^{\mathtt{GN}}$ and $[\overline{\forall} x A(x)]^{\mathtt{GN}} :\equiv \overline{\forall} x [A(x)]^{\mathtt{GN}}$

Theorem 2.37 (Soundness of GN-translation - the monotonic case) Let Π be the usual arbitrary but fixed set of quasi-purely-universal axioms. There exists an algorithm which transforms a given proof $\mathcal{P} : \{C_i\}_{i=1}^n \vdash A$ in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ into a proof $\mathcal{P}^{\mathtt{GN}} : \{C_i^{\mathtt{GN}}\}_{i=1}^n \vdash A^{\mathtt{GN}}$ in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$ such that Π is preserved by the GN-translation.

Proof: The algorithm proceeds by recursion on the input proof $\mathcal{P}$. The translated proof $\mathcal{P}^{\mathtt{GN}}$ mimics the structure of $\mathcal{P}$, in a fairly more direct way than the corresponding Kuroda proof $\mathcal{P}^{\mathtt{KN}}$ from Theorem 2.31. This is because the Gödel-Gentzen translation of a rule of $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ is nothing but an(other) instance of the very same rule. Also the interpretation of AxSTAB becomes an(other) instance of AxSTAB, but restricted to the language $\mathcal{L}^{\mathtt{nc}}$. We already know from Proposition 1.65 that such a restriction is provable in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}}$ already. We will therefore only have to give the treatment of the strong existential axioms and of $\mathtt{AxAC}_{\mathtt{nc}}$, which are the only principles of $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ which make a visible use of the strong $\exists$. We begin with

$$[\mathtt{AxAC}_{\mathtt{nc}}]^{\mathtt{GN}} \equiv \forall x \exists^{\mathtt{cl}} y B^{\mathtt{nc}}(x,y) \to \exists^{\mathtt{cl}} Y \forall x B^{\mathtt{nc}}(x, Yx) \equiv \mathtt{AxAC}_{\mathtt{nc}}^{\mathtt{cl}} \quad ,$$

which is an axiom of the image system $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$.

We now treat the strong existential axioms. The GN-translation of $\mathtt{Ax}\exists^+$ is $\forall z_1 [A^{\mathtt{GN}}(z_1) \to \exists^{\mathtt{cl}} z_2 A^{\mathtt{GN}}(z_2)]$ which is an instance of $\mathtt{Lm}\exists_+^{\mathtt{cl}}$, that has a simple logical proof in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$, a system which enjoys full stability (which is provable in it due to the lack of restrictions on $\to^+$). See also Lemma 1.43. The situation is identical for $[\mathtt{Ax}\overline{\exists}^+]^{\mathtt{GN}}$. The GN-translation of $\mathtt{Ax}\exists^-$ is $\exists^{\mathtt{cl}} z_1 A^{\mathtt{GN}}(z_1) \wedge \forall z_2 (A^{\mathtt{GN}}(z_2) \to B^{\mathtt{GN}}) \to B^{\mathtt{GN}}$, which can be easily proved in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$ by means of Stability for $B^{\mathtt{GN}}$, $\neg\neg B^{\mathtt{GN}} \to B^{\mathtt{GN}}$. The situation is fairly similar for $[\mathtt{Ax}\overline{\exists}^-]^{\mathtt{GN}}$. The rule $\overline{\forall}^+$ can always be applied in the verifying proofs in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$, since in such proofs the restrictive situation of $\mathtt{VC}_3(\cdot)$ never occurs. $\square$

Theorem 2.38 (Main theorem on bound extraction by GNLMD-intrp.) There exists an algorithm which from a given input proof $\vdash A(a)$ in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ produces at output (pre-)majorizing terms $T[a]$ together with the verifying proof $\vdash \exists x [(\lambda a.\, T) \succeq x \wedge \forall a, y (A^{\mathtt{N}})_{\mathtt{D}}(x(a); y; a)]$ in system $\mathtt{WeZ}_{\mathtt{m}}$ plus $\mathtt{Ax}\exists^-$

and $\mathtt{Ax}\exists^+$ for the newly introduced existential variables x and for the existential variables Φ of those $\mathtt{AxCA}$ necessarily involved by the verifying proof only. The same hold if Bezem's `smaj` is used instead of Howard's $\succeq$, see (2.30).

Proof: Immediate from Theorem 2.37 followed by Corollary 2.14, taking into account that the $\mathtt{GN}$-translated input proof is in $\mathtt{WeZ}_{\mathtt{m}}^{\mathtt{nc}+}$ only (no strong existentials!). For the `smaj` variant one uses Theorem 2.23. $\square$

2.4 Light Monotone Dialectica extractions from classical analytical proofs by elimination of extensionality and ε-arithmetization

We here shortly describe how the techniques developped in the previous section can be successfully employed for the treatment of non-trivial classical analytical proofs. We follow closely a small part of the exhaustive exposition due to Kohlenbach on the subject of mining proofs which use the highly ineffective principle of Weak (aka "binary") König's Lemma (abbreviated `WKL`, see Section A.4.2 for a formal definition and a short discussion of this badly unconstructive principle), see [63] for a survey of such practical applications of the Monotone Dialectica, particularly to Numerical Functional Analysis. Our goal here is to outline the treatment of proofs which may involve the `ncm` quantifiers. We thus explore the adaptability of the various proof-theoretic techniques employed by Kohlenbach (ε-arithmetization, elimination-of-extensionality) to the new "light" setting. As one would expect, all these techniques adapt straightforwardly to the non-computational-content context. Yet some care has to be put into the formulation of some peculiar restrictions.

Epsilon-arithmetization in the context with `ncm` quantifiers

The so-called "ε-arithmetization" technique was developed by Kohlenbach initially in [64] and later in [69, 70] (see also [63] for a survey). Let

$$\Delta \quad :\equiv \quad \forall u^{\rho}\, \exists v \leq_{\sigma} ru\, \forall w^{\tau}\; B^{\mathtt{nc}}(u,v,w)$$

be an arbitrary-but-fixed (single) sentence just like in Definition 1.67, but here also restricted by $dg(\tau) \leq 2$. We associate to it the following sentences [below $B \equiv \widetilde{B^{\mathtt{nc}}}$ is the direct regular-quantifier translation of the pure-`ncm` formula

$B^{\mathtt{nc}}$, as usual; also recall that $\widetilde{\Delta} \equiv \exists V \leq_{\rho\sigma} r\, \forall u^{\rho}\, \forall w^{\tau}\, B(u, Vu, w)\,]$:

$$\overline{\Delta} \;:\equiv\; \exists V \leq_{\rho\sigma} r\, \forall u^{\rho}\, \forall w^{\tau}\, B^{\mathtt{nc}}(u, Vu, w) \tag{2.37}$$

$$\Delta_{\varepsilon} \;:\equiv\; \forall w^{\tau}, u^{\rho}\, \exists v \leq_{\sigma} ru\, \forall \widetilde{w} \leq_{\tau} w\; B(u, v, \widetilde{w}) \tag{2.38}$$

$$\widetilde{\Delta}_{\varepsilon} \;:\equiv\; \forall w^{\tau}\, \exists V \leq_{\rho\sigma} r\, \forall u^{\rho}\, \forall \widetilde{w} \leq_{\tau} w\; B(u, Vu, \widetilde{w}) \tag{2.39}$$

Thus (2.38) is the so-called "ε-weakening" of the direct regular-quantifier translation $\mathtt{\Delta_{reg}} \equiv \forall u^{\rho}\, \exists v \leq_{\sigma} ru\, \forall w^{\tau}\, B(u, v, w)$ of Δ. Δ_{ε} will be useful in the proof of Corollary 2.41 below. Similarly, (2.39) is the epsilon-weakening of Δ's $\widetilde{\cdot}$-corresponding set $\widetilde{\Delta}$. $\widetilde{\Delta}_{\varepsilon}$ is a central actor of Theorem 2.39. On the other hand, we will use (2.37) only as an intermediate step in the proof below.

Theorem 2.39 (ε-arithmetization of Δ premises in KNLMD-extraction) Let the formula A_1 and the term s be as in Theorem 2.35 (recall that $dg(\gamma) \leq 2$). Let Δ be the explicit sentence described above, which is different from the set of sentences Δ that is implicitely contained in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$, see Remark 1.80. There exists an algorithm which from a given proof

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+} \;\vdash\; \Delta \to \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_{\delta} sxk\, \exists z^{\gamma}\; A_1(x, k, y, z) \tag{2.40}$$

produces the closed term $\mathtt{t}^{(\iota\iota)\iota\gamma} \in \mathcal{T}_{\mathtt{m}}$ together with a verifying proof

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists} \;\vdash\; \widetilde{\Delta}_{\varepsilon} \to \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_{\delta} sxk\, \exists z \leq_{\gamma} \mathtt{t}xk\; \widetilde{A_1}(x, k, y, z) \tag{2.41}$$

Proof: We here write $A_1 \equiv \exists u' A^{\mathtt{nc}}(x, k, y, z, u')$ with no restriction on the degree of u'. Since immediately $\mathtt{ML}_0^{\exists,\mathtt{nc}} \vdash \overline{\Delta} \to \Delta$, we obtain effectively a proof

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+} \;\vdash\; \overline{\Delta} \to \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_{\delta} sxk\, \exists z^{\gamma}, u'\; A^{\mathtt{nc}}(x, k, y, z, u')$$

from which (using just classical logic) we get effectively a proof in $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ of

$$\forall x^{\iota\iota}, k^{\iota}\, \forall y \leq_{\delta} sxk\, \forall V \leq_{\rho\sigma} r\, \exists u^{\rho}, w^{\tau}, z^{\gamma}, u'\; [B^{\mathtt{nc}}(u, Vu, w) \to A^{\mathtt{nc}}(x, k, y, z, u')]$$

By a minor adaptation of Theorem 2.35 we then obtain effectively closed terms $\mathtt{t}, \mathtt{t}' \in \mathcal{T}_{\mathtt{m}}$ and a proof in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ of

$$\forall x^{\iota\iota}, k^{\iota}\, \forall y \leq_{\delta} sxk\, \forall V \leq_{\rho\sigma} r\, \exists u^{\rho}, u'\, \exists w \leq_{\tau} \mathtt{t}'xk\, \exists z \leq_{\gamma} \mathtt{t}xk \\ [\, B(u, Vu, w) \;\to\; A(x, k, y, z, u')\,]$$

and hence, using just an intuitionistic reasoning[12], we have a proof in $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ of

$$\forall x,k[\exists V\leq r\forall u\forall w\leq \mathtt{t}'xkB(u,Vu,w)\rightarrow \forall y\leq sxk\exists z\leq \mathtt{t}xk\exists u'A(x,k,y,z,u')]$$

which can be intuitionistically weakened to a proof in $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ of

$$\forall x,k[\forall w\exists V\leq r\forall u\forall \widetilde{w}\leq wB(u,Vu,\widetilde{w})\rightarrow \forall y\leq sxk\exists z\leq \mathtt{t}xk\widetilde{A_1}(x,k,y,z)]$$

and thus we have obtained effectively a proof in $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ of (2.41). $\square$

We now begin to use (2.38) in the following adaptation of a proposition established by Kohlenbach (see, e.g., [63]).

Proposition 2.40 $\mathtt{WeZ}^{\exists}_{\mathtt{m}}\vdash\widetilde{\Delta}_{\varepsilon}$ effectively-iff $\mathtt{WeZ}^{\exists}_{\mathtt{m}}\vdash\Delta_{\varepsilon}$.

Proof: System $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ is closed under the (full) rule of choice. This can be proved, e.g., by means of the (pure) Dialectica interpretation, see [63]. $\square$

Corollary 2.41 (Full elimination of Δ premises by ε-arithmetization) Let the formula A_1 and the term s be as in Theorem 2.35. Let Δ be a set (or conjunction) of explicit sentences as in Theorem 2.39 for which one moreover has that $\mathtt{WeZ}^{\exists}_{\mathtt{m}}\vdash\Delta_{\varepsilon}$. There exists an algorithm which from a given proof

$$\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}\ \vdash\ \Delta\rightarrow\forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y\leq_{\delta}sxk\,\exists z^{\gamma}\,A_1(x,k,y,z)$$

produces the closed term $\mathtt{t}^{(\iota\iota)\iota\gamma}\in\mathcal{T}_{\mathtt{m}}$ together with a verifying proof

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}}\ \vdash\ \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y\leq_{\delta}sxk\,\exists z\leq_{\gamma}\mathtt{t}xk\ \widetilde{A_1}(x,k,y,z)\quad .$$

As Kohlenbach points out in [70], for many ineffective theorems Δ of Mathematics (and particularly of the Analysis), the corresponding ε-weakenings Δ_{ε} are (constructively) provable in (subsystems of) $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$.

Remark 2.42 (Binary König Lemma) Principle $\mathtt{WKL}$ can be formalized in more ways as a sentence Δ for which $\mathtt{WeZ}^{\exists}_{\mathtt{m}}\vdash\Delta_{\varepsilon}$. See, e.g., [63] for a survey.

Corollary 2.43 (Elimination of $\mathtt{WKL}$ assumptions by ε-arithmetization) Let the formula A_1 and the term s be as in Theorem 2.35. There exists an algorithm which from a given proof

$$\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}\oplus\mathtt{WKL}\ \vdash\ \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y\leq_{\delta}sxk\,\exists z^{\gamma}\,A_1(x,k,y,z)$$

[12] Although in our case the verifying system $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ subsumes the full classical logic, see Proposition 1.65.

produces the closed term $\mathtt{t}^{(\iota\iota)\iota\gamma} \in \mathcal{T}_{\mathtt{m}}$ together with a verifying proof

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}} \ \vdash \ \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_{\delta} sxk\, \exists z \leq_{\gamma} \mathtt{t}xk\ \widetilde{A_1}(x,k,y,z)$$

Here the addition of WKL to $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ by $\oplus$ instead of $+$ marks the restriction that WKL shall not be used in any of the proofs of premises of instances of the quantifier-free rule of (extensionality) compatibility CMP from the input proof.

Proof: The deduction theorem applies to such a weaker adjoining of WKL via $\oplus$ to the weakly extensional system $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ (for which the usual deduction theorem fails in general). Then Corollary 2.41 applies, modulo Remark 2.42. □

Elimination-of-extensionality in the context with ncm quantifiers

Recall from Section 1.6 the definition of our pure-ncm Axioms of Choice:

$$\begin{aligned} \mathtt{AxAC}_{\mathtt{nc}} &\equiv \forall x\, \exists y\, B^{\mathtt{nc}}(x,y) \ \to \ \exists Y\, \forall x\, B^{\mathtt{nc}}(x, Y(x)) \\ \mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}} &\equiv \forall x\, \exists^{\mathtt{cl}} y\, B^{\mathtt{nc}}(x,y) \ \to \ \exists^{\mathtt{cl}} Y\, \forall x\, B^{\mathtt{nc}}(x, Y(x)) \end{aligned}$$

Recall from Definition 1.81 that our ncm weakly-extensional classical arithmetic $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ subsumes both $\mathtt{AxAC}_{\mathtt{nc}}$ and $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$. For the simplification of the exposition, we will assume in this whole subsection that the systems $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ (and its corresponding $\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ defined below) contain no implicit Δ set of axioms. In fact, Δ axioms will be explicitely added whenever this will be useful.

Let $\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ be the fully extensional variant of system $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$, obtained by adding to it the full compatibility axiom $x =_{\sigma} y \to B(x) \to B(y)$ and by restricting the axioms of choice $\mathtt{AxAC}_{\mathtt{nc}}/\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ to those instances with $dg(x) + dg(y) \leq 1$, and moreover such that:

a) If $dg(x) = 1$ then the pure-ncm radical $B^{\mathtt{nc}}(x,y)$ shall actually be an (ncm-)quantifier-free formula.

b) If $dg(x) = 0$ then all the (ncm-)quantified variables of $B^{\mathtt{nc}}$ shall have type degree at most 1 (hence all variables of $\mathtt{AxAC}_{\mathtt{nc}}/\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ shall have type degree at most 1, since in this case also $dg(Y) = 0$ is enforced).

Then we can state the following ncm-adaptation of the elimination-of-extensionality procedure (originally due to Luckhardt [90]) described in [63]:

Proposition 2.44 (ncm-elimination-of-extensionality) Assume that the definition from [63] of the extensional translation $A \mapsto A_e$ of a formula A is extended to formulas with ncm quantifiers by (as expected):

$$\begin{array}{lcl} (\overline{\exists} x^\rho A)_e & :\equiv & \overline{\exists} x^\rho (x =^e_\rho x \wedge A_e) \\ (\overline{\forall} x^\rho A)_e & :\equiv & \overline{\forall} x^\rho (x =^e_\rho x \to A_e) \end{array} \quad ,$$

where $y =^e_\sigma z$ denotes the strong hereditarily extensional equality between (higher-type) functionals (y and z). Then from a proof $\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}} \vdash A(\underline{a})$ one can effectively construct a proof $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}} \vdash \underline{a} =^e \underline{a} \to A_e(\underline{a})$ [here $\underline{a}$ are all the free variables of A and the $^+$ items of $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ are exactly those of $\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$].

Proof: By induction on the structure of the input proof, following [63]. The $\overline{\exists}$-axioms and $\overline{\forall}$-rules can be treated exactly like their isomorphic regular correspondents (i.e., the $\exists$-axioms and $\forall$-rules). No violation of the ncm restrictions may appear. The treatment of $\mathtt{AxAC}_{\mathtt{nc}}/\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ with $dg(x) = 1$ is the same as Kohlenbach's, due to the enforced quantifier-free radical. This restriction appears to be necessary because of the argument using a primitive recursive bounded search over such a radical, which therefore must be decidable[13]. For the treatment of the here more general $\mathtt{AxAC}_{\mathtt{nc}}/\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ with $dg(x) = 0$ (which can employ an unrestricted pure-ncm radical) we need to use the following very important lemma:

Lemma 2.45 (Stability of sentences relative to the *e*-translation) The following hold for arbitrary formulas $A \in \mathcal{F}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ [hence of $(\mathtt{We})\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$]:

a) If all positively (ncm-)universal and negatively (ncm-)existential quantified variables of A have type degree at most 1, then $\mathtt{IL}^{\exists,\mathtt{nc}}_{\mathtt{m}} \vdash A_e \to A$.

b) If all positively (ncm-)existential and negatively (ncm-)universal quantified variables of A have type degree at most 1, then $\mathtt{IL}^{\exists,\mathtt{nc}}_{\mathtt{m}} \vdash A \to A_e$.

Proof: We employ a simultaneous induction on the structure of A to prove that both a) and b) hold at each induction step. The base case, when A has no quantifiers at all, follows immediately by the definition of A_e, which in this case is just identical to A. The induction step for $\wedge$ and $\to$ follows by simple propositional logic, since $(A \wedge B)_e \equiv A_e \wedge B_e$ and also

[13] Notice that, unlike the regular system $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$, which subsumes the full comprehension axiom AxCA and therefore enjoys the decidability property also for formulas with quantifiers, the ncm system $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ does not necessarily feature such a property .

$(A \to B)_e \equiv A_e \to B_e$. For the quantifier steps $A \equiv \forall x A'(x)$, $A \equiv \overline{\forall} x A'(x)$, $A \equiv \exists x A'(x)$ and $A \equiv \overline{\exists} x A'(x)$ we use that $\vdash x =^e_\rho x$ for x with $dg(\rho) \leq 1$, see [63] for this working lemma. $\square$

Returning to the proof of Proposition 2.44, our choice axioms restricted for the situation when $dg(x) = 0$ fall into both categories a) and b) of Lemma 2.45, hence their e-translations are actually logically equivalent to the original. $\square$

Using both Proposition 2.44 and Lemma 2.45 we obtain the following corollary of Theorem 2.39 (recall that here $\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ subsumes no implicit Δ axioms):

Theorem 2.46 (Uniform bound extraction by EKNLMD-interpretation) Let $A_1 \equiv \exists u A^{\mathtt{nc}}(x^{\iota\iota}, k^\iota, y^\delta, z^\gamma, u)$ be a formula of $\mathcal{F}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ with x, k, y, z all its free variables, $dg(\gamma) \leq 2$, $dg(\delta) \leq 1$ and such that all positively (ncm-)universal and negatively (ncm-)existential quantified variables of $A^{\mathtt{nc}}$ have type degree at most 1. Let $s^{(\iota\iota)\iota\delta}$ be a closed term of $\mathcal{T}_{\mathtt{m}}$. Let Δ be a set of sentences like in Definition 1.67, but here restricted by $dg(\tau) \leq 2$, $dg(\sigma) \leq 1$ and further such that all positively (ncm-)existential and negatively (ncm-)universal quantified variables of $B^{\mathtt{nc}}$ have type degree at most 1. Then there exists an algorithm which from a given proof

$$\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}} + \Delta \;\vdash\; \forall x^{\iota\iota}\, \forall k^\iota\, \forall y \leq_\delta sxk\, \exists z^\gamma\, A_1(x,k,y,z) \tag{2.42}$$

produces the closed term $\mathtt{t}^{(\iota\iota)\iota\gamma}$ of $\mathcal{T}_{\mathtt{m}}$ together with a verifying proof

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}} \;\vdash\; \widetilde{\Delta}_\varepsilon \;\to\; \forall x^{\iota\iota}\, \forall k^\iota\, \forall y \leq_\delta sxk\, \exists z \leq_\gamma \mathtt{t}xk\; \widetilde{A_1}(x,k,y,z) \tag{2.43}$$

where $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ includes no implicit $\widetilde{\Delta}$ axioms and $\widetilde{A_1}(x,k,y,z) \equiv \exists u A(x,k,y,z,u)$ is the direct regular-quantifier translation of A_1. Hence $\mathtt{t}$ is a bound for z which is uniform relative to y. If moreover $\mathtt{WeZ}^{\exists}_{\mathtt{m}} \vdash \Delta_\varepsilon$ then (2.43) can be replaced by

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}} \;\vdash\; \forall x^{\iota\iota}\, \forall k^\iota\, \forall y \leq_\delta sxk\, \exists z \leq_\gamma \mathtt{t}xk\; \widetilde{A_1}(x,k,y,z)$$

Proof: We use the important fact that the fully extensional system $\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ features the deduction theorem, therefore we can rewrite (2.42) as

$$\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}} \;\vdash\; \Delta \;\to\; \forall x^{\iota\iota}\, \forall k^\iota\, \forall y \leq_\delta sxk\, \exists z^\gamma\, A_1(x,k,y,z) \tag{2.44}$$

Due to the various type-degree restrictions (not on γ, τ) from our hypothesis, the conclusion *sentence* of (2.44) can be placed into the situation a) of Lemma

2.45. By combining this with the outcome Proposition 2.44 applied to (2.44), it follows that we can construct effectively (by e-interpretation) a proof

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+} \;\vdash\; \Delta \;\rightarrow\; \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y \leq_{\delta} sxk\,\exists z^{\gamma}\,A_1(x,k,y,z)$$

which, also due to the restrictions on $dg(\gamma)$ and $dg(\tau)$, fits as an acceptable input (2.40) to the algorithm of Theorem 2.39. $\square$

Corollary 2.47 (Elimination of WKL axioms by EKNLMD-interpretation) Let the formula A_1 and the term s be as in Theorem 2.46 above. Then there exists an algorithm which from a given proof

$$\mathtt{Z}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+} + \mathtt{WKL} \;\vdash\; \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y \leq_{\delta} sxk\,\exists z^{\gamma}\,A_1(x,k,y,z)$$

produces the closed term $\mathtt{t}^{(\iota\iota)\iota\gamma} \in \mathcal{T}_{\mathtt{m}}$ together with a verifying proof

$$\mathtt{WeZ}_{\mathtt{m}}^{\exists} \;\vdash\; \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y \leq_{\delta} sxk\,\exists z \leq_{\gamma} \mathtt{t}xk\;\widetilde{A_1}(x,k,y,z)$$

Discussion

This chapter has truly represented the core of our thesis. It has described the extraction-and-soundness theorems which were established for all the techniques that we propose in this thesis, namely Light Dialectica, Light Monotone Dialectica, and their extensions to fully classical (and even partly analytical) systems. The reading of the present chapter is somewhat inseparable from the one of the previous chapter, which exposes the arithmetical systems that we use. See also the discussion at the end of Chapter 1. In the next chapter we outline an adaptation of our `ncm` techniques to the extraction of poly-time computable programs. Both this and the previous chapters are necessary for the reading of the following Chapter 3.

We would here like to add one more last comment. Let $\mathcal{S}^{\omega}$ denote as usual the full ZFC set-theoretic type structure. Let also $\mathcal{M}^{\omega}$ denote Bezem's type structure of all strongly majorizable functionals, as usual. Let `AxBAC` denote the following principle (Axiom) of Bounded Choice:

$$\forall R^{\rho\rightarrow\sigma}\,[\,\forall x^{\rho}\,\exists y \leq_{\sigma} Rx\;C(x,y,R) \;\rightarrow\; \exists Y \leq_{\rho\rightarrow\sigma} R\,\forall x\,C(x,Yx,R)\,] \qquad (2.45)$$

where ρ and σ are arbitrary types and C is an arbitrary regular formula of $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$. In this discussion we assume that $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$ contains no implicit $\widetilde{\Delta}$ set of

axioms. Since both $\mathcal{S}^\omega$ and $\mathcal{M}^\omega$ are models of the full comprehension axiom $\mathtt{AxCA}$, it can be easily established (using the related litterature) that

$$\begin{array}{rcll} \mathcal{S}^\omega & \models & \mathtt{WeZ}^{\exists}_{\mathtt{m}} + \mathtt{AxBAC} & \text{and also} \\ \mathcal{M}^\omega & \models & \mathtt{WeZ}^{\exists}_{\mathtt{m}} + \mathtt{AxBAC} & . \end{array}$$

Let $\Delta \equiv \{ \forall x^\rho \exists y \leq_\sigma rx \forall z^\tau \ B^{\mathtt{nc}}(x,y,z)\}$ be a set of sentences as in Definition 1.67. For $R := r$ and $C(x,y,R) :\equiv \forall z B(x,y,z)$ in (2.45) one obtains that

$$\forall x^\rho \exists y \leq_\sigma rx \forall z^\tau \ B(x,y,z) \ + \ \mathtt{AxBAC} \quad \vdash \quad \exists Y \leq_{\rho\sigma} r \forall x^\rho \forall z^\tau \ B(x, Yx, z)$$

Hence the whole $\widetilde{\cdot}$-correspondent of a sentence in Δ is a fully regular formula, which can be obtained from the direct regular correspondent of the original sentence and $\mathtt{AxBAC}$. In consequence,

$$\Delta_{\mathtt{reg}} + \mathtt{AxBAC} \quad \vdash \quad \widetilde{\Delta}$$

hence one can also write

$$\mathtt{WeZ}^{\exists}_{\mathtt{m}} + \Delta_{\mathtt{reg}} + \mathtt{AxBAC} \quad \vdash \quad \mathtt{WeZ}^{\exists}_{\mathtt{m}} + \widetilde{\Delta}$$

where $\Delta_{\mathtt{reg}} \equiv \{ \forall x^\rho \exists y \leq_\sigma rx \forall z^\tau \ B(x,y,z)\}$ is the direct regular-quantifier translation of Δ. We thus have that

a) If $\mathcal{S}^\omega \models \Delta_{\mathtt{reg}}$ then $\mathcal{S}^\omega \models \mathtt{WeZ}^{\exists}_{\mathtt{m}} + \widetilde{\Delta}$.

b) If $\mathcal{M}^\omega \models \Delta_{\mathtt{reg}}$ then $\mathcal{M}^\omega \models \mathtt{WeZ}^{\exists}_{\mathtt{m}} + \widetilde{\Delta}$.

If instead of a syntactic verifying proof, a simple guarantee that the verification holds in $\mathcal{S}^\omega$ $\mathtt{or}$ in $\mathcal{M}^\omega$ suffices, then w.r.t. Theorems 2.20, 2.35 and 2.46 we can consider that the input system $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}/\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ only includes Δ axioms for which $\mathcal{S}^\omega \models \Delta_{\mathtt{reg}}$ $\mathtt{or}$ only includes Δ axioms for which $\mathcal{M}^\omega \models \Delta_{\mathtt{reg}}$ and then the verification goes through in the model of choice, $\mathcal{S}^\omega$ $\mathtt{or}$ $\mathcal{M}^\omega$. Since, e.g., the Weak König's Lemma principle ($\mathtt{WKL}$) is valid in $\mathcal{S}^\omega$, and also can be expressed as a Δ-shape axiom, one thus achieves the admissibility of the addition of $\mathtt{WKL}$ as *axiom* to the *weakly extensional* systems $\mathtt{WeZ}^{\exists,\mathtt{nc}+}_{\mathtt{m}}/\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ in the (Light) Monotone Dialectica proof mining. One can thus avoid for $\mathtt{WKL}$ (and the like, see also Corollary 3.28 in Section 3.2.2) the use of the ε-arithmetization and elimination-of-extensionality techniques presented in Section 2.4 above (see Corollaries 2.43 and 2.47), if we accept a verification in $\mathcal{S}^\omega$. The gain would be that, on one hand we do not need to type-degree restrict the pure-$\mathtt{ncm}$

Axioms of Choice in the system at input and yet we can freely add $\mathtt{WKL}$ as a full-standing axiom, via a "+" and not just a "$\oplus$".

A slightly more complicated situation arrives when we are given that $\mathcal{M}^\omega \models \Delta_{\mathtt{reg}}$, but we are asked for a verification in $\mathcal{S}^\omega$, instead of $\mathcal{M}^\omega$. This can nevertheless be solved by imposing some low-type-degree restrictions both on the Δ axioms and on the conclusion formula of the proof at input. See Section 3.2.2 for more on this, in the context of a "feasible" variant of $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$. The results of Section 3.2.2, i.e., Theorem 3.27 and its Corollary 3.28 can nonetheless be rewritten back with $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc},\mathtt{c}+}$ instead of $\mathtt{PbZ}^{\mathtt{c}+}$ and with "$\exists z \leq_\gamma \mathtt{t}xk$" instead of "$\exists z \leq_\gamma \lambda u^\iota, l^\iota.\, \overline{p}[x^M, k, u^M, l]$". Notice the partial similarity of the restrictions on the degrees of the types from both Δ and the conclusion sentence A_1 in Theorems 2.46 and 3.27. In fact the two theorems complement each other. On one hand one allows full extensionality and less type-degree restrictions on Δ and A_1, but on the other hand the pure-$\mathtt{ncm}$ Axioms of Choice are no longer type-degree restricted. This comparison implicitely assumed that we are concerned with sentences Δ which are valid in $\mathcal{S}^\omega$ or in $\mathcal{M}^\omega$, which is actually the case in the proof-mining practice, see the work of Kohlenbach [63].

Chapter 3

Feasible systems of Arithmetic and Analysis

Ever since its beginnings, the research in the area of program extraction from proofs has been concerned with the issue of the efficiency of extracted programs. In the case of Gödel's Dialectica interpretation [45] it rapidly became clear that any concrete mathematical application of such a powerful proof-theoretical technique would have to face the difficult Contraction problem. Many simplifications of the treatment of Contraction were proposed. This is how the Diller-Nahm variant [25] of Dialectica and Kreisel's non-counter-example interpretation and Modified Realizability [85] came into being. The drawback of all these proposed simplifications was the loss of logical power in the proofs at input. In the case of Modified Realizability, a partial repair of this problem was provided by the Friedman-Dragalin $\mathtt{A}$-translation, which allows some limited amount of non-constructive reasoning in the proofs to be constructively interpreted. Also the monotone functional interpretation [68] was designed in order to give a simpler treatment of contraction by taking a maximum of the previously extracted terms instead of deciding which one to chose according to the actual validity of the contraction formula.

In parallel with this research on the simplification of the logical framework of the term-extraction techniques, a related work had started more from the computer-science side on the computational-complexity simplification of the term system, basically Gödel's $\mathbf{T}$. Full logical-arithmetical systems were then to be designed in order to formalize reasoning about such low-complexity functionals. In [22] a system $\mathtt{PV}^\omega$ of terms denoting poly-time computable functions was designed and corresponding intuitionistic $\mathtt{IPV}^\omega$ and classical

$\mathtt{CPV}^\omega$ arithmetics were given in order to reason about properties of such terms. Moreover, program-extraction procedures by both modified realizability and Dialectica interpretation were described for $\mathtt{IPV}^\omega$. It follows that the programs extracted in this way from proofs in (extensions of) $\mathtt{IPV}^\omega$ denote poly-time computable functions and hence are feasible in this sense[1].

Ferreira gives in [32] a system BTFA of *Feasible Analysis* for which the Skolem functions for the provable Π^0_2 sentences denote poly-time computable functions. BTFA is a second-order theory and the method of reading the Skolem functions is model-theoretic. More recently, the connection with Cook and Urquhart's systems was made by Oliva in [95]. It is shown in Section 3.2 of [95] how BTFA can be almost entirely simulated into $\mathtt{CPV}^\omega + \mathtt{AxAC}_0$, where $\mathtt{AxAC}_0$ denotes the quantifier-free restriction of the Axiom of Choice AxAC (here introduced in Section 1.4). Basically only the more general version of *bounded collection* $\Sigma^b_\infty - \mathtt{BC}$ of BTFA appears to be unprovable in $\mathtt{CPV}^\omega + \mathtt{AxAC}_0$, although a weaker version of it is mentioned to be provable. Let $\Pi^0_1 - \mathtt{WKL}$ be the axiom scheme of weak König's Lemma for trees defined by Π^0_1 formulas of $\mathtt{CPV}^\omega$. The main result of [95] is that from proofs $\mathtt{CPV}^\omega + \mathtt{AxAC}_0 + \Pi^0_1 - \mathtt{WKL} \vdash \forall x \exists y A_0(x,y)$ one extracts by purely syntactical means terms of $\mathtt{PV}^\omega + \mathcal{B}$ (where $\mathcal{B}$ is a special constant, see below) which denote poly-time computable functions h such that $A_0(x, hx)$ holds true in the standard model (stated as Corollary 5.2 in [95]). The syntactic extraction is performed by double-negation translation followed by an extension of Cook and Urquhart's Dialectica interpretation for system $\mathtt{IPV}^\omega$. The function h is denoted by a term in $\mathtt{PV}^\omega$ extended by a simpler, binary form of Howard's simplification [59] of Spector's bar recursion [119], introduced in [95] as a new constant denoted $\mathcal{B}$. Since it represents a form of unbounded search, the functional denoted by $\mathcal{B}$ is not expressible as a $\mathtt{PV}^\omega$ term. However, $\mathcal{B}$ is interpretable into Scarpellini's [108] type structure $\mathcal{C}$ of all continuous set-theoretical functionals where it can be eliminated in terms of limited recursion on notation. The execution of the poly-time Skolem function h is therefore not fully syntactical by the method of [95].

A complete syntactical extraction technique was later developed by Ferreira and Oliva in [33, 34] so that not only the reading, but also the execution of Skolem functions becomes strictly algorithmic. Initially one extracts just bounds for the realizers [33], but with a verifying proof in pure $\mathtt{IPV}^\omega$. Exact

[1] The full syntactic execution of these programs is nonetheless not necessarily poly-time, but only after an initial partial evaluation phase which has a super-exponential (in basic term data) worst-case complexity, see Remarks 3.7 and 3.9 below.

realizers (for the original specification) can then be obtained from this $\mathtt{IPV}^\omega$ proof by just the original technique of Cook and Urquhart [34].

In parallel an arithmetical system G_2A^ω was designed by Kohlenbach [69] as part of a full hierarchy of so-called Grzegorczyk Arithmetics $(G_nA^\omega)_{n\in\mathbb{N}}$ such that the bounds extracted by the monotone Dialectica interpretation are syntactically reducible to integer polynomials and hence denote poly-time computable functions as well. Moreover, a *hereditarily polynomial bounded analysis* PBA is built upon G_2A^ω and a comparison with Ferreira's BTFA is given in [72]. It turns out that a good deal of Analysis can be formalized in PBA (see [72]), whereas much less could be formalized in BTFA (see, e.g., [31]). However, the two frameworks are found to be incomparable in general [72].

3.1 The poly-time Arithmetic/Analysis $\mathtt{PtZ}^+$

We here use a Natural Deduction adaptation of Oliva's variant [95] of the poly-time intuitionistic arithmetic $\mathtt{IPV}^\omega$ originally introduced by Cook and Urquhart in [22] as a higher-order and logical extension of Cook's equational term system PV from [21]. Aside for the more readable names for the *Chop*, *Pad* and *Smash* function symbols, we also retain Oliva's choice that the set of constants is restricted to a finite number of base symbols, unlike [22] where each PV-term introduces a new function symbol.

We regard the underlying term system of PtZ (which is a slight adaptation of $\mathtt{PV}^\omega$) not as an equational theory but as a term rewriting system. This is more suitable when working with purely syntactical methods in view of a computer implementation of the syntactic reductions. We also use a *boolean* base type o, just like in the development of system WeZ in Chapter 2. We will denote by $\flat^-$ the base type for *binary* strictly positive integers. This is different from the base type ι for *unary* positive integers which was introduced in Chapter 2. We will establish later in the sequel the natural embedding of ι into $\flat^-$, which is a bijective function. Since ι and $\flat^-$ are syntactically different it will be possible to use both of them in a single coherent theory. For this goal the development of o here is fully compatible with o of Chapter 2 so that they can be confused without any harm. Unlike in [95, 22] the natural number 0 is not part of the $\flat^-$ type. We find it more convenient to use a constant $\mathtt{1}^{\flat^-}$ which denotes the positive integer 1 but corresponds to $\mathtt{O}^\iota$ in the embedding $\iota \mapsto \flat^-$. On the other hand $\mathtt{1}^{\flat^-}$ corresponds to Cook and Urquhart's term $s_1(0)$. Hence the standard model for $\flat^-$ is the set of strictly positive integers (whose least

element is 1), denoted $\mathbb{N}^\star$ as usual.

Remark 3.1 We may introduce the special constant $\mathtt{0}$ in an extension of type $\flat^-$ denoted $\flat$, whose model is $\mathbb{N}$. We can then build an alternative variant of $\mathtt{PtZ}$ based on type $\flat$ in which $\mathtt{0}$ can only be obtained as the (unary) predecessor of $\mathtt{1}$ and equally $\mathtt{1}$ is the *unary* successor of $\mathtt{0}$ (see Definition 3.4 below). Special computation rules must be given for $\mathtt{0}$ in relationship with the other constants, particularly in connection with the recursor (see Definition 3.2 below). Also the induction axiom has to be adapted as well (see Definition 3.3 below).

Accordingly, the *binary length* $|n|$ of the strictly positive integer n, is the number of bits $\{0,1\}$ following the initial bit 1. Hence $|\overline{1}| \equiv 0$, $|\overline{11}| \equiv |\overline{10}| \equiv 1$, etc. We generally denote by m, n natural numbers from the model $\mathbb{N}^\star$ and we write $\overline{m}, \overline{n}$ for their binary syntactic correspondents in $\flat^-$. For $\mathtt{PtZ}$ terms t of type $\flat^-$ we denote by $t_{\mathbb{N}^\star}$ their correspondent in the model (hence, e.g., $(\overline{n})_{\mathbb{N}^\star} \equiv n$).

Definition 3.2 The term constants of $\mathtt{PtZ}$ are as follows:

- $\mathtt{tt}^o$ and $\mathtt{ff}^o$ which denote boolean truth and falsity;
- for each type τ the *selector* $\mathtt{If}_\tau$ of type $o\,\tau\,\tau\,\tau$ which denotes choice according to a boolean condition with the usual *if-then-else* semantics;
- equality $\mathtt{Eq}^{\flat^-\flat^-o}$ and inequality $\mathtt{Leq}^{\flat^-\flat^-o}$, both functional constants here;
- $\mathcal{R}_\tau$ of type $\tau\,(\flat^-\,\flat^-\,\tau)\,(\flat^-\,\tau)\,\flat^-\,\tau$ which denotes *limited* recursion on notation; here type τ is restricted to be arithmetic, i.e., $\tau \equiv \underline{\sigma}\flat^-$
- $\mathtt{1}^{\flat^-}$ which denotes the natural number 1;
- $\overset{0}{\mathtt{Su}}^{\flat^-\flat^-}$ and $\overset{1}{\mathtt{Su}}^{\flat^-\flat^-}$ which denote the functions which append a 0, respectively a 1 to the right of their binary argument, i.e., $\overset{0}{\mathtt{Su}}_{\mathbb{N}^\star}(\overline{n}) \equiv \overline{n0} \equiv 2n$ and $\overset{1}{\mathtt{Su}}_{\mathbb{N}^\star}(\overline{n}) \equiv \overline{n1} \equiv 2n+1$;
- $\mathtt{Even}^{\flat^-o}$ which denotes the function returning $\mathtt{tt}$ iff the rightmost bit of its binary argument is 0, i.e., $\mathtt{Even}_{\mathbb{N}^\star}(\overline{n0}) \equiv \mathtt{tt}$ and $\mathtt{Even}_{\mathbb{N}^\star}(\overline{n1}) \equiv \mathtt{ff}$;
- $\mathtt{Hal}^{\flat^-\flat^-}$ s.t. $\mathtt{Hal}_{\mathbb{N}^\star}(\overline{n})$ eliminates the rightmost bit of $\overline{n}$, if $n > 1$;
- $\mathtt{Cho}^{\flat^-\flat^-\flat^-}$ s.t. $\mathtt{Cho}_{\mathbb{N}^\star}(\overline{m}, \overline{n})$ chops off $|\overline{n}|$ bits from the right of $\overline{m}$;

- $\mathtt{Pad}^{\flat^-\flat^-\flat^-}$ s.t. $\mathtt{Pad}_{\mathbb{N}^\star}(\overline{m},\overline{n})$ appends $|\overline{n}|$ zero bits to the right of $\overline{m}$;
- $\mathtt{Sma}^{\flat^-\flat^-\flat^-}$ s.t. $\mathtt{Sma}_{\mathbb{N}^\star}(\overline{m},\overline{n})$ appends $|\overline{m}|\cdot|\overline{n}|$ zero bits to $\overline{1}$;

Definition 3.3 System $\mathtt{PtZ}$ is built exactly like system $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ from Section 1.3 (see Definition 1.47) - regarding definitions of terms, predicates, formulas and logic - but with the following differences:

- type $\flat^-$ replaces the type ι;
- the full induction rule $\mathtt{IR}_\mathrm{o}$ is replaced by the following *polynomial* induction axiom scheme (below $s \leq_{\flat^-} t$ is a short for $\mathtt{at}(\mathtt{Leq}\, s\, t)$, as expected)

  $\mathtt{PIND}^\omega: \quad A(\mathtt{1}) \to \forall x\,[A(\mathtt{Hal}\, x) \to A(x)] \to \forall x A(x)$

  where A has the special form $(\exists y \leq_{\flat^-} t)\, A_0(x,y) \equiv \exists y.\, y \leq_{\flat^-} t \wedge A_0(x,y)$, with x, y and t all of type $\flat^-$ and also all free variables of t have type $\flat^-$;
- the following rewrite rules are used for natural numbers:

$$\begin{array}{rcl@{\qquad}rcl}
\mathtt{Eq}(\overset{0}{\mathtt{Su}}\, z, \mathtt{1}) & \hookrightarrow & \mathtt{ff} & \mathtt{Eq}(\overset{1}{\mathtt{Su}}\, z, \mathtt{1}) & \hookrightarrow & \mathtt{ff} \\
\mathtt{Eq}(\overset{0}{\mathtt{Su}}\, x, \overset{1}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{ff} & \mathtt{Eq}(\overset{1}{\mathtt{Su}}\, x, \overset{0}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{ff} \\
\mathtt{Eq}(\overset{0}{\mathtt{Su}}\, x, \overset{0}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Eq}(x,y) & \mathtt{Eq}(\overset{1}{\mathtt{Su}}\, x, \overset{1}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Eq}(x,y) \\
\mathtt{Leq}(\overset{0}{\mathtt{Su}}\, x, \overset{0}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Leq}(x,y) & \mathtt{Leq}(\overset{0}{\mathtt{Su}}\, z, \mathtt{1}) & \hookrightarrow & \mathtt{ff} \\
\mathtt{Leq}(\overset{1}{\mathtt{Su}}\, x, \overset{1}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Leq}(x,y) & \mathtt{Leq}(\overset{1}{\mathtt{Su}}\, z, \mathtt{1}) & \hookrightarrow & \mathtt{ff} \\
\mathtt{Leq}(\overset{0}{\mathtt{Su}}\, x, \overset{1}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Leq}(x,y) & \mathtt{Leq}(\mathtt{1}, z) & \hookrightarrow & \mathtt{tt} \\
\mathtt{Leq}(\overset{1}{\mathtt{Su}}\, x, \overset{0}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{If}_o\, (\mathtt{Eq}\; x\; y)\; \mathtt{ff}\; (\mathtt{Leq}\; x\; y) & & & \\
\mathtt{Even}(\overset{0}{\mathtt{Su}}\, z) & \hookrightarrow & \mathtt{tt} & \mathtt{Even}(\overset{1}{\mathtt{Su}}\, z) & \hookrightarrow & \mathtt{ff} \\
\mathtt{Hal}(\overset{0}{\mathtt{Su}}\, z) & \hookrightarrow & z & \mathtt{Hal}(\overset{1}{\mathtt{Su}}\, z) & \hookrightarrow & z \\
\mathtt{Pad}(x, \overset{0}{\mathtt{Su}}(y)) & \hookrightarrow & \mathtt{Pad}(\overset{0}{\mathtt{Su}}(x), y) & \mathtt{Hal}(\mathtt{1}) & \hookrightarrow & \mathtt{1} \\
\mathtt{Pad}(x, \overset{1}{\mathtt{Su}}(y)) & \hookrightarrow & \mathtt{Pad}(\overset{0}{\mathtt{Su}}(x), y) & \mathtt{Pad}(x, \mathtt{1}) & \hookrightarrow & x \\
\mathtt{Cho}(\mathtt{1}, y) & \hookrightarrow & \mathtt{1} & \mathtt{Cho}(x, \mathtt{1}) & \hookrightarrow & x \\
\mathtt{Cho}(\overset{0}{\mathtt{Su}}\, x, \overset{0}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Cho}(x,y) & \mathtt{Cho}(\overset{1}{\mathtt{Su}}\, x, \overset{1}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Cho}(x,y) \\
\mathtt{Cho}(\overset{0}{\mathtt{Su}}\, x, \overset{1}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Cho}(x,y) & \mathtt{Cho}(\overset{1}{\mathtt{Su}}\, x, \overset{0}{\mathtt{Su}}\, y) & \hookrightarrow & \mathtt{Cho}(x,y)
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathtt{Sma}(1,y) & \hookrightarrow & 1 & \mathtt{Sma}(\overset{\mathtt{i}}{\mathtt{Su}}\,x,y) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Pad}\,(\mathtt{Sma}\,x\,y)\,y)\\
\mathtt{Sma}(x,1) & \hookrightarrow & 1 & \mathtt{Sma}(x,\overset{\mathtt{i}}{\mathtt{Su}}\,y) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Pad}\,(\mathtt{Sma}\,x\,y)\,x)
\end{array}$$

$$\begin{array}{rcl}
\mathcal{R}_\tau\,x\,y^{\flat^-\flat^-\tau}\,v\,1\,\underline{w} & \hookrightarrow & x\,\underline{w}\\
\mathcal{R}_\tau\,x^\tau\,y\,v^{\flat^-\tau}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,z^{\flat^-})\,\underline{w} & \hookrightarrow & \mathtt{If}_{\flat^-}\,(\mathtt{Eq}\,1\,(\mathtt{Cho}\,(t_0[z])\,(v\,(\overset{\mathtt{0}}{\mathtt{Su}}\,z)\,\underline{w}))\,,\,t_0[z]\,,\,v\,(\overset{\mathtt{0}}{\mathtt{Su}}\,z)\,\underline{w})\\
\mathcal{R}_\tau\,x^\tau\,y\,v^{\flat^-\tau}\,(\overset{\mathtt{1}}{\mathtt{Su}}\,z^{\flat^-})\,\underline{w} & \hookrightarrow & \mathtt{If}_{\flat^-}\,(\mathtt{Eq}\,1\,(\mathtt{Cho}\,(t_1[z])\,(v\,(\overset{\mathtt{1}}{\mathtt{Su}}\,z)\,\underline{w}))\,,\,t_1[z]\,,\,v\,(\overset{\mathtt{1}}{\mathtt{Su}}\,z)\,\underline{w})
\end{array}$$

– where we abbreviated by $\quad t_i[z] :\equiv y\,(\overset{\mathtt{i}}{\mathtt{Su}}\,z)\,(\mathcal{R}_\tau\,x\,y\,v\,z\,\underline{w})\,\underline{w} \quad$ for $i \in \overline{0,1}$.

Just like in Chapter 1, predicate equality at base types is defined by $s =_o t :\equiv \mathtt{at}(s) \leftrightarrow \mathtt{at}(t)$ and $s =_{\flat^-} t :\equiv \mathtt{at}(\mathtt{Eq}\,s\,t)$. Equality at higher types is extensionally defined as well, i.e., $s =_{\sigma\tau} t \equiv \forall x^\sigma\,(s\,x =_\tau t\,x)$ and non-equality is defined as $s \neq_\tau t :\equiv \neg(s =_\tau t)$. The behaviour of equality is governed by the rules of (higher order) reflexivity `REF`, symmetry `SYM`, transitivity `TRZ`, compatibility `CMP` and substitution `SUB` together with the special axiom scheme `AxEQL`. Recall that the latter adds an axiom $s =_\tau t$ for all terms s^τ and t^τ for which $s \hookrightarrow^\star \cdot\, {}^\star\!\hookleftarrow t$ can be established (recall from Definition 1.45 that the rewrite relation $\hookrightarrow$ includes the rules of α, β and η reduction).

Definition 3.4 We distinguish the following particular `PtZ` terms:

- $\mathtt{Suc} :\equiv \lambda z.\,\mathcal{R}_{\flat^-}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,1\,,\ \lambda x,y.\,\mathtt{If}_{\flat^-}\,(\mathtt{Even}\,x\,,\,\overset{\mathtt{1}}{\mathtt{Su}}\,(\mathtt{Hal}\,x)\,,\,\overset{\mathtt{0}}{\mathtt{Su}}\,y)\,,\,\overset{\mathtt{1}}{\mathtt{Su}}\,,\,z\,)$ which denotes the unary successor, i.e., $\mathtt{Suc}_{\mathbb{N}^\star}(n) \equiv n+1$;
- $\mathtt{Pred} :\equiv \lambda z.\,\mathcal{R}_{\flat^-}\,(\,1\,,\ \lambda x,y.\,\mathtt{If}_{\flat^-}\,(\mathtt{Even}\,x\,,\,\overset{\mathtt{1}}{\mathtt{Su}}\,y\,,\,\overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Hal}\,x))\,,\,\overset{\mathtt{0}}{\mathtt{Su}}\,,\,z\,)$ which denotes the unary predecessor, i.e., $\mathtt{Pred}_{\mathbb{N}^\star}(n) \equiv n-1$, $\mathtt{Pred}_{\mathbb{N}^\star}(1) \equiv 1$;
- boolean conjunction $\mathtt{And}^{ooo}$ and implication $\mathtt{Imp}^{ooo}$ and the *n-selector* $\mathtt{If}^n_\tau$ of type $\overbrace{o\ldots o}^{n}\,\overbrace{\tau\ldots\tau}^{n}\,\tau\,\tau$ which are identical to those in Chapter 1;
- the *zero terms* $\mathtt{0}_\tau$ are defined for every type τ like in Chapter 1, here with $\mathtt{0}_{\flat^-} :\equiv 1$.

System `PtZ` is immediately seen to be equivalent with $\mathtt{IPV}^\omega$ without 0, in the sense that $\mathtt{IPV}^\omega \vdash \forall z A(\mathtt{Suc}\,z)$ iff $\mathtt{PtZ} \vdash \forall z A(z)$. Hence `PtZ` terms of shape $t^{\flat^-}[x_1^{\flat^-},\ldots,x_k^{\flat^-}]$ denote poly-time computable functions $t_{\mathbb{N}^\star}[n_1,\ldots,n_k]$. However, for both `PtZ` and $\mathtt{IPV}^\omega$, the poly-time computability only holds in the model and is not doubled by a purely syntactic reduction of length at most

a polynomial in the lengths of the arguments. Only the terms $t^{\bar{\flat}}[x_1^{\bar{\flat}}, \ldots, x_k^{\bar{\flat}}]$ of the first-order system PV have the property that $t[\overline{n_1}, \ldots, \overline{n_k}]$ syntactically reduces to its normal form $\overline{t_{\mathbb{N}^\star}[n_1, \ldots, n_k]}$ in less than $p_t(|\overline{n_1}|, \ldots, |\overline{n_k}|)$ steps, where $p_t[m_1, \ldots, m_k]$ is an integer polynomial. This property can only be achieved because of the heavy restriction that the maximal type degree of PV (sub)terms is 1. Therefore lambda-abstracted variables can only be of base type (o or $\bar{\flat}$) and new constant function symbols of type degree 1 must be introduced for each recursive definition, in order to avoid the use of type 2 recursor constants.

The syntactic reduction from terms $t^{\bar{\flat}}[x_1^{\bar{\flat}}, \ldots, x_k^{\bar{\flat}}]$ of $\mathtt{PV}^\omega$ to their corresponding PV term $t^{\mathtt{PV}}[x_1^{\bar{\flat}}, \ldots, x_k^{\bar{\flat}}]$ is fully described in [22], but no bounds are given there on its length in terms of t data (like size, depth, maximal type degree). Notice that the length of the reduction $t \overset{\mathtt{PV}}{\rightsquigarrow} t^{\mathtt{PV}}$ obviously does not depend on the lengths of the actual inputs to the program t.

Definition 3.5 The super-exponential function $2_{(\cdot)}(\cdot) : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$ is recursively defined by $2_0(n) :\equiv n$ and $2_{m+1}(n) :\equiv 2^{2_m(n)}$.

Remark 3.6 Recall from Definition 1.6 that $mdg(t)$, $d(t)$, $S(t)$ denote the maximal type degree of a subterm, the depth and respectively the size of t.

Remark 3.7 From [8] (see also [113, 112]) it follows that the $\beta\eta$-reduction of t to its $\beta\eta$ normal form alone (hence without reducing the higher-type recursors), which is part of the more complex $\overset{\mathtt{PV}}{\rightsquigarrow}$ reduction, is upper bounded by $2_{mdg(t)}(S(t))$ and $2_{mdg(t)+1}(d(t))$. On the other hand, almost matching super-exponential lower bound examples[2] can be given, i.e., there exist terms $(t_n^m)_{m,n\in\mathbb{N}}$ such that $S(t_n^m) = O(n)$ for all $m \in \mathbb{N}$, $mdg(t_n^m) = m + 1$ for all $m, n \in \mathbb{N}$ and `every` $\beta\eta$ reduction sequence of t_n^m has length greater than $2_{m-1}(n) - n$, for sufficiently large n.

The situation gets even worse when the reduction of higher type recursors $\mathcal{R}_\tau$ is taken into account as well. But, after this at least (worst case) super-exponential partial reduction $\overset{\mathtt{PV}}{\rightsquigarrow}$, the reduced $t^{\mathtt{PV}}$ will further syntactically reduce in a polynomial number of steps to its normal form $\overline{t_{\mathbb{N}^\star}[n_1, \ldots, n_k]}$, when presented with the actual inputs $\overline{n_1}, \ldots, \overline{n_k}$ (this follows immediately from [21, 22]).

[2] In [8] these are directly adapted from [112], but the origin of all such lower bound examples can be traced back to works of Statman [120], Orevkov [99] and Pudlak [105].

Remark 3.8 Let $\mathtt{PV}(\mathtt{PtZ})$ be the subset of $\mathtt{PtZ}$-terms of maximal type degree 1, except that type-2 recursor constants are allowed as subterms of elements of $\mathtt{PV}(\mathtt{PtZ})$. These type-2 recursors are in an obvious bijective correspondence with Cook's [21, 22] "flat" recursors $\mathrm{R}[g,h,k]$ of system $\mathtt{PV}$. Then $\mathtt{PV}(\mathtt{PtZ})$ is immediately seen to be equivalent to Cook's $\mathtt{PV}$ without 0 and hence terms $t^{\flat^-}[x_1^{\flat^-},\ldots,x_k^{\flat^-}] \in \mathtt{PV}(\mathtt{PtZ})$ are syntactically poly-time computable in the sense mentioned above. Also the reduction from a regular term $t^{\flat^-}[x_1^{\flat^-},\ldots,x_k^{\flat^-}]$ of $\mathtt{PtZ}$ to its corresponding $t^{\mathtt{PV}}[x_1^{\flat^-},\ldots,x_k^{\flat^-}] \in \mathtt{PV}(\mathtt{PtZ})$ can be defined equivalently to the reduction $\overset{\mathtt{PV}}{\rightsquigarrow}$ of Cook and Urquhart.

It follows that the syntactic reduction of $\mathtt{PtZ}$ terms $t[\overline{n_1},\ldots,\overline{n_k}]$ to their normal form $\overline{t_{\mathbb{N}^\star}[n_1,\ldots,n_k]}$ can be split in two stages:

1. first the generic reduction $t^{\flat^-}[x_1^{\flat^-},\ldots,x_k^{\flat^-}] \overset{\mathtt{PV}}{\rightsquigarrow} t^{\mathtt{PV}}[x_1^{\flat^-},\ldots,x_k^{\flat^-}]$ is performed once and for all as a partial evaluation of the given term; this potentially costs enormously in terms of t data like $mdg(t)$, $d(t)$ and $S(t)$, since super-exponential lower bounds exist for (part of) this reduction, see Remark 3.7 above;

2. the $\mathtt{PV}(\mathtt{PtZ})$ term $t^{\mathtt{PV}}[\overline{n_1},\ldots,\overline{n_k}]$ reduces in at most $p_t(|\overline{n_1}|,\ldots,|\overline{n_k}|)$ steps to the normal form $\overline{t_{\mathbb{N}^\star}[n_1,\ldots,n_k]}$, where $p_t[m_1,\ldots,m_k]$ is an integer polynomial; hence this part is computationally feasible, in contrast to 1.

Remark 3.9 This division of labour strategy outlines the importance of partial evaluation, since the most costly part of the reduction steps is carried out just once, regardless of the actual numerical inputs. The generic reduction is more suitably processed on a super-computer, especially when t features large size or maximal type degree. But after $t^{\mathtt{PV}}$ is obtained, the reductions for its actual values can be executed efficiently (i.e., poly-time) just by a normal computer.

System $\mathtt{PtZ}^+$ is obtained from $\mathtt{PtZ}$ in the same way that $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ is obtained from $\mathtt{WeZ}^{\exists,\mathtt{nc}}$ in Section 1.4 (see Definition 1.58). I.e., by adding to $\mathtt{PtZ}$ the axioms $\mathtt{AxSTAB}^{\mathtt{nc}}$, $\mathtt{AxMK}$, $\mathtt{AxIP}^{-\exists}_{+\forall}$, $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$, $\mathtt{AxAC}$ and $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$, all adapted to the new language and term system of $\mathtt{PtZ}$. Similarly, also system $\mathtt{PtZ}^-$ is obtained from $\mathtt{PtZ}$ just like $\mathtt{WeZ}^{\exists}$ from $\mathtt{WeZ}^{\exists,\mathtt{nc}}$, i.e., by eliminating the $\mathtt{ncm}$ quantifiers (and their contingent restrictions). Let also $\mathtt{PtZ}^-_{\mathtt{cl}}$ be the extension of $\mathtt{PtZ}^-$ with full stability $\neg\neg A \to A$. Then the following analogue of Corollary 2.11 holds for our variant of the Cook-Urquhart Dialectica interpretation.

Theorem 3.10 (Poly-time realizer synthesis by the LD-interpretation) There exists an algorithm which from a given proof $\mathtt{PtZ}^+ \vdash A(a)$ produces exact realizing terms $T[a] \in \mathtt{PV}(\mathtt{PtZ})$ with a verifying proof $\mathtt{PtZ}^-_{\mathtt{cl}} \vdash \forall y A_{\mathtt{D}}(T; y; a)$.

Proof: A soundness theorem for the Cook-Urquhart Dialectica interpretation of $\mathtt{IPV}^\omega + \mathtt{AxMK} + \mathtt{AxAC}$ is given by Oliva in [95] as an immediate extension of Cook and Urquhart's original proof from [22]. Since the light Dialectica interpretation of $\mathtt{AxIP}^{-\exists}_{+\forall}$ and $\mathtt{AxIP}^{\mathtt{cl}}_{\mathtt{nc}}$ is realized by simply projection functionals (see the proof of Theorem 2.10), which are known to be part of the term system of PtZ, it is then immediate that LD-interpretation is sound for $\mathtt{PtZ}^+$ (the Light Dialectica treatment of $\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ in the Cook-Urquhart setting is similar to the treatment of AxAC). The extracted PtZ terms can thereafter be reduced to terms in PV(PtZ) via the technique mentioned in Remark 3.8, which is originally due to Cook and Urquhart [22] as well. □

3.2 Polynomial bounded Arithm/Analysis $\mathtt{PbZ}^{\mathtt{c}+}$

In the survey paper [72] Kohlenbach gives and arguments for his proposal of a framework for the development of "Feasible Analysis". He introduces systems of "hereditarily Polynomial Bounded Analysis" (abbreviated PBA) as subsets of analysis whose provably recursive functions can be bounded with integer polynomials. PBA are defined as $G_2A^\omega + \mathtt{AxAC}_{\mathtt{o}} + \Delta$, where G_2A^ω is the second system in a discrete hierarchy of so-called Grzegorczyk Arithmetics and Δ are sets of analytical axioms having the logical form $\forall x^\delta \exists y \leq_\rho sx \forall z^\tau B_0(x, y, z)$ (as in Definition 1.67, but with the pure-ncm root reduced to quantifier-free only) which cover important subsets of classical analysis (see [69, 70, 72]).

We give below an adaptation of Kohlenbach's systems G_2A^ω and G_3A^ω from [69] (the latter being necessary for the verifying proof). Since none of the systems G_nA^ω with $n \geq 4$ is involved in the description or analysis of PBA systems we choose a more direct exposition of (our variant of) G_2A^ω and G_3A^ω, without mentioning the Ackerman functions or the Grzegorczyk hierarchy.

Definition 3.11 (Polynomial-bound system PbZ) We thus introduce the system PbZ as an adaptation of Kohlenbach's G_2A^ω by describing its differences to system $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ (without any Π or Δ included, i.e., with $\Pi, \Delta \equiv \emptyset$) from Section 1.5. The only elements of $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$ which are not included in PbZ are the induction rule $\mathtt{IR}_{\mathtt{o}}$ and Gödel's recursor R. The type ι for naturals of PbZ is identical to that of $\mathtt{WeZ}^{\exists,\mathtt{nc}}_{\mathtt{m}}$, hence PbZ uses a unary representation of

positive integers as well. We also add a *minimum* constant $\mathtt{min}$ of type $\iota\iota$ together with its defining axioms:

$$\mathtt{min}\, x\, y \leq_\iota x \qquad \mathtt{min}\, x\, y \leq_\iota y \quad .$$

Here $s \leq_\tau t :\equiv t \geq_\tau s$, with $\geq_\tau$ extensionally defined like in Section 1.5. *Predecessor*, *addition* and *multiplication* are denoted by the functional constants $\mathtt{Pred}^{\iota\iota}$, $\mathtt{Pls}^{\iota\iota\iota}$ and $\mathtt{Tms}^{\iota\iota\iota}$, which are provided with the defining rewrite rules:

$$\begin{array}{rclcrcl}
\mathtt{Pred}\,\mathtt{0} & \hookrightarrow & \mathtt{0} & \qquad & \mathtt{Pred}\,(\mathtt{Suc}\,x) & \hookrightarrow & x \\
\mathtt{Pls}(x,\mathtt{0}) & \hookrightarrow & x & & \mathtt{Pls}(\mathtt{Suc}\,x,y) & \hookrightarrow & \mathtt{Suc}\,(\mathtt{Pls}\,x\,y) \\
\mathtt{Pls}(\mathtt{0},x) & \hookrightarrow & x & & \mathtt{Pls}(x,\mathtt{Suc}\,y) & \hookrightarrow & \mathtt{Suc}\,(\mathtt{Pls}\,x\,y) \\
\mathtt{Tms}(x,\mathtt{0}) & \hookrightarrow & \mathtt{0} & & \mathtt{Tms}(\mathtt{Suc}\,x,y) & \hookrightarrow & \mathtt{Pls}\,y\,(\mathtt{Tms}\,x\,y) \\
\mathtt{Tms}(\mathtt{0},x) & \hookrightarrow & x & & \mathtt{Tms}(x,\mathtt{Suc}\,y) & \hookrightarrow & \mathtt{Pls}\,x\,(\mathtt{Tms}\,x\,y)
\end{array}$$

Also the functional constants $\mathtt{Max}$ and $\mathtt{Sum}$ of type $(\iota\iota)\iota\iota$ are included, where their meaning is given by

$$\begin{array}{rcl}
\mathtt{Max}_{\mathbb{N}}\, f_{\mathbb{N}}\, n_{\mathbb{N}} & \equiv & max_{\mathbb{N}}(f\,0, f\,1, \ldots, f\,n) \\
\mathtt{Sum}_{\mathbb{N}}\, f_{\mathbb{N}}\, n_{\mathbb{N}} & \equiv & f(0) +_{\mathbb{N}} f(1) +_{\mathbb{N}} \ldots +_{\mathbb{N}} f(n)
\end{array}$$

which is achieved syntactically by the following rewrite rules:

$$\begin{array}{rclcrcl}
\mathtt{Max}\, f\, \mathtt{0} & \hookrightarrow & f\,\mathtt{0} & \qquad & \mathtt{Max}\, f\, (\mathtt{Suc}\, x) & \hookrightarrow & \mathtt{Max}(f\,(\mathtt{Suc}\,x), \mathtt{Max}\, f\, x) \\
\mathtt{Sum}\, f\, \mathtt{0} & \hookrightarrow & f\,\mathtt{0} & & \mathtt{Sum}\, f\, (\mathtt{Suc}\, x) & \hookrightarrow & \mathtt{Pls}(f\,(\mathtt{Suc}\,x), \mathtt{Sum}\, f\, x)
\end{array}$$

Recursion is limited to the bounded and predicative recursor $\widetilde{\mathtt{R}}$ of type

$$(\sigma_1 \ldots \sigma_k\, \iota)\; (\iota\, \iota\, \sigma_1 \ldots \sigma_k\, \iota)\; \iota\; (\iota\, \sigma_1 \ldots \sigma_k\, \iota)\; \sigma_1 \ldots \sigma_k\; \iota$$

syntactically defined by the following rewrite rules:

$$\begin{array}{rcl}
\widetilde{\mathtt{R}}\, x\, y\, \mathtt{0}\, v\, \underline{w} & \hookrightarrow & x\, \underline{w} \\
\widetilde{\mathtt{R}}\, x\, y\, (\mathtt{Suc}\, z)\, v\, \underline{w} & \hookrightarrow & \mathtt{min}(y\, z\, (\widetilde{\mathtt{R}}\, x\, y\, z\, v\, \underline{w})\, \underline{w}\,,\; v\, z\, \underline{w})
\end{array}$$

The bounded search operator μ_b of type $(\iota\iota o)\iota\iota$ is syntactically given by:

$$\begin{array}{rcl}
\mu_b\; f^{\iota\iota o}\; x & \hookrightarrow & \mu'_b\; (f^{\iota\iota o}\, x)\; x \\
\mu'_b\; g^{\iota o}\; \mathtt{0} & \hookrightarrow & \mathtt{0} \\
\mu'_b\; g^{\iota o}\; (\mathtt{Suc}\, z) & \hookrightarrow & \mathtt{If}_o\; (g\,\mathtt{0})\; \mathtt{0}\; [\,\mathtt{If}_o\; (g\,(\mathtt{Suc}\, z)) \\
 & & \qquad [\mathtt{If}_o\,(\mathtt{Eq}\,(\mu'_b\, g\, z)\,\mathtt{0})\,(\mathtt{Suc}\, z)\,(\mu'_b\, g\, z)]\; (\mu'_b\, g\, z)\,]
\end{array}$$

where the additional operator μ'_b of type $(\iota o)\iota\iota$ is used and their semantics is

$$\begin{aligned} \mu'_b\, g_{\mathbb{N}}\, n_{\mathbb{N}} &\equiv min\,\{m \leq_{\mathbb{N}} n \mid g(m) \text{ is true}\} \\ \mu_b\, f_{\mathbb{N}}\, n_{\mathbb{N}} &\equiv min\,\{m \leq_{\mathbb{N}} n \mid f(n,m) \text{ is true}\} \quad . \end{aligned}$$

Definition 3.12 (Verifying systems $\mathtt{PbZ}_{\mathtt{v}}$ and $\mathtt{PbZ}^{+}_{\mathtt{v}}$) System $\mathtt{PbZ}_{\mathtt{v}}$ is the extension with the full comprehension axiom $\exists \Phi^{\tau\to o}\, \forall x^{\tau}\, [\,\mathtt{at}(\Phi x) \leftrightarrow B(x)\,]$ ($\mathtt{AxCA}$) of the restriction of $\mathtt{PbZ}$ to the language without $\mathtt{ncm}$ quantifiers. It thus corresponds to $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ (without any $\widetilde{\Pi}$ or $\widetilde{\Delta}$, i.e., with $\widetilde{\Pi}, \widetilde{\Delta} \equiv \emptyset$), see Definition 1.64. Moreover, system $\mathtt{PbZ}^{+}_{\mathtt{v}}$ is obtained by adding to $\mathtt{PbZ}_{\mathtt{v}}$ the functional constants $\mathtt{Exp}^{\iota\iota\iota}$ and $\mathtt{Prod}^{(\iota\iota)\iota\iota}$ together with their defining rewrite rules:

$$\begin{array}{rclcrcl} \mathtt{Exp}(x,\mathtt{0}) & \hookrightarrow & 1 & \qquad & \mathtt{Exp}(x,\mathtt{Suc}y) & \hookrightarrow & \mathtt{Tms}(x,\mathtt{Exp}\,x\,y) \\ \mathtt{Prod}(f,\mathtt{0}) & \hookrightarrow & f\,\mathtt{0} & & \mathtt{Prod}(f,\mathtt{Suc}\,x) & \hookrightarrow & \mathtt{Tms}(f(\mathtt{Suc}\,x),\mathtt{Prod}\,f\,x) \end{array}$$

Thus, whereas $\mathtt{PbZ}_{\mathtt{v}}$ is still a polynomial system, its extension $\mathtt{PbZ}^{+}_{\mathtt{v}}$ becomes an even over-poly-time system since it contains true exponentials.

Remark 3.13 Of course, $\mathtt{Pred}$, $\mathtt{Max}$, $\mathtt{Sum}$, μ_b and $\mathtt{Prod}$ could be defined as lambda-terms from the remaining constants, but we here follow [69] in choosing to introduce them directly as constants (with their own defining rewrite rules) in order to attain a larger flexibility of the language. Also recall from Section 1.5 that in the given context with $\mathtt{at}$ and $\bot$ being the only predicate symbols, $\mathtt{AxCA}$ implies full stability $\neg\neg A \to A$ in systems $\mathtt{PbZ}_{\mathtt{v}}$ and $\mathtt{PbZ}^{+}_{\mathtt{v}}$, which thus subclude full classical logic.

Definition 3.14 Let $\mathtt{Tm}^{-}(\mathtt{PbZ})$ denote the set of all terms of $\mathtt{PbZ}$ which do not contain $\mathtt{Max}$, $\mathtt{Sum}$, μ_b and $\widetilde{\mathtt{R}}$ (and eventually also not $\mathtt{Pred}$, $\mathtt{min}$ and $\mathtt{Max}$, see Remark 3.16 below).

Proposition 3.15 To every term $t[x_1,\ldots,x_k]$ of $\mathtt{PbZ}$ (with $x_1,\ldots,x_k$ all free variables of t) one can syntactically associate a term $t^{\star}[x_1,\ldots,x_k] \in \mathtt{Tm}^{-}(\mathtt{PbZ})$ (with $x_1,\ldots,x_k$ all free variables of $t^{\star}$) such that:

$$\vdash\ x_1^{\star} \succeq x_1 \to \ldots \to x_k^{\star} \succeq x_k \to t^{\star}[x_1^{\star},\ldots,x_k^{\star}] \succeq t[x_1,\ldots,x_k] \qquad (3.1)$$

Proof: This follows immediately from Proposition 1.78 once majorants from $\mathtt{Tm}^{-}(\mathtt{PbZ})$ are provided for the constants specific to $\mathtt{PbZ}$. This is fulfilled since

$$\begin{array}{rcll} \mathtt{Max}^{\star} :\equiv \lambda f^{\iota\iota}, x^{\iota}.\, f(x) & \succeq_{(\iota\iota)\iota\iota} & \mathtt{Max} \\ \mathtt{Sum}^{\star} :\equiv \lambda f^{\iota\iota}, x^{\iota}.\, \mathtt{Tms}(f(x),\mathtt{Suc}\,x) & \succeq_{(\iota\iota)\iota\iota} & \mathtt{Sum} \\ \mu_b^{\star} :\equiv \lambda f^{\iota\iota o}, x^{\iota}.\, x & \succeq_{(\iota\iota o)\iota\iota} & \mu_b \\ \widetilde{\mathtt{R}}^{\star} :\equiv \lambda x^{\underline{\sigma}\iota}, y^{\iota\underline{\sigma}\iota}, z^{\iota}, v^{\iota\underline{\sigma}\iota}, \underline{w}^{\underline{\sigma}}.\, \mathtt{Max}(x\,\underline{w}\,,\ v(\mathtt{Pred}\,z,\underline{w})) & \succeq_{(\underline{\sigma}\iota)(\iota\underline{\sigma}\iota)\iota(\iota\underline{\sigma}\iota)\underline{\sigma}} & \widetilde{\mathtt{R}} \end{array}$$

and also $\mathtt{O} \succeq \mathtt{O}$, $\mathtt{Suc} \succeq \mathtt{Suc}$, $\mathtt{Pred} \succeq \mathtt{Pred}$, $\mathtt{min} \succeq \mathtt{min}$ and $\mathtt{Max} \succeq \mathtt{Max}$. Constants inherited from $\mathtt{WeZ}$ have their majorants provided by Proposition 1.78 already in $\mathtt{Tm}^{-}(\mathtt{PbZ})$. Only the majorant of $\mathtt{If}_\tau$ for arithmetic τ needs to be majorized by $\lambda p^o, y^\tau, z^\tau, \underline{w}.\, \mathtt{Pls}(y\underline{w})(z\underline{w})$ if one uses the more stripped version of $\mathtt{Tm}^{-}(\mathtt{PbZ})$, see Remark 3.16 below. □

Remark 3.16 The proposition above corresponds to Proposition 2.2.21 of [69]. Since $(\lambda x^\iota.\, x) \succeq \mathtt{Pred}$, $\mathtt{Pls} \succeq \mathtt{min}$ and $\mathtt{Pls} \succeq \mathtt{Max}$, also $\mathtt{Pred}$, $\mathtt{min}$ and $\mathtt{Max}$ can be eliminated from $t^\star$ and hence generally from terms of $\mathtt{Tm}^{-}(\mathtt{PbZ})$.

Proposition 3.17 Let $t \in \mathtt{Tm}^{-}(\mathtt{PbZ})$ be a term of type ι such that all its free variables are $x_1^{\sigma_1}, \ldots, x_k^{\sigma_k}$ with $dg(\sigma_i) \leq 1$ for $i \in \overline{1,k}$. Then the $\beta\eta$ normal form $\hat{t}[x_1, \ldots, x_k]$ of t is an element of $\mathtt{Tm}^{-}(\mathtt{PbZ})$ built without λ-abstraction.

Proof: It is obvious that $\hat{t} \in \mathtt{Tm}^{-}(\mathtt{PbZ})$. Assume that $\hat{t}$ contains a λ-abstraction and let $\lambda x.\, r$ be a maximal lambda subterm of $\hat{t}$ in the sense that no y and s exist such that $\lambda y.\, s$ is a subterm of $\hat{t}$ and $\lambda x.\, r$ is a subterm of s. Since $dg(\hat{t}) = 0$ it is impossible that $\hat{t} \equiv \lambda x.\, r$. Hence there exists a subterm s of $\hat{t}$ such that $(\lambda x.\, r)s$ or $s(\lambda x.\, r)$ is a subterm of $\hat{t}$. The former case is excluded because $\hat{t}$ is in $\beta\eta$ normal form and for the same reason also the case that s is in lambda form. It follows that s decomposes as $s_0 \ldots s_n$ with $n \in \mathbb{N}$ and s_0 not in lambda form. Since s_0 can also not be in application form, it follows that s_0 is a constant or variable. Since $dg(\lambda x.\, r) \geq 1$, the type degree of s_0 is then at least 2, hence the former case is excluded. For the same reason s_0 can also not be one of $\hat{t}$'s free variables $x_1, \ldots, x_k$. But if s_0 is a bound variable y, then the whole $s(\lambda x.\, r)$ is a subterm of $\hat{t}$'s subterm bound by λy, which contradicts the maximality of $\lambda x.\, r$ in this respect. This concludes the proof. □

Remark 3.18 The proposition above corresponds to Proposition 2.2.22 of [69], there formulated only for closed terms of a $\Sigma\Pi$ combinator calculus. It also corresponds to the reduction $\overset{\mathtt{PV}}{\rightsquigarrow}$ of Cook and Urquhart [22] and to its analogue reduction from $t^{\flat^-}[x_1^{\flat^-}, \ldots, x_k^{\flat^-}]$ of $\mathtt{PtZ}$ to its corresponding $t^{\mathtt{PV}}[x_1^{\flat^-}, \ldots, x_k^{\flat^-}]$ (an element of $\mathtt{PV}(\mathtt{PtZ})$), which we describe in Section 3.1. The same almost matching super-exponential upper and lower bounds (adapted from [8, 113, 112]) apply to the length of the $\beta\eta$ reduction from t to $\hat{t}$.

Definition 3.19 Let $p \in \mathbb{N}[x_1, \ldots, x_k]$ be a polynomial in variables $x_1, \ldots, x_k$ with coefficients in $\mathbb{N}$. To it we associate the *syntactic polynomial* $\overline{p} \in \mathtt{Tm}^{-}(\mathtt{PbZ})$

obtained from p by using $\overbrace{\texttt{Suc}\ldots\texttt{Suc}}^{c\text{ times}}\,\texttt{0}$ for the natural constant $c_{\mathbb{N}}$, $\texttt{Pls}$ for the integer addition $+_{\mathbb{N}}$ and $\texttt{Tms}$ for the integer multiplication $\times_{\mathbb{N}}$.

Proposition 3.20 Let $t[x_1,\ldots,x_k]^\iota$ be a term of $\texttt{PbZ}$ such that $x_1^{\sigma_1},\ldots,x_k^{\sigma_k}$ are all its free variables and $dg(\sigma_i)\leq 1$ for $i\in\overline{1,k}$. Then there exists an integer polynomial $p_t\in\mathbb{N}[x_1,\ldots,x_k]$ such that

$$\vdash\ x_1^\star\ \succeq\ x_1\to\ldots\to x_k^\star\ \succeq\ x_k\to\overline{p_t}[x_1^\star,\ldots,x_k^\star]\ \succeq\ t[x_1,\ldots,x_k] \tag{3.2}$$

where $\overline{p_t}$ denotes the syntactic version of p_t in the sense of Definition 3.19.

Proof: We here use the more stripped version of $\texttt{Tm}^-(\texttt{PbZ})$ from Remark 3.16. By combining Proposition 3.15 with Proposition 3.17, one obtains a term $\widehat{t^\star}$ of $\texttt{PbZ}$ built only by application from just $\texttt{0}$, $\texttt{Suc}$, $\texttt{Pls}$, $\texttt{Tms}$ and $x_1,\ldots,x_k$, such that

$$\vdash\ x_1^\star\ \succeq\ x_1\to\ldots\to x_k^\star\ \succeq\ x_k\to\widehat{t^\star}[x_1^\star,\ldots,x_k^\star]\ \succeq\ t[x_1,\ldots,x_k] \tag{3.3}$$

It is immediate that $\widehat{t^\star}$ can be put in a (syntactic) polynomial form, in the sense that there exists a $p_t\in\mathbb{N}[x_1,\ldots,x_k]$ such that

$$\vdash\ \widehat{t^\star}[x_1,\ldots,x_k]=_\iota\overline{p_t}[x_1,\ldots,x_k] \tag{3.4}$$

and then (3.2) follows from (3.3) and (3.4) by Lemma 1.72. $\square$

Remark 3.21 Even though the (syntactic) obtaining of $\widehat{t^\star}$ from $t[x_1^\iota,\ldots,x_k^\iota]^\iota$ is of (worst-case) iterated exponential complexity in the maximal type degree of t and the size of t (see Remark 3.18, Section 3.1 and [8, 113, 112]), the syntactic execution of $\widehat{t^\star}$ on actual inputs $\overline{n_1},\ldots,\overline{n_k}$ is of poly-time complexity $p_t[n_1,\ldots,n_k]$, where $\vdash\ \widehat{t^\star}[x_1,\ldots,x_k]=_\iota\overline{p_t}[x_1,\ldots,x_k]$ is given by (3.4). Notice that a *true* poly-time complexity $p_t[|n_1|,\ldots,|n_k|]$ in the binary lengths of the integer inputs can be attained only if natural numbers are syntactically (re)presented in binary form like in type $\flat^-$ of system $\texttt{PtZ}$ from Section 3.1.

Definition 3.22 (Input system $\texttt{PbZ}^{\texttt{c+}}$) We define the input system $\texttt{PbZ}^{\texttt{c+}}$ as the extension of $\texttt{PbZ}$ with $\texttt{AxSTAB}$ and $\texttt{AxAC}_{\texttt{nc}}/\texttt{AxAC}_{\texttt{nc}}^{\texttt{cl}}$ (recall Definition 1.81 from Section 1.6) adapted to the language and terms of $\texttt{PbZ}$. Similar to system $\texttt{WeZ}_{\texttt{m}}^{\exists,\texttt{nc},\texttt{c+}}$ from Section 1.6, also the polynomial-bounded system $\texttt{PbZ}^{\texttt{c+}}$ includes the arbitrary but fixed sets of sentences (below $B^{\texttt{nc}}$ denotes pure-$\texttt{ncm}$ formulas)

$$\begin{aligned}\Pi\ &\equiv\ \{\,\forall\underline{b}\,B^{\texttt{nc}}(\underline{b})\,\}\quad\text{and}\\ \Delta\ &\equiv\ \{\,\forall x^\rho\,\exists y\leq_\sigma rx\,\forall z^\tau\ B^{\texttt{nc}}(x,y,z)\,\}\quad.\end{aligned}$$

(see Convention 1.57 and Definition 1.67), here all restricted to the language and terms of $\mathtt{PbZ}$. Parallel to that, we also include into the polynomial-bounded verifying system $\mathtt{PbZ}_\mathtt{v}$ the corresponding sets of sentences

$$\begin{aligned}\widetilde{\Pi} &\equiv \{\ \forall \underline{b}\ B(\underline{b})\ \}\\ \widetilde{\Delta} &\equiv \{\ \exists Y \leq_{\rho\sigma} r\ \forall x^\rho\ \forall z^\tau\ B(x, Yx, z)\ \}\end{aligned}$$

which are bijectively associated to Π and respectively Δ above. Thus $\mathtt{PbZ}_\mathtt{v}$ corresponds to the "regular" monotonic (classical) system $\mathtt{WeZ}^\exists_\mathtt{m}$ from Section 1.5.

We then obtain the following adaptation of Theorem 3.2.2 of [69], here based on Theorem 2.35.

Theorem 3.23 (Polynomial bound synthesis by KNLMD-interpretation) Let $A_1(x^{\iota\iota}, k^\iota, y^\delta, z^\gamma)$ be a quasi-purely-existential formula with x, k, y, z all its free variables (i.e., $A_1 \equiv \exists v A^{\mathtt{nc}}(x, k, y, z, v)$) and $dg(\gamma) \leq 2$. Let $s^{(\iota\iota)\iota\delta}$ be a closed term of $\mathtt{PbZ}$. There exists an algorithm which from a given proof

$$\mathtt{PbZ}^{\mathtt{c}+} \ \vdash\ \forall x^{\iota\iota}\ \forall k^\iota\ \forall y \leq_\delta sxk\ \exists z^\gamma\ A_1(x, k, y, z) \tag{3.5}$$

produces the syntactic polynomial $\overline{p}[x^{\iota\iota}, k^\iota, u^{\iota\iota}, l^\iota]^\iota \in \mathtt{Tm}^{-}(\mathtt{PbZ})$ such that

$$\mathtt{PbZ}_\mathtt{v} \ \vdash\ \forall x^{\iota\iota}\ \forall k^\iota\ \forall y \leq_\delta sxk\ \exists z \leq_\gamma \lambda u^{\iota\iota}, l^\iota.\, \overline{p}[x^M, k, u^M, l]\ \widetilde{A_1}(x, k, y, z) \tag{3.6}$$

where u, l are (possibly empty) tuples determined by γ and $\widetilde{A_1}(x, k, y, z) \equiv \exists v A(x, k, y, z, v)$, i.e., it is the regular-quantifier translation of A_1. Hence if $\gamma \equiv \iota$ then p is a polynomial bound for z in x^M and k which is uniform w.r.t. y.

Proof: We first apply Theorem 2.35 to (3.5) and obtain a closed $t \in \mathtt{Tm}(\mathtt{PbZ})$ such that

$$\mathtt{PbZ}_\mathtt{v} \ \vdash\ \forall x^{\iota\iota}\ \forall k^\iota\ \forall y \leq_\delta sxk\ \exists z \leq_\gamma txk\ \widetilde{A_1}(x, k, y, z) \tag{3.7}$$

Let $u^{\iota\iota}$ and l^ι be (possibly empty) tuples such that $txkul$ has type ι. Moreover, x, k, u and l are all the free variables of $txkul$. Therefore we can apply Proposition 3.20 and we thus obtain effectively the syntactic polynomial $\overline{p}[x, k, u, l]^\iota \in \mathtt{Tm}^{-}(\mathtt{PbZ})$ such that

$$\vdash\ x^\star \succeq x \wedge k^\star \succeq k \wedge u^\star \succeq u \wedge l^\star \succeq l\ \rightarrow\ \overline{p}[x^\star, k^\star, u^\star, l^\star]\ \succeq txkul$$

and since $x^M \succeq x$, $k \succeq k$, $u^M \succeq u$ and $l \succeq l$ we obtain that

$$\mathtt{PbZ}_\mathtt{v} \ \vdash\ \forall x, k, u, l\ (\ \overline{p}[x^M, k, u^M, l]\ \geq_\iota\ txkul\ )$$

hence

$$\mathtt{PbZ}_\mathtt{v} \ \vdash\ \forall x, k\ (\ \lambda u, l.\, \overline{p}[x^M, k, u^M, l]\ \geq_\gamma\ \lambda u, l.\, txkul\ =_\gamma\ txk\ ) \tag{3.8}$$

Then the conclusion (3.6) immediately follows from (3.7) and (3.8) $\square$

3.2.1 Elimination of the non-standard analytical axiom $\mathtt{F}^-$

Among the sentences Δ of Definition 3.22 which are restricted to the language without $\mathtt{ncm}$ quantifiers one distinguishes the very important axiom

$$\begin{aligned}\mathtt{F}^- \quad :\equiv \quad & \forall \Phi^{\iota(\iota\iota)\iota}\, \forall x^{\iota\iota\iota}\, \exists y \leq_{\iota\iota\iota} x\, \forall k^{\iota}\, \forall z^{\iota\iota}\, \forall n^{\iota} \\ & [\, \wedge_{i<_\iota n}(zi \leq_\iota xki) \,\rightarrow\, \Phi k\,(\lambda k^{\iota}.\, \mathtt{If}_\iota(k <_\iota n)(zk)\mathtt{0}) \leq_\iota \Phi k(yk)\,]\end{aligned}$$

The sentence $\mathtt{F}^-$ is *non-standard* in the sense that it does not hold in the full set-theoretic type structure $\mathcal{S}^\omega$ but is valid, e.g., in the type structure $\mathcal{M}^\omega$ of strongly majorizable functionals. It is possible to eliminate the use of (the $\widetilde{=}$-corresponding) $\widetilde{\mathtt{F}^-}$ in the verifying proof whenever $\mathtt{F}^-$ is not used in any of the proofs of premises of instances of the quantifier-free rule of (extensionality) compatibility $\mathtt{CMP}$ in the proof at input. One marks this restriction on the use of $\mathtt{F}^-$ by adding it via a $\oplus\mathtt{F}^-$ to the main system, instead of the usual $+\mathtt{F}^-$.

We then obtain, in the spirit of Theorem 2.39 here (and of its Corollaries 2.41 and 2.43), the following adaptation of Theorem 4.21 of [69].

Corollary 3.24 In Theorem 3.23 the premise (3.5) can be replaced by

$$\mathtt{PbZ}^{\mathtt{c}+} \oplus \mathtt{F}^- \;\vdash\; \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_\delta sxk\, \exists z^{\gamma}\, A_1(x,k,y,z)$$

or, equivalently, by

$$\mathtt{PbZ}^{\mathtt{c}+} \;\vdash\; \mathtt{F}^- \rightarrow \forall x^{\iota\iota}\, \forall k^{\iota}\, \forall y \leq_\delta sxk\, \exists z^{\gamma}\, A_1(x,k,y,z)$$

and then the verifying proof (3.6) is in $\mathtt{PbZ}^+_{\mathtt{v}}$ instead of just $\mathtt{PbZ}_{\mathtt{v}}$, where $\mathtt{PbZ}^+_{\mathtt{v}}$ is the extension of $\mathtt{PbZ}_{\mathtt{v}}$ with $\mathtt{Exp}$ and $\mathtt{Prod}$, see Definition 3.12. The exponential functional constants $\mathtt{Exp}$ and $\mathtt{Prod}$ are needed in the verifying proof only because of the elimination of $\mathtt{F}^-$.

Let $\mathtt{E}-\mathtt{PbZ}^{\mathtt{c}+}$ be the fully extensional variant of system $\mathtt{PbZ}^{\mathtt{c}+}$, obtained from it by removing all implicit Δ axioms, by adding the full compatibility axiom "$x =_\sigma y \rightarrow B(x) \rightarrow B(y)$" and by restricting the choice axioms $\mathtt{AxAC}_{\mathtt{nc}}/\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ to those instances with $dg(x) + dg(y) \leq 1$, and moreover such that:

a) If $dg(x) = 1$ then the pure-$\mathtt{ncm}$ radical $B^{\mathtt{nc}}(x,y)$ shall actually be an ($\mathtt{ncm}$-)quantifier-free formula.

b) If $dg(x) = 0$ then all the ($\mathtt{ncm}$-)quantified variables of $B^{\mathtt{nc}}$ shall have type degree at most 1 (hence all variables of $\mathtt{AxAC}_{\mathtt{nc}}/\mathtt{AxAC}^{\mathtt{cl}}_{\mathtt{nc}}$ shall have type degree at most 1, since in this case also $dg(Y) = 0$ is enforced).

Then, by means of the ncm-adapted (see Section 2.4) procedure for the elimination of extensionality, originally due to Luckhardt [90], we obtain the following adaptation of Theorem 3.3 of [72]:

Corollary 3.25 In Theorem 3.23, the premise (3.5) can be replaced by

$$\mathtt{E}\text{-}\mathtt{PbZ}^{\mathtt{c}+} + \mathtt{F}^{-} \;\vdash\; \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y \leq_{\delta} sxk\,\exists z^{\gamma}\; A_1(x,k,y,z)$$

where besides $dg(\gamma) \leq 2$ we also restrict $dg(\delta) \leq 1$. Just like in Corollary 3.24, the verifying proof (3.6) is in $\mathtt{PbZ}^{+}_{\mathtt{v}}$ instead of $\mathtt{PbZ}_{\mathtt{v}}$. Here as well, the exponential functional constants Exp and Prod are only needed in the verifying proof because of the elimination of $\mathtt{F}^{-}$.

Of course, the elimination of $\mathtt{F}^{-}$ in Corollaries 3.24 and 3.25 can be obtained as a particular case of the (ncm-adapted, see Section 2.4) more general "ε-arithmetization" technique developed by Kohlenbach initially in [64] and later in [69, 70] (see also [63] for a survey). Basically all the main results of Section 2.4, i.e., Theorem 2.35, Corollary 2.41, Proposition 2.40 and Theorem 2.46 can be rewritten in the polynomial-bound context by replacing $\mathtt{WeZ}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ with $\mathtt{PbZ}^{\mathtt{c}+}$, $\mathtt{Z}^{\exists,\mathtt{nc},\mathtt{c}+}_{\mathtt{m}}$ with $\mathtt{E}\text{-}\mathtt{PbZ}^{\mathtt{c}+}$, and then their corresponding $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$ with $\mathtt{PbZ}^{+}_{\mathtt{v}}$.

Remark 3.26 (Kohlenbach,[70]) For many ineffective theorems Δ of the Analysis, the corresponding ε-weakenings Δ_{ε} are (constructively) provable in (subsystems of) $\mathtt{WeZ}^{\exists}_{\mathtt{m}}$. E.g., for $\Delta \equiv \mathtt{F}^{-}$ one has $\mathtt{PbZ}^{+}_{\mathtt{v}} \vdash (\mathtt{F}^{-})_{\varepsilon}$, see [69].

It is now clear how Corollaries 3.24 and 3.25 can be obtained as immediate consequences of the polynomial-bound adapted Corollary 2.41 and respectively Theorem 2.46. Even more important in the ncm context is that instead of $\mathtt{F}^{-}$ we can here also use the ncm variant $\mathtt{F}^{\mathtt{nc}}$ of the stronger axiom **F**, defined by (3.13) in the next subsection 3.2.2, see it also for the definition of $\mathtt{F}^{\mathtt{nc}}$.

3.2.2 Verification in the full set-theoretic type structure

Let $\Delta \equiv \{\,\forall x^{\rho}\,\exists y \leq_{\sigma} rx\,\forall z^{\tau}\; B^{\mathtt{nc}}(x,y,z)\}$ be as in Definition 3.22, hence possibly including formulas which may contain ncm quantifiers. From here and to the end of this section, the sentences in Δ are restricted to formulas in which all positively (ncm-)universal and negatively (ncm-)existential quantified variables have type degree at most 2 and also all positively (ncm-)existential and negatively (ncm-)universal quantified variables have type degree at most 1. Hence in particular, the regularly universal quantified variables x^{ρ} and z^{τ} have

the restriction $dg(\rho), dg(\tau) \leq 2$, and also the regularly existential quantified variable y^σ has the restriction $dg(\sigma) \leq 1$. We further assume that $\mathcal{S}^\omega \models \Delta_{\mathtt{reg}}$, where $\Delta_{\mathtt{reg}} \equiv \{ \forall x^\rho \exists y \leq_\sigma rx \forall z^\tau \ B(x,y,z)\}$ is the direct regular-quantifier translation of Δ (here B is the usual full regular-quantifier translation of $B^{\mathtt{nc}}$). Let also Δ' be another set of sentences like in Definition 3.22, but moreover such that $\mathcal{M}^\omega \models \Delta'_{\mathtt{reg}}$, where $\Delta'_{\mathtt{reg}}$ is the direct regular-quantifier translation of Δ', as expected (no type degree restriction). From here on to the end of this section, system $\mathtt{PbZ}^{\mathtt{c}+}$ will no longer contain any implicit Δ-kind of axiom set.

Finally, in this section we settle that the Π-set of axioms of $\mathtt{PbZ}^{\mathtt{c}+}$ (recall Definition 3.22) consists of all possible sentences of the Π-shape (see Conventions 1.66 and 1.57) for which the direct regular-quantifier correspondents from $\widetilde{\Pi}$ are true (in the sense of the full ZFC set-theoretic type structure $\mathcal{S}^\omega$, i.e., $\mathcal{S}^\omega \models \widetilde{\Pi}$) and which moreover fulfill the following restriction[3] on the type degrees of their variables[4]: all positively (`ncm`-)universal and negatively `ncm`-existential quantified variables have type degree at most 2 and also all positively `ncm`-existential and negatively `ncm`-universal quantified variables have type degree at most 1. Notice that this limitation is consistent with Kohlenbach's restriction from [69], where the variables of the (there) *purely universal* Π-sentences of G_nA^ω are forced to have type degree ≤ 2.

If instead of a syntactic verifying proof, a simple guarantee that the verification holds in the full set-theoretic type structure $\mathcal{S}^\omega$ suffices, then the following extraction theorems can be established in the spirit of Theorem 4.9 of [69].

Theorem 3.27 Let $A_1(x^{\iota\iota}, k^\iota, y^\delta, z^\gamma)$ be a quasi-purely-existential `PbZ` formula with x, k, y, z all its free variables, i.e., $A_1 \equiv \exists v A^{\mathtt{nc}}(x^{\iota\iota}, k^\iota, y^\delta, z^\gamma, v^\alpha)$, and moreover such that $dg(\delta) \leq 1$, $dg(\gamma), dg(\alpha) \leq 2$ and further all positively `ncm`-universal and negatively `ncm`-existential quantified variables of $A^{\mathtt{nc}}$ have type degree at most 1 and also all positively `ncm`-existential and negatively `ncm`-universal quantified variables of $A^{\mathtt{nc}}$ have type degree at most 2. Let $s^{(\iota\iota)\iota\delta}$ be a closed term of `PbZ`. Let Δ and Δ' be the explicit sets of axiom sentences defined above (recall that $\mathcal{S}^\omega \models \Delta_{\mathtt{reg}}$ and $\mathcal{M}^\omega \models \Delta'_{\mathtt{reg}}$). Then there

[3] Notice that this is the same restriction as for the explicit Δ axiom set above. Since the only regular quantifiers of a Π axiom are the leading positive universal ones, the restriction is here somewhat simpler, with only one parenthesized "(`ncm`-)".

[4] All these explicit type-degree limitations, although apparently artifficial due to their complicated expression, are truly effectively needed for the extraction and soundness theorems presented below.

exists an algorithm which from a given proof

$$\mathtt{PbZ}^{\mathtt{c}+}+\Delta+\Delta' \ \vdash \ \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y\leq_{\delta} sxk\,\exists z^{\gamma}\,A_1(x,k,y,z) \tag{3.9}$$

produces the syntactic polynomial $\overline{p}[x^{\iota\iota},k^{\iota},u^{\iota\iota},l^{\iota}]^{\iota}\in \mathtt{Tm}^{-}(\mathtt{PbZ})$ such that

$$\mathcal{S}^{\omega} \ \models \ \forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y\leq_{\delta} sxk\,\exists z\leq_{\gamma}\lambda u^{\iota\iota},l^{\iota}.\,\overline{p}[x^{M},k,u^{M},l]\ \widetilde{A_1}(x,k,y,z) \tag{3.10}$$

where u,l are (possibly empty) tuples determined by γ and $\widetilde{A_1}(x,k,y,z)\equiv \exists v A(x,k,y,z,v)$, i.e., it is the regular-quantifier translation of A_1. Hence if $\gamma\equiv\iota$ then p is a polynomial bound for z in x^M and k which is uniform w.r.t. y.

Proof: Just like in the proof of Theorem 3.23, one first algorithmically obtains a verifying proof in $\mathtt{PbZ}_{\mathtt{v}}+\widetilde{\Delta}+\widetilde{\Delta'}$ of

$$\forall x^{\iota\iota}\,\forall k^{\iota}\,\forall y\leq_{\delta} sxk\,\exists z\leq_{\gamma}\lambda u^{\iota\iota},l^{\iota}.\,\overline{p}[x^{M},k,u^{M},l]\ \widetilde{A_1}(x,k,y,z) \tag{3.11}$$

Note that the sentences in both $\widetilde{\Delta}$ and $\widetilde{\Delta'}$ are all of the shape $\exists Y\leq_{\rho\sigma} r\forall x^{\rho}\,\forall z^{\tau}\,B(x,Yx,z)$, where the formula B may use only regular quantifiers, which are the direct regular correspondents of the $\mathtt{ncm}$ quantifiers which possibly occur in the Δ-sentence of origin, namely $\forall x^{\rho}\exists y\leq_{\sigma} rx\forall z^{\tau}\,B^{\mathtt{nc}}(x,y,z)$. Let $\mathtt{AxBAC}$ denote the following principle (Axiom) of Bounded Choice:

$$\forall R^{\rho\to\sigma}\,[\,\forall x^{\rho}\exists y\leq_{\sigma} Rx\,C(x,y,R)\ \to\ \exists Y\leq_{\rho\to\sigma} R\forall x\,C(x,Yx,R)\,]$$

where ρ and σ are arbitrary types and C is an arbitrary regular formula of $\mathtt{PbZ}$, see also Definition 3.2.1 of [69]. Thus for $R:=r$ and $C(x,y,R):\equiv\forall z B(x,y,z)$ one obtains that

$$\forall x^{\rho}\,\exists y\leq_{\sigma} rx\,\forall z^{\tau}\,B(x,y,z)\ +\ \mathtt{AxBAC}\quad\vdash\quad\exists Y\leq_{\rho\sigma} r\,\forall x^{\rho}\,\forall z^{\tau}\,B(x,Yx,z)$$

Hence the whole $\widetilde{\cdot}$-correspondent of a sentence in Δ or Δ' is a fully regular formula, which can be obtained like in Theorems 3.2.2 - 4.9 of [69] from the direct regular correspondent of the original sentence and $\mathtt{AxBAC}$. In consequence,

$$\Delta_{\mathtt{reg}}+\Delta'_{\mathtt{reg}}+\mathtt{AxBAC}\quad\vdash\quad\widetilde{\Delta}+\widetilde{\Delta'}$$

hence one can also write

$$\mathtt{PbZ}_{\mathtt{v}}+\Delta_{\mathtt{reg}}+\Delta'_{\mathtt{reg}}+\mathtt{AxBAC}\quad\vdash\quad\mathtt{PbZ}_{\mathtt{v}}+\widetilde{\Delta}+\widetilde{\Delta'} \tag{3.12}$$

Because of the restrictions on the types of the variables in $\Delta_{\mathtt{reg}}$ and the $\widetilde{\Pi}$ implicitely included in $\mathtt{PbZ}_{\mathtt{v}}$, it follows that all these sentences are valid not

only in $\mathcal{S}^\omega$ but also in Bezem's type structure of all strongly majorizable functionals $\mathcal{M}^\omega$ (using that $\mathcal{M}^0 = \mathcal{S}^0$, $\mathcal{M}^1 = \mathcal{S}^1$ and $\mathcal{M}^2 \subsetneq \mathcal{S}^2$, see [69]). Since by assumption $\mathcal{M}^\omega$ is also a model for $\Delta'_{\mathtt{reg}}$, it follows that $\mathcal{M}^\omega$ is generally a model for $\mathtt{PbZ_v}+\Delta_{\mathtt{reg}}+\Delta'_{\mathtt{reg}}+\mathtt{AxBAC}$ - see also [65, 69] for an indication to the easy proof[5] of $\mathcal{M}^\omega \models \mathtt{PbZ_v}+\mathtt{AxBAC}$. Hence from (3.12) we have that (3.11) is valid in $\mathcal{M}^\omega$. Due to the restriction that the type degree of all positively universal and negatively existential quantified variables of (3.11) is at most 1 (this includes $dg(\delta) \leq 1$) and also all positively existential and negatively universal quantified variables of (3.11) have type degree at most 2 (this includes $dg(\gamma), dg(\alpha) \leq 2$) and using $\mathcal{M}^0 = \mathcal{S}^0$, $\mathcal{M}^1 = \mathcal{S}^1$ and $\mathcal{M}^2 \subsetneq \mathcal{S}^2$, in consequence also $\mathcal{S}^\omega$ is a model of (3.11), which establishes the conclusion (3.10). □

It is easy to check that the analytical axiom $\mathtt{F}^-$ can be included into the axiom set Δ' of Theorem 3.27, since $\mathtt{F}^-$ is a Δ-shape axiom and moreover $\mathcal{M}^\omega \models \mathtt{F}^-$ (see Remark 4.17 of [69]). On the other hand, the stronger axiom

$$\mathbf{F} \quad :\equiv \quad \forall \Phi^{\iota(\iota\iota)\iota} \forall x^{\iota\iota\iota} \exists y \leq_{\iota\iota\iota} x \forall k^\iota \forall z \leq_{\iota\iota} yk \, (\, \Phi kz \leq_\iota \Phi k(yk) \,) \quad (3.13)$$

is not directly of Δ shape, because of the type-ι negative universal quantifier expanded from the extensional definition of $z \leq_{\iota\iota} yk$. Nevertheless, $\mathbf{F}$ can be made into a Δ' axiom by using an $\mathtt{ncm}$-universal quantifier instead of the regular universal quantifier which causes the trouble. Let

$$\mathtt{F}^{\mathtt{nc}} \quad :\equiv \quad \forall \Phi^{\iota(\iota\iota)\iota} \forall x^{\iota\iota\iota} \exists y \leq_{\iota\iota\iota} x \forall k^\iota \forall z^{\iota\iota} \, [\, \overline{\forall} l^\iota \, (zl \leq_\iota ykl) \, \rightarrow \, \Phi kz \leq_\iota \Phi k(yk) \,]$$

be such an $\mathtt{ncm}$-variant of $\mathbf{F}$, which is easily seen to be a Δ-shape axiom. Since moreover $\mathcal{M}^\omega \models \mathbf{F}$ (this is Proposition 4.6 of [69]), which is the direct regular-quantifier translation of $\mathtt{F}^{\mathtt{nc}}$, i.e., $\mathbf{F} \equiv (\mathtt{F}^{\mathtt{nc}})_{\mathtt{reg}}$, it follows that $\mathtt{F}^{\mathtt{nc}}$ is also a Δ' axiom. Note that none of $\mathtt{F}^-$, $\mathtt{F}^{\mathtt{nc}}$ is an explicit Δ axiom (in the sense of Theorem 3.27 above) because $\mathcal{S}^\omega \not\models \mathtt{F}^-$ and also $\mathcal{S}^\omega \not\models \mathbf{F} \equiv (\mathtt{F}^{\mathtt{nc}})_{\mathtt{reg}}$ (see [69] for indications to the proofs of these).

One thus obtains, without using Luckhardt's elimination-of-extensionality procedure (in contrast to Theorem 4.9 of [69] which uses it), the following:

Corollary 3.28 (Admissibility at input of axioms $\mathtt{F}^-$ and $\mathtt{F}^{\mathtt{nc}}$) Theorem 3.27 above holds as well in the variant when the hypothesis (3.9) is replaced by

$$\mathtt{PbZ}^{\omega+} + \Delta + \mathtt{F}^- + \mathtt{F}^{\mathtt{nc}} \quad \vdash \quad \forall x^{\iota\iota} \, \forall k^\iota \, \forall y \leq_\delta sxk \, \exists z^\gamma \, A_1(x,k,y,z)$$

[5] The only novelty here, relative to the corresponding proof in [69], appears to be the inclusion of $\mathtt{AxCA}$ in $\mathtt{PbZ_v}$. But both $\mathcal{S}^\omega$ and $\mathcal{M}^\omega$ are models of $\mathtt{AxCA}$, which thus poses no problem.

3.3 Proposal for feasible Arithmetic/Analysis

We here combine PtZ and PbZ (and thus $\mathtt{IPV}^\omega$ and G_2A^ω) into a new system for feasible Arithmetic and Analysis denoted PtbZ (a short for "poly-time bounded" Z) which extends both systems and has the characteristic property that the Skolem functions for its provable Π^0_2 sentences can be majorized by poly-time computable functions. Intuitively, the new system adds to PbZ the ability to use binary integers in computations at ground type such that also PtZ gets included into PtbZ. Kohlenbach's original hierarchy $(G_nA^\omega)_{n\in\mathbb{N}}$ only uses Peano (i.e., unary) integers, and therefore the gap is big already between G_2A^ω, for which the majorants extracted by the (light) monotone Dialectica are at most integer polynomials and G_3A^ω, for which such majorants already include elementary recursive functions. The new system PtbZ attempts to fit into this gap by allowing the extraction of poly-time computable majorants, see Remark 3.29 below. The use of binary integers allows the inclusion in PtbZ of exponential functions with the exponent at most a polynomial in the binary lengths of the arguments. System PtbZ will thus characterize the class of poly-time computable majorants obtained by the (light) monotone Dialectica interpretation.

Remark 3.29 The class of poly-time computable functions $(\mathbb{N},\dots,\mathbb{N})\mapsto\mathbb{N}$ is strictly bigger than the class of polynomials $(\mathbb{N},\dots,\mathbb{N})\mapsto\mathbb{N}$ since it contains the mapping $n\mapsto n^{|n|}$, where $|n|$ is the *length* of n written in *binary*. This function grows strictly faster than any polynomial, but is poly-time computable.

System PtbZ has base types o for booleans and $\flat$ for positive integers, where $\flat$ is a superset of both ι and $\bar{\flat}$. Hence all constants of PtZ and PbZ are included in PtbZ together with their defining rewrite rules and axioms such that type $\flat$ replaces type ι in PtZ signatures and type $\bar{\flat}$ in PbZ signatures.

The only exception is the binary recursor $\mathcal{R}$ which needs to be adapted in order to carry on the value on the new argument 0. The type of $\mathcal{R}_\tau$ therefore becomes $\tau\,\tau\,(\flat\,\flat\,\tau)\,(\flat\,\tau)\,\flat\,\tau$. Because of the presence of min in our system we also find more suitable to adapt the defining rewrite rules of $\mathcal{R}_\tau$ to the more readable:

$$\begin{aligned}
\mathcal{R}_\tau\, x_0\, x_1\, y^{\flat\flat\tau}\, v\, \mathtt{0}\, \underline{w} &\hookrightarrow x_0\, \underline{w}\\
\mathcal{R}_\tau\, x_0\, x_1\, y^{\flat\flat\tau}\, v\, \mathtt{1}\, \underline{w} &\hookrightarrow x_1\, \underline{w}\\
\mathcal{R}_\tau\, x_0^\tau\, x_1^\tau\, y\, v^{\flat\tau}\, (\overset{0}{\mathtt{Su}}\, z^\flat)\, \underline{w} &\hookrightarrow \mathtt{min}_\tau\, (t_0[z]\,,\, v\, (\overset{0}{\mathtt{Su}}\, z)\, \underline{w})\\
\mathcal{R}_\tau\, x_0^\tau\, x_1^\tau\, y\, v^{\flat\tau}\, (\overset{1}{\mathtt{Su}}\, z^\flat)\, \underline{w} &\hookrightarrow \mathtt{min}_\tau\, (t_1[z]\,,\, v\, (\overset{1}{\mathtt{Su}}\, z)\, \underline{w})
\end{aligned}$$

– where we abbreviated by $\quad t_i[z] \;:\equiv\; y\,(\overset{\mathtt{i}}{\mathtt{Su}}\,z)\,(\mathcal{R}_\tau\,x\,y\,v\,z\,\underline{w})\,\underline{w} \quad$ for $i \in \{0,1\}$.

The following rewrite rules for type $\flat$ are also added:

$$\begin{array}{rcl@{\qquad}rcl}
\overset{\mathtt{0}}{\mathtt{Su}}\,\mathtt{0} & \hookrightarrow & \mathtt{0} & \overset{\mathtt{1}}{\mathtt{Su}}\,\mathtt{0} & \hookrightarrow & \mathtt{1} \\
\mathtt{Suc}\,\mathtt{0} & \hookrightarrow & \mathtt{1} & \mathtt{Suc}\,\mathtt{1} & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,\mathtt{1} \\
\mathtt{Suc}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,z) & \hookrightarrow & \overset{\mathtt{1}}{\mathtt{Su}}\,z & \mathtt{Suc}\,(\overset{\mathtt{1}}{\mathtt{Su}}\,z) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Suc}\,z)
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathtt{Pred}\,\mathtt{1} & \hookrightarrow & \mathtt{0} & \mathtt{Suc}\,(\mathtt{Pred}\,(\overset{\mathtt{i}}{\mathtt{Su}}\,z)) & \hookrightarrow & \overset{\mathtt{i}}{\mathtt{Su}}\,z \\
\mathtt{Pred}\,(\overset{\mathtt{1}}{\mathtt{Su}}\,z) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,z & \mathtt{Pred}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,z) & \hookrightarrow & \overset{\mathtt{1}}{\mathtt{Su}}\,(\mathtt{Pred}\,z)
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathtt{Pls}(x,\mathtt{1}) & \hookrightarrow & \mathtt{Suc}\,x & \mathtt{Pls}(\overset{\mathtt{1}}{\mathtt{Su}}\,x,\overset{\mathtt{1}}{\mathtt{Su}}\,y) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Suc}\,(\mathtt{Pls}\,x\,y)) \\
\mathtt{Pls}(\mathtt{1},x) & \hookrightarrow & \mathtt{Suc}\,x & \mathtt{Pls}(\overset{\mathtt{1}}{\mathtt{Su}}\,x,\overset{\mathtt{0}}{\mathtt{Su}}\,y) & \hookrightarrow & \mathtt{Suc}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Pls}\,x\,y)) \\
\mathtt{Pls}(\overset{\mathtt{0}}{\mathtt{Su}}\,x,\overset{\mathtt{0}}{\mathtt{Su}}\,y) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Pls}\,x\,y) & \mathtt{Pls}(\overset{\mathtt{0}}{\mathtt{Su}}\,x,\overset{\mathtt{1}}{\mathtt{Su}}\,y) & \hookrightarrow & \mathtt{Suc}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Pls}\,x\,y))
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathtt{Tms}(x,\mathtt{1}) & \hookrightarrow & x & \mathtt{Tms}(\overset{\mathtt{1}}{\mathtt{Su}}\,x,y) & \hookrightarrow & \mathtt{Pls}(\overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Tms}\,x\,y)\,,\,y) \\
\mathtt{Tms}(\mathtt{1},x) & \hookrightarrow & x & \mathtt{Tms}(x,\overset{\mathtt{1}}{\mathtt{Su}}\,y) & \hookrightarrow & \mathtt{Pls}(\overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Tms}\,x\,y)\,,\,x) \\
\mathtt{Tms}(\overset{\mathtt{0}}{\mathtt{Su}}\,x,y) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Tms}\,x\,y) & \mathtt{Tms}(x,\overset{\mathtt{0}}{\mathtt{Su}}\,y) & \hookrightarrow & \overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Tms}\,x\,y)
\end{array}$$

Remark 3.30 It follows that $\mathtt{Suc}\,\mathtt{0} =_\flat \mathtt{1}$ and $\mathtt{Pred}\,\mathtt{1} =_\flat \mathtt{0}$ are axioms of PtbZ. The constant Suc of PtbZ is fully equivalent with the term Suc of PtZ defined in Section 3.1. In contrast, there is a small difference between the PtbZ constant Pred and the PtZ term Pred from Definition 3.4, where we had to set $\mathtt{Pred}\,\mathtt{1} =_{\flat^-} \mathtt{1}$ (since 0 was not available there).

Convention 3.31 From here on by PtZ we understand the subsystem of PtbZ which corresponds to system PtZ defined in Section 3.1, but which includes constants 0, Pls and Tms as above and $\mathcal{R}$ is defined as above. These additions do not alter[6] the properties and results involving system PtZ from Section 3.1.

Remark 3.32 Notice that the inclusion in PtbZ of rewrite rules like

$$\overset{\mathtt{0}}{\mathtt{Su}}\,(\mathtt{Suc}\,z) \;\hookrightarrow\; \mathtt{Suc}\,(\mathtt{Suc}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,z)) \qquad \overset{\mathtt{1}}{\mathtt{Su}}\,(\mathtt{Suc}\,z) \;\hookrightarrow\; \mathtt{Suc}\,(\mathtt{Suc}\,(\mathtt{Suc}\,(\overset{\mathtt{0}}{\mathtt{Su}}\,z)))$$

[6] This is because Pls and Tms denote poly-time computable functions and therefore can be expressed in the original PtZ. However this can be done only by means of the binary recursor $\mathcal{R}$, fact which we want to avoid in view of the subsequent development.

would allow two normal forms for certain closed terms of type $\flat$: a *unary* form in terms of $\mathtt{Suc}$, $\mathtt{O}$ and a binary form in terms of $\overset{0}{\mathtt{Su}}$, $\overset{1}{\mathtt{Su}}$, $\mathtt{O}$, $\mathtt{1}$. Our interest is to have a unique, binary normal form for all closed terms of type $\flat$, hence such rules are banned from our system. Nevertheless,

$$\overset{0}{\mathtt{Su}}\,(\mathtt{Suc}\,z) \;=_{\flat}\; \mathtt{Suc}\,(\mathtt{Suc}\,(\overset{0}{\mathtt{Su}}\,z)) \qquad \overset{1}{\mathtt{Su}}\,(\mathtt{Suc}\,z) \;=_{\flat}\; \mathtt{Suc}\,(\mathtt{Suc}\,(\mathtt{Suc}\,(\overset{0}{\mathtt{Su}}\,z)))$$

are immediately seen to be theorems of $\mathtt{PtbZ}$ (due to the reverse rewriting).

Definition 3.33 Let $\mathtt{Tm}^{-}(\mathtt{PtbZ})$ denote the set of all terms of $\mathtt{PtbZ}$ which do not contain $\mathtt{Max}$, $\mathtt{Sum}$, μ_b, $\widetilde{\mathtt{R}}$, $\mathtt{Pred}$, $\mathtt{min}$, $\mathtt{Max}$ and also not the binary recursor $\mathcal{R}$. We also denote by $\mathtt{Tm}^{-}(\mathtt{PtZ})$ the set of all terms of $\mathtt{PtZ}$ built without $\mathcal{R}$.

Proposition 3.34 To each term $t[x_1,\ldots,x_k]$ of $\mathtt{PtbZ}$ (with $x_1,\ldots,x_k$ all free variables of t) one can syntactically associate a term $t^{\star}[x_1,\ldots,x_k] \in \mathtt{Tm}^{-}(\mathtt{PtbZ})$ (with $x_1,\ldots,x_k$ all free variables of $t^{\star}$) such that:

$$\vdash\; x_1^{\star} \succeq x_1 \to \ldots \to x_k^{\star} \succeq x_k \to t^{\star}[x_1^{\star},\ldots,x_k^{\star}] \succeq t[x_1,\ldots,x_k] \qquad (3.14)$$

Proof: Follows immediately from Proposition 3.15 once we provide majorizing terms in $\mathtt{Tm}^{-}(\mathtt{PtbZ})$ for the $\mathtt{PtZ}$ arithmetic constants (see Definition 1.63) included in $\mathtt{PtbZ}$. Thus $\overset{0}{\mathtt{Su}}$, $\overset{1}{\mathtt{Su}}$, $\mathtt{Pad}$, and $\mathtt{Sma}$ are monotone increasing on $\flat$ and therefore they majorize themselves. Constants $\mathtt{Hal}$ and $\mathtt{Cho}$ are majorized by the identity on $\flat$, respectively the left projection $\flat \times \flat \mapsto \flat$. The selector $\mathtt{If}_{\tau}$ with $\tau \equiv \tau_1 \ldots \tau_n \flat$ is majorized by $\lambda p^{o}, y^{\tau}, z^{\tau}, x_1^{\tau_1}, \ldots, x_n^{\tau_n}.\, \mathtt{Pls}^{\flat}(y\underline{x})(z\underline{x})$, where $\underline{x} \equiv x_1,\ldots,x_n$. At last, the binary recursor $\mathcal{R}_{\tau}$ is majorized by $\lambda x, y, v, z.\, \overset{1}{\mathtt{Su}}(vz)$. $\square$

Proposition 3.35 Let $t \in \mathtt{Tm}^{-}(\mathtt{PtbZ})$ be a term of type ι such that all its free variables are $x_1^{\sigma_1},\ldots,x_k^{\sigma_k}$ with $dg(\sigma_i) \leq 1$ for $i \in \overline{1,k}$. Then the $\beta\eta$ normal form $\hat{t}[x_1,\ldots,x_k]$ of t is an element of $\mathtt{Tm}^{-}(\mathtt{PtbZ})$ built without λ-abstraction.

Proof: Identical to the proof of the corresponding Proposition 3.17 since also the term constants allowed in $\mathtt{Tm}^{-}(\mathtt{PtbZ})$ have type degree at most 1. $\square$

Remark 3.36 In fact, according to Convention 3.31, $\mathtt{Tm}^{-}(\mathtt{PtbZ}) \equiv \mathtt{Tm}^{-}(\mathtt{PtZ})$, hence such terms denote poly-time computable functions. On the other hand not all $\mathtt{PtZ}$ terms are comprised among $\mathtt{Tm}^{-}(\mathtt{PtbZ})$, since, e.g., the binary recursor $\mathcal{R}_{\tau}$ is not comprised in $\mathtt{Tm}^{-}(\mathtt{PtbZ})$. Also $\mathtt{Tm}^{-}(\mathtt{PtbZ})$ is strictly bigger than

$\mathtt{Tm}^{-}(\mathtt{PbZ})$ because $\mathtt{Sma}$ cannot be majorized by any term of $\mathtt{Tm}^{-}(\mathtt{PbZ})$, see also Remark 3.29. Otherwise, all terms of $\mathtt{Tm}^{-}(\mathtt{PtbZ})$ which do not contain $\mathtt{Sma}$ can, in fact, be majorized by terms of $\mathtt{Tm}^{-}(\mathtt{PbZ})$.

Theorem 3.37 (Poly-time bound synthesis by monotone Dialectica) Theorems 3.23, 3.27 and Corollaries 3.24 3.25 hold also when $\mathtt{PbZ}$ is replaced by $\mathtt{PtbZ}$ in the premise and correspondingly in the conclusion the demand for a syntactic polynomial $\overline{p}$ is replaced by a demand for a syntactic poly-time computable term of $\mathtt{Tm}^{-}(\mathtt{PtZ})$.

Proof: Immediate in view of Propositions 3.34 and 3.35 and Remark 3.36. □

Discussion

This chapter has developed the theory of the extraction of feasible, poly-time computable programs from proofs by means of our adaptations of Gödel's Dialectica and of the Monotone Dialectica Interpretation. For this purpose we combined two pre-existent and rather incomparable frameworks. The crossbreeding of Ferreira's base theory for feasible Analysis and Kohlenbach's polynomially bounded Analysis is exposed in the Natural Deduction framework developed earlier in Chapter 1, including the $\mathtt{ncm}$ quantifiers. Whereas Kohlenbach developed many concrete applications of the Monotone Dialectica extraction of polynomial bounds, we have not yet found a situation where the extraction of a poly-time computable program would be possible, in contrast to the mere polynomial bound synthesis. In theory such situations are nonetheless quite possible and the development of concrete examples is not to be excluded. We thus hope to have justified the novelty of our contribution to the extraction of feasible programs from (classical) proofs by means of proof interpretations.

Chapter 4

Comparison with other techniques for extraction of exact realizers. Case Studies

[1] The proof-theoretic techniques for program extraction from (classical) proofs can be viewed as either based on cut elimination or on proof interpretations. In contrast to the hyper-exponential worst-case complexity of cut elimination, proof interpretations provide practically feasible extraction algorithms [56], mainly because of their modularity. The normalization of terms (programs) extracted by proof interpretations would nevertheless equate the worst-case complexity of the two aforementioned classes of techniques. The separation of program extraction from cut elimination (normalization) by means of proof interpretations appears nonetheless to be more useful in practice since the normalization of extracted programs is actually not needed in important applications (see [63]). Also various optimizations of normalization can be performed during the extraction process, see [53] for such an example.

Amidst proof interpretations, Kreisel's modified realizability [85] and Gödel's functional *Dialectica* interpretation (D-interpretation) [45, 4] have a prominent place. The two differ only with respect to the treatment of logical implication. In contrast to modified realizability, Gödel's technique takes counterexamples into account and therefore directly extends to classical (arithmetical) proofs via some negative translation. The extension goes further to (classical) analytical proofs [68]. Remarkable new theorems in Numerical Analysis

[1] This chapter is based on [50], but the text is extended and largely rephrased in view of the new developments theresince, reported in [51] and the previous chapters here.

have recently been obtained by means of (Kohlenbach's monotone variant of) functional interpretation (see [81, 63] for surveys of such results). Unlike the D-interpretation, modified realizability requires an intermediate so-called Friedman-Dragalin "A-translation" after a negative translation and the combination of the three interpretations is only known to handle classical proofs of Π^0_2-formulas. Various refinements of the latter combined proof interpretation were produced in recent years [10, 23]. One purpose of such refinements was the simplification of the translation in terms of complexity of the extracted programs. On the other hand the extension of the class of proofs at input was attempted.

4.1 The BBS refined A-translation

The Berger-Buchholz-Schwichtenberg (BBS for short) refined interpretation [10] succeeds in both directions. It can directly handle classical proofs of formulas in a class which extends Π^0_2. The type complexity of the extracted programs is significantly reduced in comparison with the unrefined combination of the three aforementioned translations. The BBS refined A-translation [10] was implemented in the proof system MinLog [116] and has been successfully used to extract programs from classical proofs (see, e.g., [15]). However, this technique cannot directly handle all proofs of Π^0_2-formulas in the semi-classical arithmetical system Z of Berger, Buchholz and Schwichtenberg [10], as we explain in the sequel.

Definition 4.1 Let $\mathtt{Z}^\exists$ be the extensional variant of system $\mathtt{WeZ}^\exists$ from Chapter 1, obtained by adding to it the compatibility axiom $x =_\sigma y \to B(x) \to B(y)$ or the full extensionality axiom $\forall z^{\sigma\tau}, x^\sigma, y^\sigma . x =_\sigma y \to zx =_\tau zy$. Also let $\mathtt{Z}^\exists_0$ denote the *minimal logic* variant of $\mathtt{Z}^\exists$ which includes only AxFLS, the so-called *boolean* ex-falso-quodlibet schema $\mathtt{at}(\mathtt{ff}) \to A$ but not the full, so-called *logical* ex-falso $\bot \to A$. The original BBS systems Z and $\mathtt{Z}_0$ from [10] can be obtained by eliminating the strong existential quantifier $\exists\, \mathtt{Z}^\exists$, respectively $\mathtt{Z}^\exists_0$. Obviously, systems $\mathtt{Z}^\exists$ and Z can be obtained by simply adding an axiom $\bot \to \mathtt{at}(\mathtt{ff})$ to $\mathtt{Z}^\exists_0$, respectively $\mathtt{Z}_0$.

The BBS refined A-translation fails to directly interpret (in the sense of Theorem 3.3 of [10]) the axiom $\bot \to \mathtt{at}(\mathtt{ff})$, hence Theorem 3.3 of [10] would no longer hold if $\mathtt{Z}_0$ were replaced by Z as input system. On the other hand, the BBS technique can directly extract programs from (minimal arithmetical)

$\mathtt{Z_0}$ proofs of formulas in a class which extends Π^0_2 by means of so-called *definite* and *goal* formulas. Namely from proofs $\mathtt{Z_0} \vdash \overrightarrow{D} \to \forall \underline{y}(\overrightarrow{G} \to \bot) \to \bot$, where $\overrightarrow{D}$ are definite and $\overrightarrow{G}$ are goal formulas (by their definition in [10], $\overrightarrow{D}$ and $\overrightarrow{G}$ do not contain the strong, intuitionistic, $\exists$), the `BBS` refined `A`-translation syntactically produces a proof $\mathtt{Z} \vdash \overrightarrow{D} \to \exists \underline{y} \overrightarrow{G}$. From this *intuitionistic* proof exact realizers $\underline{t}$ for $\exists \underline{y}$ can be produced by means of Kreisel's modified realizability which yields syntactically a verifying proof $\mathtt{Z} \vdash \overrightarrow{D} \to \overrightarrow{G}[\underline{y} \leftarrow \underline{t}]$, also using the fact that $\overrightarrow{D}$ and $\overrightarrow{G}$ are $\exists$-free.

An extension of this result is indicated in [9]. Both the strong $\exists$ and the `ncm` quantifiers can be allowed in the proof translations mentioned above. They can also be included in an extension of the definition of definite and goal formulas so that from proofs $\mathtt{Z}_0^{\exists} \vdash \overrightarrow{D} \to \forall \underline{y}(\overrightarrow{G} \to \bot) \to \bot$ the extended `BBS` refined `A`-translation syntactically produces proofs $\mathtt{Z}^{\exists} \vdash \overrightarrow{D} \to \exists \underline{y} \overrightarrow{G}$. Even with the new definition, definite formulas D are $\exists$-negative and goal formulas G are $\exists$-positive. Therefore, from the latter intuitionistic proof realizing terms $\underline{t}$ can be read by modified realizability and a final verifying proof $\mathtt{Z}^{\exists} \vdash \overrightarrow{D} \to \overrightarrow{G}[\underline{y} \leftarrow \underline{t}]$ can be syntactically produced.

Since the input proof must be in minimal logic, not all classical proofs of Π^0_2 formulas can be directly treated by this technique. Despite the inclusion of the strong $\exists$ in parts of the proof at input, by this extraction method the existential quantifiers to be realized are *weak* (classical, $\neg\forall\neg$, usually denoted $\exists^{cl}$, see [111]). Since modified realizability gives a simpler, more direct treatment when definite and goal formulas do not include the strong $\exists$ and also because full stability $\neg\neg A \to A$ is provable in `Z` for $\exists$-free formulas A, it appears practically useful to first eliminate the use of $\exists$ in the fully classical proofs (which use stability for formulas which include the strong $\exists$). This can be achieved by simply replacing everywhere in the proof $\exists$ with $\exists^{cl}$. Since after this replacement $\mathtt{Ax}\exists^{+}$ and $\mathtt{Ax}\exists^{-}$ become provable in `Z` it follows that all fully classical proofs can actually be expressed in `Z`. However, the use of $\bot \to \mathtt{at}(\mathtt{ff})$ is essential in the proof of $\exists^{cl}$ elimination and therefore not all classical proofs can be expressed in $\mathtt{Z_0}$.

The situation can be repaired by using a preprocessing *reduced* negative translation which is exposed in [111]. This is based on the Gödel-Gentzen negative translation of `Z` into $\mathtt{Z_0}$ which simply double negates all prime formulas (see [111]). Since negations contribute negatively to the complexity of extracted programs (by the increase of the type degree), it appears important to reduce as many as possible such negations from the negative-translated

formula. Such an optimization is possible, e.g., because triple negations may appear and these can be replaced by simple negations. In the end one obtains that, after such a negative translation preprocessing, the BBS refined A-translation can handle all $\mathtt{PA}^\omega$ proofs of a class of formulas which extends Π^0_2. See also [117] for an extension of the BBS refined A-translation to formulas containing inductive definitions and also to extractions using so-called *external* realizers for part of the lemmas used in the input proof.

4.1.1 Theoretical comparison with the BBS technique

In contrast to the BBS refined A-translation, the (light) (M)D-interpretation directly interprets all WeZ-proofs $\mathcal{P}$ in the sense that it algorithmically synthesizes a realizer (or majorant) for the existential quantifiers of the translated conclusion formula of $\mathcal{P}$.

System Z may be viewed as the Natural Deduction formulation of some restriction of $\mathtt{PA}^\omega$ to the language without the strong $\exists$. It can also be viewed as a superset of $\mathtt{HA}^\omega$ without the strong $\exists$. The design of system Z and of functional interpretation in Natural Deduction style [51] trigger the elimination of a preprocessing negative translation in the interpretation of this subsystem of $\mathtt{PA}^\omega$. The treatment of full $\mathtt{PA}^\omega$ under FI nevertheless requires a negative translation preprocessing, just like BBS. The (rather big) difference is that FI handles[2] all proofs in (suitable versions of) $\mathtt{PA}^\omega$ via this preprocessing.

In conclusion, all proofs which BBS handles directly, FI handles directly as well. On the other hand, there exist proofs which FI handles directly but BBS handles only via a negative translation.

4.2 Berger's *hsh* example

An example of such is the proof of the statement that if $h, s : \mathbb{N} \mapsto \mathbb{N}$ are two functions over the set of natural numbers and s has only strictly positive values then $h \circ s \circ h$ cannot be the identity (this case-study suggested

[2]In the sense that realizing terms can be produced for the negative translation of the conclusion of the (arbitrary) $\mathtt{PA}^\omega$ proof at input. This is the basis for the full modularity feature of FI-based techniques which contrasts with the more limited modularity of the techniques based on modified realizability (including the BBS technique). Here *modularity* means the ability of making unrestricted use of Lemmas in the proof at input, see also [56]. The techniques based on modified realizability can only use those Lemmas for which a realizer can be manually (external) if not machine (automated, internal) provided [117].

by Berger got to be called *hsh*). More formally,

$$\forall s, h\,[\overbrace{(\forall m\; s(m) \neq 0)}^{A} \longrightarrow \exists n\; h(s(h(n))) \neq n\,] \tag{4.1}$$

We implemented a `FI`-extraction module in the proof system `MinLog`. We compared the machine performance of the two program-extraction algorithms on the *hsh* example. The solution found by machine via both `BBS` and `FI` is unexpectedly clever. It relies on the fact that the conjunction of the following three clauses cannot hold in the presence of the implicative assumption of (4.1), which we denoted by A :

$$hs\,\underbrace{h(shh0)} = shh0 \tag{4.2}$$

$$hsh(h0) = h0 \tag{4.3}$$

$$hsh(0) = 0 \tag{4.4}$$

More exactly, the following hold:

$$(4.2) \wedge (4.3) \implies hsh0 = shh0 \tag{4.5}$$

$$(4.4) \wedge (4.5) \implies shh0 = 0 \tag{4.6}$$

$$(4.6) \wedge A \implies \bot \tag{4.7}$$

We conclude that $(4.2) \wedge (4.3) \wedge (4.4) \wedge A \Longrightarrow \bot$, hence

$$A \implies \neg(4.2) \vee \neg(4.3) \vee \neg(4.4) \quad .$$

In contrast to `BBS`, `FI` directly provides the additional term $h(h(0))$ which realizes the $\forall m$ of (4.1). This gives more information on the reasoning process employed for the extraction of the realizer for $\exists n$. Only the instance $m := h(h(0))$ is needed in the verifying proof. Notice that $s(h(h(0))) = 0$ of (4.6) is just a counterexample to A.

Since the `FI` algorithm is already known to have cubic time complexity (see [56] for a detailed proof of this), it would be interesting to investigate the computational complexity of `BBS` in order to have a quantitative comparison as well between the two extraction techniques.

4.2.1 `MinLog` source code for Berger's *hsh* example

```
(load "$MINLOGPATH/init.scm")
```

```
(mload "../lib/nat.scm")
(add-var-name "f" "g" "h" "s" (py "nat=>nat"))
(add-program-constant "c"
          (py "(nat=>nat)=>(nat=>nat)=>nat=>nat") 1)
(add-computation-rule (pt "c g f n") (pt "g(f n)"))
(add-global-assumption "Compat"
        (pf "allnc f,n,m.n=m -> f n=f m"))

(set-goal
   (pf "all f,g.all m excl n g(f n)=m -> all m excl n g n=m"))
(search)
(save "Surj-Lemma")

(set-goal (pf "all f,g.(all n,m.g(f n)=g(f m) -> n=m)
                            ->  all n,m.f n=f m -> n=m"))
(strip)
(use 1)
(use "Compat")
(use 2)
(save "Inj-Lemma")

(set-goal (pf "all f,g.(all m excl n g(f n)=m) ->
                             (all n,m.g n=g m -> n=m) ->
                                 all m excl n f n=m"))
(search)
(save "Surj-Inj-Lemma")

;;; We now prove the hsh-Theorem
(set-goal
   (pf "all s,h.(all n.s n=0 -> bot)
                   ->  all n h(s(h n))=n -> bot"))
(assume "s" "h" "s-not-0" "hsh-is-id")
(cut (pf "all m excl n.h(s(h n))=m"))
(assume "hsh-surj")
(cut (pf "all m excl n.h(s n)=m"))
(assume "hs-surj")
(cut (pf "all n,m.h(s(h n))=h(s(h m)) -> n=m"))
```

```
(assume "hsh-inj")
(cut (pf "all n,m.s(h n)=s(h m) -> n=m"))
(assume "sh-inj")
(cut (pf "all n,m.h n=h m -> n=m"))
(assume "h-inj")
(cut (pf "all m excl n s n=m"))
(assume "s-surj")
(use-with "s-surj" (pt "0") "s-not-0")
(use "Surj-Inj-Lemma" (pt "h"))
(use "hs-surj")
(use "h-inj")
(use "Inj-Lemma" (pt "s"))
(use "sh-inj")
(use-with "Inj-Lemma" (pt "c s h") (pt "h") "?")
(use "hsh-inj")
(assume "n" "m")
(inst-with-to "hsh-is-id" (pt "n") "hshn-is-n")
(simp "hshn-is-n")
(inst-with-to "hsh-is-id" (pt "m") "hshn-is-m")
(simp "hshn-is-m")
(prop)
(use-with "Surj-Lemma" (pt "h") (pt "c h s") "?")
(use "hsh-surj")
(assume "m" "m-not-hsh-value")
(use "m-not-hsh-value" (pt "m"))
(use "hsh-is-id")
(save "hsh-Theorem")

(define hsh-proof (theorem-name-to-proof "hsh-Theorem"))
(mload "../FI/fiets.scm")
(define vatmp (time (FI-extracted-vatmpair hsh-proof)))
(define FI-tmtup (tmpair-to-tuple
                       (vatmpair-to-tmpair vatmp)))
(define FI-tmlst (tmtuple-to-tmlist FI-tmtup))
(define n1 (car  FI-tmlst))
(define n2 (cadr  FI-tmlst))
(define n-n1 (time (nt n1)))
```

```
(define n-n2 (time (nt n2)))
(term-to-string n-n1)
; "[f0,f1]f1(f1 0)"
; With renaming (f0 -> s and f1 -> h)
; "[s,h] h(h 0)"

(term-to-string n-n2)
;"[f0,f1][if (f1(f0(f1 0))=0)
;           [if (f1(f0(f1(f0(f1(f1 0)))))=f0(f1(f1 0))) (f1 0)
;      (f0(f1(f1 0)))]
;    0]"
; With renaming (f0 -> s and f1 -> h) and indentation
;"[s,h][if (h(s(h 0))=0)
;           [if (h(s(h(s(h(h 0)))))=s(h(h 0))) (h 0)
;      (s(h(h 0)))]
;    0]"

(define ggn-proof
     (proof-to-reduced-goedel-gentzen-translation hsh-proof))
(define min-excl-proof  (expand-thm gg-proof "Surj-Lemma"))
(mload "../modules/atr.scm")
(define et
   (atr-min-excl-proof-to-structured-extracted-term
                                 min-excl-proof))
; (term-to-string (nt et))
; "[f0,f1][if (f1(f0(f1(f1 0)))=f1 0)
;           [if (f1(f0(f1(f0(f1(f1 0)))))=f0(f1(f1 0))) 0
;                              (f0(f1(f1 0)))]
;                  (f1 0)]"

; With renaming (f0 -> s and f1 -> h) and indentation
; [s,h][if (h(s(h(h 0)))=h 0)
;          [if (h(s(h(s(h(h 0)))))=s(h(h 0)))
;              0
;              (s(h(h 0)))]
;          (h 0)]
```

4.3 The (semi-)classical Fibonacci proof

[3] We here demonstrate program extraction by the Light Dialectica interpretation on a minimal logic proof of the classical existence of Fibonacci numbers. This semi-classical proof is available in MinLog's library of examples. The term of Gödel's **T** extracted by the LD-interpretation is, after strong normalization, exactly the usual recursive algorithm which defines the Fibonacci numbers (in pairs). This outcome of the Light Dialectica meta-algorithm is much better than the **T**-program extracted by means of the pure Gödel Dialectica interpretation. This situation holds both for the original input classical Fibonacci proof from [10] (human built) and for a "distorted" variant of this proof which is generated by the unrestricted automated proof search of MinLog (see Footnote 14 in the sequel). On the other hand, a different outcome is yielded by the refined A-translation technique of Berger, Buchholz and Schwichtenberg on the two classical Fibonacci proofs. Thus the program obtained by Light Dialectica is strictly less complex than the result obtained by means of the BBS refined A-translation on the artificially distorted variant of the input proof, but otherwise it is identical with the term yielded by Berger's Kripke-style refined A-translation. Although syntactically different, it also has the same computational complexity as the original program yielded by the refined A-translation from the undistorted input classical Fibonacci proof.

4.3.1 Motivation for treatment of Fibonacci in MinLog

There has been quite some work in the last years in the field of program extraction from *classical* proofs. Although strong mathematical results have recently been obtained in the Proof Mining of classical analytical proofs (see, e.g., [78, 63, 81, 89, 97]), the computer-implemented program extraction meta-algorithms were able to produce only limited results, for rather small test-cases and even then, the extracted program is not the optimal one.

Such a situation one partly encounters in the extraction of a rather unusual, distorted algorithm for the computation of Fibonacci numbers by means of the Berger-Buchholz-Schwichtenberg (BBS) refined A-translation of [10]. The term $\mathtt{t}_{\mathtt{BBS}}$ of Gödel's **T** extracted via this BBS refined A-translation from an artificially distorted[4] variant of the MinLog minimal logic proof of the *weak*

[3] This section is mainly based on [54], from which only the exposition of Light Dialectica was excerpted.

[4] This artificial distortion is due to the automated proof-search mechanism of MinLog,

(classical) existence of the Fibonacci numbers, followed by Kreisel's Modified Realizability [85] and finally strongly normalized [11, 12] not only makes necessarily use of a type-2 Gödel recursor (present also in the original extraction from [10]), but also uses two times the corresponding type-2 functional, fact which strictly increases its computational complexity. The program $\mathtt{t}_{\mathtt{BBS}}$ has an unexpected exponential time complexity, see the end of Section 4.3.4. On the other hand, the program extracted by the BBS technique from the original classical Fibonacci proof outlined in [10] is nonetheless linear-time in the unary representation of natural numbers, see [10] for full technical details.

The aforementioned type-2 recursor is $\mathtt{R}_{(\iota\to\iota\to\iota)\to\iota}$, where the type level (degree) of $(\iota \to \iota \to \iota) \to \iota$ is 2. Here ι is the base type which denotes the set of natural numbers $\mathbb{N}$ and $\mathtt{R}_\rho$ is the denotation for the so-called "type-ρ Gödel recursor", which actually has the type $\rho \to (\iota \to \rho \to \rho) \to \iota \to \rho$ in Gödel's $\mathbf{T}$[5]. This situation is quite unexpected since the usual recursive definition of Fibonacci numbers (in pairs) can be expressed in Gödel's $\mathbf{T}$ by means of a type-0 Gödel recursor only, namely $\mathtt{R}_{\iota\times\iota}$. Here $\sigma \times \tau$ denotes the pairing of types σ and τ. In fact such a $\mathbf{T}$-term was actually extracted in MinLog [116] by pure Modified Realizability, from the usual pure intuitionistic proof of the *strong* (intuitionistic) existence of Fibonacci numbers, see [10][6].

The point of the endeavour of extracting programs from classical rather than constructive or even purely intuitionistic proofs is that (semi-)classical proofs are much easier and more direct to build, both by human brain and also in the various computer-implemented proof-systems. It is therefore desirable that the algorithms synthesized from classical proofs by means of the more complex program extraction meta-algorithms[7] are at least as good as those yielded by the more common extraction techniques[8] from the corresponding constructive/purely intuitionistic proofs. When applied to the semi-classical MinLog Fibonacci proof (by this we hereon mean the distorted variant obtained by automated proof-search, present in [116, 49], and not the manual one, originally introduced in [10]), this is not the case, neither for the BBS refined

relative to the more manually given input proof which was originally used in the classical Fibonacci extraction reported in [10].

[5]See paper [10] for more such technical details.

[6]On the other hand, this linear - in the unary representation of natural numbers - algorithm is outperformed by other logarithmic algorithms, see [111] for such an example.

[7]Here we think particularly (but not exclusively) at those from the Dialectica family (see [98] for a nice unification work) and the Refined A-translation family.

[8]Basically variants of Kreisel's Modified Realizability [85], which is a simpler but weaker form of Gödel's functional (*Dialectica*) interpretation [4, 45].

A-translation, nor for the pure Gödel Dialectica Interpretation, as we show later in the sequel. A repair of this situation can be provided for the BBS refined A-translation by eliminating the distortion from the proof at input[9] or by using its Kripke-style variant due to Berger in [9][10]. The latter extraction technique actually produces exactly the same program as our Light Dialectica interpretation (originally introduced in [51], but see also Section 2.1 here for a much larger and more unified exposition).

On the other hand, none of the *monotone* [68] or *bounded* [33] optimizations of Gödel's technique can handle such an exact realizer extraction problem (in an efficient manner). It is the Light Dialectica interpretation (abbreviated LD-interpretation) which gives the solution. The term of Gödel's **T** extracted by the LD-interpretation is, after strong normalization, exactly the usual recursive algorithm which defines the Fibonacci numbers (in pairs).

4.3.2 The semi-classical Fibonacci proof in MinLog

MinLog is an interactive proof- and program-extraction system developed by H. Schwichtenberg and members of the logic group at the University of Munich. It is based on first order Natural Deduction calculus and uses as primitive *minimal* rather than classical or intuitionistic logic. See [49, 116] for full details.

Definition 4.2 (Fibonacci Numbers) The inductive definition is as usual

$$\texttt{Base}: \; F_0 :\equiv 0, \; F_1 :\equiv 1 \qquad \texttt{Step}: \; F_{n+1} :\equiv F_n + F_{n-1} \;\text{ for }\; n \geq 1,\, n \in \mathbb{N}$$

The Fibonacci Numbers example was implemented in MinLog and it was comparatively analysed in [10] by both pure Modified Realizability (from the usual pure intuitionistic proof) and also by the BBS refined A-translation (from a minimal logic proof of the weak, classical existence of Fibonacci Numbers; we dub such proofs as "semi-classical") followed by Modified Realizability.

[9] See [10, 111] or the end of Section 4.3.4 for a display of the original program that is obtained by BBS from the undistorted classical Fibonacci proof, which is also of linear-time complexity in the unary representation of the natural-number input. See also Footnote 14.

[10] Berger's *Kripke-style* refined A-translation introduced in [9] nicely combines the optimizing (in the sense of the efficiency of programs extracted from classical proofs) features of both the BBS [10] and the Coquand-Hofmann [23] refined A-translations. It also furthermore adds the so-called *uniform* quantifiers, which are used to "label" and thus isolate parts of the input proof which are meant not to have a computational content under such a translation.

The semi-classical Fibonacci proof in MinLog is a Natural Deduction proof of $\forall n \exists^{\mathrm{d}} k\ G(n,k)$ – where $\exists^{\mathrm{d}} k\ G(n,k) :\equiv (\forall k.\, G(n,k) \to \bot) \to \bot$ – from assumptions expressing that G is the graph of the Fibonacci function, i.e.,

$$G(0,0)\ \ \mathtt{AND}\ \ G(1,1)\ \ \mathtt{AND}\ \ \forall n,k,l.\,[G(n,k) \land G(n+1,l)] \to G(n+2,k+l)\,.$$

The best source for reading and analysing this proof is the MinLog distribution [49] (or [116]), particularly that this differs, due to the use of automated proof-search, from the more manually given proof from Section 6 of [10]. See also Footnote 14 for some hints on how these semi-classical proofs can be constructed in MinLog. Notice that in the context of program extraction by the Light Dialectica interpretation (shortly presented in Section 4.3.3 below) the assumption on G is rather expressed as

$$G(0,0)\ \ \mathtt{AND}\ \ G(1,1)\ \ \mathtt{AND}\ \ \overline{\forall} n,k,l.\,[G(n,k) \land G(n+1,l)] \to G(n+2,k+l)\ ,$$

where $\overline{\forall}$ is the universal quantifier without computational meaning, see below.

4.3.3 The light functional "Dialectica" interpretation

The "light" variant of Gödel's functional "Dialectica" Interpretation was introduced in [51] as an optimization for term-extraction of Gödel's original technique[11] from [45]. The main feature of "Dialectica Light" is the elimination *already at extraction time* of a number of *relevant* (for the Dialectica program extraction) Contractions which are identified as *redundant* and in consequence are isolated by means of an adaptation of Berger's quantifiers without computational content[12] (introduced in [9] as "uniform quantifiers"). See Section 2.1 for full details on Light Dialectica.

Remark 4.3 Recall that Gödel's functional "Dialectica" interpretation becomes relatively (far) more complicated at the moment when it has to face computationally relevant *contraction*. In the Natural Deduction setting, contraction amounts to the discharging of at least two copies (from the same parcel[13]) of an open assumption formula A during an Implication Introduction $\dfrac{[A]\ \ldots\ /B}{A \to B}$. This is because, for the so-called "Dialectica-relevant" contractions (see Definition 1.18), formula A becomes part of the (raw, i.e., not

[11] Paper [4] provides a nice survey in English which includes the extensions to full Analysis.

[12] In [51] we named these special existential and universal quantifiers "without (or non-) computational meaning", abbreviated ncm. We here continue to use our own terminology.

[13] In the sense of the terminology from [43]. This is the same notion as "assumption variable" in [111].

yet normalized) realizing term. Therefore, the *a priori* (i.e., already at the extraction stage) elimination of some of these D-relevant contractions, rather than *a posteriori* (i.e., during the subsequent strong normalization process), represents an important complexity improvement of the extracted program. We exemplify our statement in the following Section 4.3.4.

4.3.4 A comparison of the three extraction techniques

It can be immediately seen, also from the machine benchmarks below, that the program yielded by the Light Dialectica interpretation clearly outperforms the algorithm given by the BBS refined A-translation[14]. The latter is at its turn much more efficient than the term extracted by means of the pure Gödel Dialectica interpretation, which contains an important quantity of redundant information. All three extracted (by the three program-synthesis techniques) terms are presented below in a human-processed adaptation of the raw MinLog output. See [49] for the pure machine-extracted programs. We stress the fact that the outcomes of the pure and the light Dialectica meta-algorithms would remain the same even if the input classical Fibonacci proof would be the original, undistorted one from [10]. Only the output of the BBS A-translation would get better when using its original input, see Footnote 14. Our point here is that if the user is unable or unwilling to optimize the input proof, then it is the responsibility of the extraction technique to deal with such practically very possible artificial situations and overcome the complexity loss.

It appears that the BBS refined A-translation is more directly dependent on the shape of the input proof and hence its performance decreases with the artificial distortions. This is because the BBS interpretation is based on an initial proof translation which literally includes the translation of the distortions. The witness is subsequently literally read from such a translated proof by Modified Realizability, which cannot avoid to preserve the distortions. In the case of the distorted classical Fibonacci proof, the redundant use twice of the (basically the induction hypothesis) assumption $\exists^{\mathrm{cl}} k, l.\, G(n,k) \wedge G(n+1,l)$ during the automated search for a proof of the induction step will yield the double appearance of the type-2 functional H in the BBS-extracted program,

[14] This situation holds only for the more artificial input proof. When its distortions are eliminated by using a (search 1) restricted proof-search command instead of just (search), the run-time performance of the two extracted programs is quite equal. Here (search 1) means that the open assumptions may only be used at most once in the wanted searched proof. See [111] and the MinLog manual [116] for more details.

see it below at 2). On the contrary, for both the D-interpretation and the LD-interpretation, the artificial distortion is harmless w.r.t. already the raw extracted program. Only a purely logical contraction, irrelevant already for the pure Dialectica, over the open assumption $\exists^{\mathtt{d}} k, l.\, G(n,k) \wedge G(n+1,l)$ will occur. This contraction has no computational content anyway, already in the case of the D-interpretation, because its formula translation has an empty universal side, see Definitions 2.1 and 1.18. For the LD-interpretation the situation is identical, without any use of the special quantifiers without computational meaning. In fact not only this extra contraction, but the whole redundant proof-branch produced by the artificial distortion is without computational content under both the D-interpretation and the LD-interpretation. This is why the raw programs extracted by the two techniques are unchanged by the redundant distortion in the proof at input, i.e., regardless of whether the afore-mentioned assumption had been used once or twice, etc. Such an invariant situation was not possible for the BBS refined A-translation because this extraction technique lacks the full modularity of the Dialectica interpretations (see also [50] for an extended comment on this issue) and is more proof-dependent (as explained above).

We now attempt a theoretical explanation of why the program extracted by the LD-interpretation outperforms so neatly the one given by the pure D-interpretation. As hinted by Remark 4.3, the difference in performance is yielded by (the elimination of) a computationally D-redundant contraction. This contraction is given by the fact that the assumption $u : \forall n, k, l.\, [G(n,k) \wedge G(n+1,l)] \rightarrow G(n+2, k+l)$ is open in the proof of the induction step of the classical Fibonacci proof. The contraction is inserted in the proof to be mined independently of the number of open occurrences of u in the original proof at input. The mechanism of the Dialectica interpretation in Natural Deduction will actually double the number of open occurrences of u, hence a logical contraction appears anyway. See Section 1.3.5 for the technical details of such a contraction yielded by the simulation of the general Induction Rule (and thus also of the Induction Axiom) in terms of the more particular rule of induction restricted to assumption-less base and step input proofs. Now, what happens as a consequence of our "light" optimization? Because of the use of the quantifier without computational meaning $\overline{\forall}$ instead of the regular $\forall$ in u, this open assumption looses its Dialectica computational content, which existed only due to the presence of (the three) regular $\forall$ in a positive position. See Section 2.1 for this terminology and full technical details. The number

of open occurrences (in the original input proof) of the computationally D-redundant assumption u becomes irrelevant since this assumption is ignored anyway by the program extraction via the Light Dialectica Interpretation.

The subsequent computer benchmarks were performed on a DELL laptop (model X1, hence powered by an Intel Centrino CPU) running the Windows XP Prof. operating system. We used the more special MinLog distribution [49], which is not yet integrated with the official MinLog [116]. As Scheme interpreter we used the Petite Chez Scheme 7.0a, see [92]. The quantitative measures of computing time and space overhead were obtained by means of the Scheme "time" procedure.

1) The (MinLog, adapted) outcome of pure Gödel's Dialectica interpretation:

```
.........................
(add-var-name "n" "m" (py "nat"))
(add-var-name "G" (py "nat=>nat=>boole"))
(add-var-name "H" (py "(nat@@(nat@@nat)@@(nat@@nat))"))
.........................
> t_{PDI} == [G,n] left right
((Rec nat=>nat@@(nat@@nat)@@(nat@@nat))((0@0@0)@0@1)
  ([m,H] [if
     [if (G left left H left right left H)
        [if (G (Succ left left H) right right left H)
              (G (Succ(Succ left left H))
                   (left right left H + right right left H))
                   True]
             True]
       (m @ right H) (left H)] @ right right H@
                                left right H + right right H)
       n)
> (time (nt (mk-term-in-app-form  t_{PDI} (pt "G") (pt "5"))))
   314 collections
   6031 ms elapsed cpu time, including 676 ms collecting
   6110 ms elapsed real time, including 687 ms collecting
   341280176 bytes allocated, including 337674848 bytes reclaimed
"5"
> (time (nt (make-term-in-app-form  t_{PDI} (pt "G") (pt "6"))))
  2700 collections
  56750 ms elapsed cpu time, including 9676 ms collecting
```

```
  58375 ms elapsed real time, including 10008 ms collecting
  2937460672 bytes allocated, including 2933419728 bytes reclaimed
"8"
```

2) The outcome of the BBS refined A-translation (MinLog output, adapted):

```
.........................
(add-var-name "i" "j" "k" "l" "m" "n" (py "nat=>nat=>nat"))
(add-var-name "f" (py "nat=>nat=>nat"))
(add-var-name "H" (py "(nat=>nat=>nat)=>nat"))
.........................
> > > t_{BBS} == "[k](Rec nat=>(nat=>nat=>nat)=>nat)
([f] f 0 1) ([l,H,f] H ([i,j] H ([n,m] f m (n+m))))
k ([n,m] n)"
> (time (nt (make-term-in-app-form t_{BBS} (pt "12"))))
  39 collections
  813 ms elapsed cpu time, including 109 ms collecting
  813 ms elapsed real time, including 107 ms collecting
  42919528 bytes allocated, including 39266296 bytes reclaimed
"144"
> (time (nt (make-term-in-app-form t_{BBS} (pt "15"))))
  321 collections
  7094 ms elapsed cpu time, including 1153 ms collecting
  7203 ms elapsed real time, including 1246 ms collecting
  348911096 bytes allocated, including 326154920 bytes reclaimed
"610"
```

3) The outcome of Light Dialectica interpretation (MinLog output, adapted):

```
.........................
(add-var-name "n" "m" (py "nat"))
(add-var-name "G" (py "nat=>nat=>boole"))
(add-var-name "H" (py "(nat@@nat)"))
.........................
> t_{LDI} == "[G,n] left ((Rec nat=>nat@@nat) (0@1)
([m,H] right H @ left H + right H) n)"
> (time (nt (mk-term-in-app-form t_{LDI} (pt "G") (pt "15"))))
 6 collections
 125 ms elapsed cpu time, including 0 ms collecting
```

```
 140 ms elapsed real time, including 0 ms collecting
 6802576 bytes allocated, including 6383624 bytes reclaimed
"610"
> (time (nt (mk-term-in-app-form t_{LDI} (pt "G") (pt "20"))))
 68 collections
 1343 ms elapsed cpu time, including 62 ms collecting
 1344 ms elapsed real time, including 63 ms collecting
 73584536 bytes allocated, including 71466424 bytes reclaimed
"6765"
> (time (nt (mk-term-in-app-form t_{LDI} (pt "G") (pt "25"))))
 750 collections
 16219 ms elapsed cpu time, including 2279 ms collecting
 16657 ms elapsed real time, including 2331 ms collecting
 816525224 bytes allocated, including 803991296 bytes reclaimed
"75025"
```

Notice that the above concrete quantitative measurements of time and space overhead correspond to the distorted classical Fibonacci proof. For both Dialectica interpretation and the LD-interpretation they would be the same also for the original input proof from [10], or the proof obtained by limited automated search via `(search 1)` (instead of the unlimited `(search)`). On the contrary, for the BBS refined A-translation the difference would be rather big, since from the cleaner input proof a linear-time program is obtained, with run-time performance fairly equal to that of the output of the LD-interpretation technique (despite the difference of syntactic shape). The program $\mathtt{t}_{\mathtt{BBS}}$ displayed above at 2) can be written as a Scheme [92] program as follows:

```
(define (FiboBis n)
  (fibo2 n (lambda (k l) k)))
(define (fibo2 n1 f)
  (if (= n1 0)  (f 0 1)
       (fibo2 (- n1 1) (lambda (kk ll)
              (fibo2 (- n1 1) (lambda (k l) (f l (+ k l))))))))
```

Recall that the algorithm originally obtained in [10] could be spelled in Scheme as:

```
(define (Fibo n)
```

```
  (fibo1 n (lambda (k l) k)))
(define (fibo1 n1 f)
  (if (= n1 0)  (f 0 1)
        (fibo1 (- n1 1) (lambda (k l) (f l (+ k l))))))
```

We immediately figure out that the price to pay for the distortion in the input proof is rather big when using the `BBS` technique. The algorithm `FiboBis` is exponential in `n` because the call of `fibo2` on `n1` induces two recursive calls of `fibo2` on `n1-1`.

4.4 Conclusions and future work

More practical examples should be found for the application of the "light" optimization of Gödel's Dialectica interpretation. A negative result exists for the case of the `MinLog`-implemented semi-classical proof of Dickson's Lemma (see [15]). Here three nested Inductions give rise to three Contractions which are thus all three included in the extracted term(s), within the triply nested recursion. It is hence immediate to figure out that such a program would be *very* complex. Unfortunately, the Light Dialectica cannot repair this situation.

4.5 The integer square root example

This example is part of `MinLog`'s library as an(other) small case-study for program-extraction from classical proofs by the `BBS` refined `A`-translation. It was first described in [14]. The problem is to extract an algorithm for the computation of the greatest $k \in \mathbb{N}$ such that $k^2 \leq n$, for the given input $n \in \mathbb{N}$. Hence $n < (k+1)^2$, and thus k is the so-called "integer part" of the square root of n, since $k \leq \sqrt{n} < k+1$.

The problem is more interesting because the program synthesis is from a minimal logic proof of the weak existence of k, hence a proof that it is not true that for all $k \in \mathbb{N}$, either $n < k^2$ or $(k+1)^2 \leq n$. In fact one considers a more general proof, where the integer square function is replaced by a more generic unbounded function $f : \mathbb{N} \to \mathbb{N}$ for which $f(0) = 0$, where the unboundedness of f is specified by a(nother) function $g : \mathbb{N} \to \mathbb{N}$ for which $\forall n \in \mathbb{N} \, . \, n < f(g(n))$, i.e., g witnesses for the unboundedness of f. Here is the `MinLog` code for building such a proof of the weak existence of k, given f, g and n :

```
(load "c:/minlog/init.scm")
(mload "../lib/nat.scm")
(add-var-name "f" "g" (py "nat=>nat"))
(set-goal (pf "all f,g,n. (all n. n < f 0 -> bot) ->
                        all n n < f(g n) ->
                        excl k. (n < f k -> bot) !  n < f(k+1)"))
(assume "f" "g" "n" 1 2 3)
(cut (pf "all k.  n < f k -> bot"))
(search)
(ind)
(search)
(search)
(save "Square-Root-Theorem")
```

The proof is quite simple and apparently harmless, except that the induction step is proved via the uncancelled assumption

$$\forall k.\,(n < fk \to \bot) \to n < f(\mathtt{Suc}\,k) \to \bot \tag{4.8}$$

which triggers a computationally relevant contraction over such formula in the simulation in terms of $\mathtt{IR}_0$. Unfortunately this contraction cannot be eliminated by means of the `ncm` quantifiers, in the sense that the $\forall$ in (4.8) cannot be replaced by $\overline{\forall}$. Nonetheless a purely cosmetic use of the `ncm` quantifiers can still improve the layout of the extracted program. One thus changes the input proof to

```
(set-goal (pf "all f,g,n. (allnc n. n < f 0 -> bot) ->
                        allnc n n < f(g n) ->
                        excl k. (n < f k -> bot) ! n < f(k+1)"))
(assume "f" "g" "n" 1 2 3)
(cut (pf "all k. n < f k -> bot"))
(search)
(ind)
(search)
(search)
```

Thus the Light Dialectica will only produce one extracted term, instead of three produced by the Pure Dialectica (two of which would be simply ignored).

The `MinLog` output of the application of Pure Dialectica is

```
> (pp (nt tk))
[f,g,n2]
 right((Rec nat=>nat@@nat) (n2@0)
     ([n3,(nat@@nat)_4]
            left(nat@@nat)_4@
            [if [if [if (n2 < f(right(nat@@nat)_4)) False True]
                [if (n2 <f(Succ right(nat@@nat)_4)) False True]
              True]
         n3
         (right(nat@@nat)_4)])
       (g n2))
```

whereas the application of Light Dialectica yields the far more readable

```
> (pp tk)
[f,g,n2]
 (Rec nat=>nat) 0
   ([n3,n4]
     [if [if [if (n2 < f n4) False True]
         [if (n2 < f(Succ n4)) False True] True]
       n3
        n4])
 (g n2)
```

The computational complexity of the two programs is the same, yet the "light" optimization of Dialectica helps to clean up some garbage from the extracted program. This gets to look very close to the output of the `BBS` refined `A`-translation (which seems to be even simpler):

```
> (pp et)
     [f,g,n2] (Rec nat=>nat) 0
                           ([n3,n4] [if (n2<f n3) n4 n3])
                           (g n2)
```

But this very small difference covers a bigger difference in the run-time performance of the programs extracted via these two different techniques for the mining of classical proofs. On the input $n \equiv 127$ the program extracted by Dialectica Light is ten times faster than the program yielded by the `BBS` refined

A-translation. Just compare these raw figures, obtained in equal execution conditions[15]:

```
================ Output by Dialectica Light ===================
> (time (nt (mk-term-in-app-form tk sqr (pt "Succ") (pt "127"))))
  251 collections
  6922 ms elapsed cpu time, including 189 ms collecting
  7031 ms elapsed real time, including 186 ms collecting
  273169608 bytes allocated, including 272116664 bytes reclaimed
"11"
================ Output by BBS A-translation ==================
> (time (nt (mk-term-in-app-form et sqr (pt "Succ") (pt "127"))))
  2327 collections
  65907 ms elapsed cpu time, including 2530 ms collecting
  66250 ms elapsed real time, including 2531 ms collecting
  2529576584 bytes allocated, including 2529612232 bytes reclaimed
"11"
```

Despite the unavoidably included Contraction, the program synthesized Light Dialectica is simply more efficient than the one given by the BBS refined A-translation. The explanation is immediate once we compare the operational semantics of the two programs. The one yielded by the BBS A-translation computes all the squares of k, downwards from $n+1$ to the integer part of $\sqrt{n}$. For $n \equiv 127$ this makes $127-11 = 116$ operations. The program given by light Dialectica makes the more clever choice of going bottom-up, computing squares of k, upwards from 0 to the integer part of $\sqrt{n}+1$, hence only 12 such operations for $n \equiv 127$. It is true that the program will have to first go recursively downwards from n to 0, hence consuming a lot of the stack-space, but at the moment when it finds the answer, squares are computed repeatedly just for $k \equiv 11$, which is much faster than the calculation of squares for bigger numbers $k \in \overline{11,127}$, particularly with unary integers. Hence there are a lot of redundancies also in the Light Dialectica program, but these are only due

[15] These computer benchmarks were performed on a DELL laptop (model X1, hence powered by an Intel Centrino CPU) running the Windows XP Prof. operating system. Each of the two runs had available all the system resources, no other user-applications were being executed in parallel. We used the more special MinLog distribution [49], which is not yet integrated with the official MinLog [116]. As Scheme interpreter we used the Petite Chez Scheme 7.0a, see [92]. The quantitative measures of computing time and space overhead were obtained by means of the Scheme "time" procedure.

to the restrictions of our term language, in particular the very generic Gödel recursor `R`, which imposes a top-down recursion for simulating the bottom-up search. Nonetheless, it is immediately clear that the Light Dialectica program is conceptually superior to the refined `A`-translation program.

Discussion

More practical examples should be found for the application of the "light" optimization of Gödel's Dialectica interpretation. A negative result exists for the case of the `MinLog`-implemented semi-classical proof of Dickson's Lemma (see [15]). Here three nested Inductions give rise at three Contractions which are thus all three included in the extracted term(s), within the triply nested recursion. It is hence immediate to figure out that such a program would be *very* complex. Unfortunately, the Light Dialectica cannot repair this situation. In connection with Dickson's Lemma we have to mention the recent impressive practical result obtained by Raffalli [106]. He extracted a quite fast algorithm from a "Mixed Logic" proof of the most general form of Dickson's Lemma, outperforming the results obtained by means of the `BBS` refined `A`-translation, which only applies for limited cases (due to the lack of full polymorphism in `MinLog`). Moreover, the `BBS` extracted programs are slower than Raffalli's.

In the comparison of results for program extraction from classical proofs one should also consider other such program-synthesis techniques. One such promising method is Berger's Kripke-style refined `A`-translation from [9], which can be viewed as an upgrade of the `BBS` technique, which integrates both `ncm` and "Kripke" optimizations and allows a limited form of quantification over predicates. Another refined `A`-translation is due to Avigad [2, 3]. The refined `A`-translation of Coquand and Hofmann [23] is already integrated into Berger's, which produces much cleaner output, see the Fibonacci example in [9].

Also the techniques of Parigot [102] and Krivine [87] should be analysed in comparison with the more proof-theoretical methods. Such work is already reported in [96].

Chapter 5

Synthesis of moduli of uniform continuity by the LMD-interpretation in MinLog

[1] We extract on the computer a number of moduli of uniform continuity for the first few elements of a sequence of closed terms $\mathbf{t}$ of Gödel's $\mathbf{T}$ of type $(\mathbb{N}\to\mathbb{N})\to(\mathbb{N}\to\mathbb{N})$. The generic solution may then be quickly inferred by the human. The automated synthesis of such moduli proceeds from a proof of the hereditarily extensional equality ($\approx$) of $\mathbf{t}$ to itself, hence a proof in a weakly extensional variant of Berger-Buchholz-Schwichtenberg's system Z of $\mathbf{t} \approx_{(\mathbb{N}\to\mathbb{N})\to(\mathbb{N}\to\mathbb{N})} \mathbf{t}$. We use an implementation on the machine, in Schwichtenberg's `MinLog` proof-system, of the light Monotone Dialectica interpretation, introduced in Section 2.2. Non-computational-meaning quantifiers put aside, this is a non-literal adaptation to Natural Deduction of Kohlenbach's monotone functional interpretation. Moreover, we here use a variant of the light Monotone Dialectica which directly produces terms in `NbE`-normal form by means of a recurrent partial `NbE`-normalization. Such partial evaluation is strictly necessary in order to obtain programs on the machine in due time. For the practical results described in this section the `ncm` quantifiers were not actually used, but we suspect that they may reduce the running time of the monotone extraction algorithm in this case. Nonetheless, they would not really impact on the normalized extracted term, which we obtain in a reasonable amount of time anyway (less than a minute, for the crucial run).

[1] This section is mainly based on [55], but the text is here largely rephrased, also in view of the new developments theresince.

In order to proceed with our exposition, please recall from Section 1.5 the definition of our base monotonic arithmetical systems and from Section 2.2 the actual definition of the LMD-interpretation. On the other hand, Kohlenbach's original Monotone Dialectica interpretation (abbreviated MD-interpretation) is a recursive syntactic translation from proofs in $\mathtt{WeZ}_{\mathtt{m}}^{\exists+}$ to proofs in $\mathtt{WeZ}_{\mathtt{m}}^{\exists}$, such that the positive occurrences of the strong $\exists$ and the negative occurrences of $\forall$ in the proof's conclusion formula get effectively (either Howard [58] or Bezem [16]) majorized at each of the proof-recursion steps [2] by terms in Gödel's $\mathbf{T}$. These *majorizing terms* are also called "the programs extracted by" the MDI and (if only the extracted terms are wanted) this translation process is also referred to as "Monotone Dialectica program-extraction". Here $\mathtt{WeZ}_{\mathtt{m}}^{\exists+}$ is the system $\mathtt{WeZ}_{\mathtt{m}}^{\exists,\mathtt{nc}+}$ without the ncm quantifiers and their related restrictions.

Gödel's Dialectica interpretation becomes far more complicated when it has to face Contraction, which in Natural Deduction amounts to the discharging of more than one copy of an uncancelled assumption in an Implication Introduction $\dfrac{[A]\ \ldots\ /B}{A \to B}$. This is because, for the contractions which are relevant to Dialectica[3], the contraction formula A becomes[4] part of the raw (not yet normalized) realizing term. A number of such *D-relevant* contraction formulas, which would not be part of the executed finally strongly normalized extracted term, can be eliminated already at the extraction stage, see [54] (or Section 4.3 here) for such an example. Unfortunately, such an a priori elimination during extraction of some of the contractions (which we named "redundant" in [54], see Remark 1.19 here) is not always possible, see also [54] for such a negative example. The MD-interpretation simplifies the Dialectica treatment of all non-redundant relevant contractions and therefore represents an important complexity improvement of the extracted program whenever such "persistent" contractions occur in the proof at input.

Even though in the application presented in this chapter we do not use the optimization brought by the ncm quantifiers, we still develop another improvement of the MD-interpretation, which shows to be very effective in the machine

[2] This is exactly the point of Kohlenbach's MD-interpretation from [68], in contrast to his precursor of the MDI from [64] which first extracts the effective Gödel's Dialectica exact realizers and subsequently majorizes them via the algorithms of either Howard or Bezem.

[3] Not all logical contractions are relevant for the Dialectica interpretations, see [54] for a short account of this issue or Section 1.2 for full details.

[4] Via the boolean term associated (Definition 2.6) to the Dialectica-radical formula $A_{\mathtt{D}}$ (a quantifier-free formula) which is at its turn associated to the formula A via Definition 2.1.

implementation. We continue to name "light Monotone Dialectica" our variant, which still differs from Kohlenbach's, even though the ncm quantifiers are not used here. See Section 5.2 below for full details of this LMD-interpretation.

5.1 The minimal arithmetic HeExtEq proof in the computer-system MinLog

We used the variant [49] of MinLog which includes the light and the monotone Dialectica extraction modules. Except for this addition, our variant does not really differ from the mainstream MinLog distribution [116]. Recall that MinLog is based on a first order Natural Deduction calculus and uses as primitive minimal rather than classical or intuitionistic logic.

The hereditarily-extensional-equality test-case (abbreviated HeExtEq) was suggested by U. Kohlenbach as an interesting example for the application of the Monotone Dialectica program extraction from proofs, see Chapter 8 of [63]. In fact it had been carried out at a theoretical level already in Chapter 5 of [71] by means of the precursor of the Monotone Dialectica introduced in [64]. The treatment in [63] is even more platonic, by means of a good number of meta-theorems. We took the challenge to use a machine extraction in order to analyse on the computer a number of concrete instances of the HeExtEq example.

Definition 5.1 (Hereditarily extensional equality) As a parallel with the majorizability relation (see Definition 1.61), the *hereditarily extensional equality* is defined over the **T** type structure by

$$\begin{array}{rcl} x \approx_\iota y & :\equiv & x =_\iota y \\ x \approx_{\rho\tau} y & :\equiv & \forall z_1^\rho, z_2^\rho (z_1 \approx_\rho z_2 \to x z_1 \approx_\tau y z_2) \quad . \end{array}$$

Recall that our equality is extensional, hence defined by $(\sigma \equiv \sigma_1 \dots \sigma_n \iota)$:

$$\begin{array}{rcl} x =_\iota y & :\equiv & \mathtt{at}(= xy) \\ x =_\sigma y & :\equiv & \forall z_1^{\sigma_1} \dots z_n^{\sigma_n} (x z_1 \dots z_n =_\iota y z_1 \dots z_n) \quad , \end{array}$$

where $=$ is the usual equality boolean function on $\mathbb{N} \times \mathbb{N}$. It is immediate that $x =_{\rho\tau} y \ \equiv \ \forall z^\rho (xz =_\tau yz)$.

Definition 5.2 (Minimal Arithmetic) We denote by $\mathtt{WeZ}_{\mathtt{min}}$ the system $\mathtt{WeZ}^\exists$ without the strong $\exists$ and also without the Ex-Falso-Quodlibet axiom $\bot \to F$, hence with an underlying Minimal Logic (in the sense of [60]) substructure.

Proposition 5.3 ([71] - 5.13 or [63] - 8.17 , adapted)
Let t^{ρ} be a closed term of Gödel's $\mathbf{T}$. Then $\mathtt{WeZ}_{\mathtt{min}} \vdash t \approx_{\rho} t$.

Proof: By induction on the combinatorial structure of t, since closed terms of Gödel's $\mathbf{T}$ can be expressed as built by application only (i.e., without lambda-abstraction) from 0, Suc, Gödel's recursor $\mathcal{R}$ and combinators Σ and Π. See Proposition 1.53 for the translation from lambda-terms to combinatorial terms which is inspired by Lemma 2.6 of [71]. □

Corollary 5.4 ([71, 63] adapted) Let $t^{(\iota\to\iota)\to(\iota\to\iota)}$ be a closed $\mathbf{T}$-term. Since

$$\mathtt{WeZ}_{\mathtt{min}} \quad \vdash \quad \forall x^{\iota\to\iota}, y^{\iota\to\iota}\,[\, x =_{\iota\to\iota} y \,\leftrightarrow\, x \approx_{\iota\to\iota} y\,]$$

it immediately follows that

$$\mathtt{WeZ}_{\mathtt{min}} \quad \vdash \quad \forall x^{\iota\to\iota}, y^{\iota\to\iota}\,[\, x =_{\iota\to\iota} y \,\to\, t(x) =_{\iota\to\iota} t(y)\,] \quad .$$

Proposition 5.5 ([71] - 5.15 or [63] - 8.19 , adapted) Let $t^{(\iota\to\iota)\to(\iota\to\iota)}$ be a closed term of Gödel's $\mathbf{T}$. Then t is uniformly continuous on every closed interval $B_y :\equiv \{x^{\iota\to\iota} \,|\, \forall z^{\iota}.\, y(z) \succeq_{\iota} x(z)\}$ with a modulus of uniform continuity which is effectively synthesizable (uniformly in $y^{\iota\to\iota}$) as a closed term $\widetilde{t}(y)^{\iota\to\iota}$ of $\mathbf{T}$, i.e., one can extract (by LMD-interpretation) a closed $\mathbf{T}$-term $\widetilde{t}^{\,(\iota\to\iota)\to(\iota\to\iota)}$ such that:

$$\mathtt{WeZ}_{\mathtt{min}} \,\vdash\, \forall y\,\forall x_1, x_2 \in B_y\,\forall k^{\iota}\,[\, \bigwedge_{i=0}^{\widetilde{t}(y)(k)} x_1(i) =_{\iota} x_2(i) \,\to\, \bigwedge_{j=0}^{k} t(x_1)(j) =_{\iota} t(x_2)(j)\,]$$

Proof: Straightforward from Corollary 5.4 and Corollary 2.14, in our majorant extraction "light" setting. See also [71, 63] for details (in the Hilbert-style setting) of the proof originally introduced in [65]. □

The HeExtEq example was implemented in MinLog [49] in the sense that a minimal arithmetic MinLog proof of

$$\forall x^{\iota\to\iota}, y^{\iota\to\iota}\,[\, x =_{\iota\to\iota} y \,\to\, t(x) =_{\iota\to\iota} t(y)\,]$$

is mechanically generated for each particular $\mathbf{T}$-term $t^{(\iota\to\iota)\to(\iota\to\iota)}$ by a Scheme [92] procedure which takes as argument such a concrete MinLog $\mathbf{T}$-term t.

5.2 The MinLog machine majorant extraction

Our approach for the MinLog extraction (theoretically described by Proposition 5.5) of the generic modulus of uniform continuity $\widetilde{t}$, given the concrete MinLog term t, differs from Kohlenbach's original Monotone Dialectica from [68] not only by the Natural Deduction formulation but also by allowing free variables in the extracted pre-majorants. We thus combine those features of the pre-existing versions due to Kohlenbach[5] which turn out to be useful on the machine. One extra particularity of this new "light" MD-interpretation is the production of terms in NbE-normal form. In general, the normal form of a term may show to be (much) bigger than its more compact representation by means of lambda-abstractions. But on the other hand normalization may eliminate many redundant parts of the lambda-terms. Our practical experience with the automated, machine program-extraction, shows that the latter situation appears more often in our experiments, in particular for the HeExtEq case.

The key features of this novel form of MD-interpretation are the following:

1. The terms extracted at each step of the recursion over the input proof structure are neither exact realizers, nor majorants, but pre-majorants (i.e. their lambda-closures are true majorants)

2. An NbE-normalization (see [13, 11, 12] for the original NbE) of such extracted pre-majorants is carried out for optimization purposes after the proof mining of the conclusion at each Implication Elimination (aka Modus Ponens) application. This recurrent form of partial normalization turns out to bring a huge improvement w.r.t. the one single final call-by-value NbE normalization process in situations of long sequences of nested Modus Ponens. We named this technique[6] "Normalization during Extraction" (abbreviated "NdE"), see [53] for a short account. The HeExtEq proof (described in Section 5.1 above) does actually contain quite long sequences of nested Modus Ponens.

3. The final such extracted pre-majorant is NbE-normalized. Let $t[x_1, x_2, j]$ be this pre-majorant (in normal form). Then the searched modulus of

[5] We distinguish three such variants of the Monotone Dialectica interpretation, which were introduced in (chronologically ordered) [64], [68] and finally [69]. See also Zucker's chapter VI in [123], particularly its sections 8.3-6, for a raw, unformalized and quite primitive form of MD-interpretation.

[6] Which is a recurrent form of Partial Evaluation. See the volume [24] for accounts of the partial evaluation programming methodology.

uniform continuity is $\widetilde{t} :\equiv \lambda y, k.\, t[y^M, y^M, k]$, where the type-2 functional $\cdot^M$ is defined in Proposition 2.19.

5.3 Machine results for the HeExtEq case-study

We used our light Monotone Dialectica MinLog extraction modules which are available within the special[7] MinLog distribution [49]. We applied the LMDI extraction on the MinLog HeExtEq proof for the following concrete instances of the term t:

- The simple sum: $f, k \mapsto f(0) + \ldots + f(k)$.
- The double sum: $f, k \mapsto f(f(0)) + \ldots + f(f(k))$.
- The triple sum: $f, k \mapsto f(f(f(0))) + \ldots + f(f(f(k)))$.

In the case of the simple sum, the machine output is, as expected, the identity function, regardless of the actual f, hence the functional $f, k \mapsto k$. Also for the double sum, the outcome is the expected one, namely

$$f, k \quad \mapsto \quad max\{k\,,\, f(0)\,,\, \ldots\,,\, f(k)\} \quad .$$

On the contrary, for the triple sum, the mathematician needs to work a good number of minutes to produce the following *optimal* result

$$f, k \;\mapsto\; max\{k\,,\, f(0)\,,\, f(1)\,,\, \ldots\,,\, f(max\{k\,,\, f(0)\,,\, f(1)\,,\, \ldots\,,\, f(k)\})\} \quad (5.1)$$

The machine produces in less than one minute an output which can be isomorphically adapted for display as follows:

$$\begin{aligned} f, k \quad \mapsto \quad & max\{k\,,\, f(0)\,,\, \ldots\,,\, f(k)\,, \\ & max\{f(0)\,,\, \ldots\,,\, f(max\{f(0)\,,\, \ldots\,,\, f(k)\})\}\} \end{aligned} \quad (5.2)$$

It is easy to notice that the machine-yielded expression (5.2) is immediately equivalent to the more human expression (5.1). Note also that in the context of a pointwise continuity demand, the optimal answer would be

$$f, g, k \;\mapsto\; max\{k\,,\, f(0)\,,\, f(1)\,,\, \ldots\,,\, f(k)\,,\, max\{f(f(0))\,,\, f(f(1))\,,\, \ldots\,,\, f(f(k))\}\}$$

[7] Our Dialectica modules are for the moment not compatible with the official MinLog distribution from [116].

which is strictly lower than the machine (or human) optimal output for the case of uniform continuity. In fact, while first trying to solve by brain the triple sum problem, we first erroneously thought that this were a modulus of uniform continuity, which is not the case. We later produced (5.1) by simplifying the machine outcome (5.2) and after some checks we realized the error. Hence we could produce a correct answer only with the help of the computer extraction.

Notwithstanding, right now a pattern can be noticed by the human in the solution of the HeExtEq problem for terms $t_l :\equiv \lambda f,k.\, f^{(l)}(0) + \ldots + f^{(l)}(k)$, with $f^{(l)}(i) :\equiv f(f \ldots (f(i)))$, where f appears l times on the right-hand side. We write again the above moduli of uniform continuity for t_l, with $l := 1, 2, 3$:

$$\begin{array}{lcl} \widetilde{t_1}(f,k) & \equiv & k \\ \widetilde{t_2}(f,k) & \equiv & max\{k, f(0), \ldots, f(\widetilde{t_1}(f,k))\} \\ \widetilde{t_3}(f,k) & \equiv & max\{k, f(0), \ldots, f(\widetilde{t_2}(f,k))\} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & & \end{array}$$

We thus immediately infer the generic (recursive) solution for every $l \in \mathbb{N}$:

$$\widetilde{t_{l+1}}(f,k) \quad \equiv \quad max\{k, f(0), \ldots, f(\widetilde{t_l}(f,k))\}$$

The verification that $\widetilde{t_l}$ is the optimal modulus of uniform continuity for t_l is now an easy exercise.

Discussion

More such MinLog extractions of moduli of uniform continuity for other various concrete instances of the input term t can and ought to be performed. Section 2.2 presents a full mathematical formalization of the light MD-interpretation, which integrates the *light* optimization of Gödel's Dialectica from [51] with the optimization of the Monotone Dialectica from [55] (also called "light"). We thus accomplished part of the research schedule outlined in the "Conclusions and future work" section of [55]. We still need to see whether some of the contractions of the HeExtEq proof can be completely eliminated. Even if this would be the case, it would nevertheless not have any impact on the already extracted program, but only on the extraction process, which is already "quick enough". Also a complete mathematical formulation of the Normalization during Extraction (NdE) ought to be given as soon as possible.

MinLog source code and output for HeExtEq

MinLog source code for the Double Sum

```
(load "$MINLOGPATH/init.scm")
(mload "../lib/nat.scm")
(mload "../FI/pds.scm")
(mload "../FI/max.scm")
(mload "../FI/heeq/heeq-def.scm")

(av "g" (py "(nat=>nat)=>nat"))
(av "h" (py "(nat=>nat)=>nat=>nat"))

(define t0 (pt "[f,k] ((Rec nat=>nat) 0 ([n,m](f (f n))+m) k)"))
(define p1 (term-to-heeq-proof t0))

(make-max)
(mload "../FI/mon_fiets.scm")
(define vatmp (time (FI-extracted-vatmpair p1)))
(define untup (tmpair-to-tuple (vatmpair-to-tmpair vatmp)))
(define tmlst (tmtuple-to-tmlist untup))
(length tmlst)
(define raw_et (car tmlst))
(string-length (term-to-string raw_et))
(define size_and_depth (term-to-sad raw_et ))
(cdr size_and_depth)
(car size_and_depth)
(set! UNFOLDING-FLAG #t)
(define norm_et  (time (nt raw_et)))
(define un_et (mk-term-in-app-form norm_et
                                   (term-to-maj (pt "f1"))
                                   (term-to-maj (pt "f2"))))
(define et (nt un_et))
(pp et)
(pp (time (nt (mk-term-in-app-form et (pt "5")))))
```

MinLog selected output for the Double Sum

```
(time (fi-extracted-vatmpair p1))
```

```
    918 collections
    53250 ms elapsed cpu time, including 3230 ms collecting
    55219 ms elapsed real time, including 3341 ms collecting
    997496240 bytes allocated, including 992936792 bytes reclaimed
> > > 1 > > 78282 > > 252 > 3911
> > (time (nt raw_et))
    7 collections
    265 ms elapsed cpu time, including 0 ms collecting
    281 ms elapsed real time, including 0 ms collecting
    7728760 bytes allocated, including 7433432 bytes reclaimed
> > > [n3]  Max
 left right((Rec nat=>nat@@(nat@@(nat@@nat)))(0@0@0@0)
  ([n4,(nat@@(nat@@(nat@@nat)))_5]
    Max left(nat@@(nat@@(nat@@nat)))_5 n4@
    Max left right(nat@@(nat@@(nat@@nat)))_5 n4@
    Max left right right(nat@@(nat@@(nat@@nat)))_5
    ((Rec nat=>nat)0
     ([n6] NatPlus (Max
        ((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))
         (Max((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))n6)(f1 n6)))
        (f1(Max((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))n6)(f1 n6)))))
     n4)@
    Max right right right(nat@@(nat@@(nat@@nat)))_5
    ((Rec nat=>nat)0
     ([n6] NatPlus (Max
        ((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))
         (Max((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))n6)(f2 n6)))
        (f2(Max((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))n6)(f2 n6)))))
     n4)) n3)
 (Max
  ((Rec nat=>nat)0([n4,n5]Max n5(f2 n4))
   left right((Rec nat=>nat@@(nat@@(nat@@nat)))(0@0@0@0)
    ([n4,(nat@@(nat@@(nat@@nat)))_5]
      Max left(nat@@(nat@@(nat@@nat)))_5 n4@
      Max left right(nat@@(nat@@(nat@@nat)))_5 n4@
      Max left right right(nat@@(nat@@(nat@@nat)))_5
      ((Rec nat=>nat)0
```

```
       ([n6] NatPlus (Max
          ((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))
           (Max((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))n6)(f1 n6)))
          (f1(Max((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))n6)(f1 n6)))))
       n4)@
      Max right right right(nat@@(nat@@(nat@@nat)))_5
      ((Rec nat=>nat)0
       ([n6] NatPlus (Max
          ((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))
           (Max((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))n6)(f2 n6)))
          (f2(Max((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))n6)(f2 n6)))))
       n4)) n3))
  (f2
   left right((Rec nat=>nat@@(nat@@(nat@@nat)))(0@0@0@0)
    ([n4,(nat@@(nat@@(nat@@nat)))_5]
      Max left(nat@@(nat@@(nat@@nat)))_5 n4@
      Max left right(nat@@(nat@@(nat@@nat)))_5 n4@
      Max left right right(nat@@(nat@@(nat@@nat)))_5
      ((Rec nat=>nat)0
       ([n6] NatPlus (Max
          ((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))
           (Max((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))n6)(f1 n6)))
          (f1(Max((Rec nat=>nat)0([n8,n9]Max n9(f1 n8))n6)(f1 n6)))))
       n4)@
      Max right right right(nat@@(nat@@(nat@@nat)))_5
      ((Rec nat=>nat)0
       ([n6] NatPlus (Max
          ((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))
           (Max((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))n6)(f2 n6)))
          (f2(Max((Rec nat=>nat)0([n8,n9]Max n9(f2 n8))n6)(f2 n6)))))
       n4)) n3)))
> (time (nt (mk-term-in-app-form et (pt "5"))))
    47 collections
    1515 ms elapsed cpu time, including 16 ms collecting
    1546 ms elapsed real time, including 15 ms collecting
    51104760 bytes allocated, including 50986624 bytes reclaimed
Max 4(Max(Max(Max(Max(f2 0)(f2 1))(f2 2))(f2 3))(f2 4))
```

Conclusions

We presented in this thesis both the theory and the practice of extracting programs from (classical) arithmetical proofs by means of an optimization of Gödel's Dialectica interpretation and of its monotone variant due to Kohlenbach. The novel proof interpretation which we named "Dialectica Light" completely eliminates in some cases the contractions from the input proof which would otherwise be included in the pure Dialectica extracted terms and therefore would be computationally redundant. This was achieved by means of an adaptation of Berger's non-computational (or "uniform") quantifiers (abbreviated `ncm`) to the Dialectica-extraction context. The addition of the `ncm` quantifiers was much simpler in the case of the monotonic arithmetics to which Kohlenbach's majorant extraction by Monotone Dialectica normally applies. Therefore, the new light monotone functional interpretation extends much easier to majorant and bound extractions from fully classical proofs.

The concrete examples treated on the computer by means of the Light (Monotone) Dialectica are quite promising in the cases where these new extraction techniques truly apply. Also the theory of extracting poly-time computable bounds by the `LMD`-interpretation is developed as an extension of Kohlenbach's framework for a polynomially bounded Analysis with elements from an older poly-time framework due to Cook and Urquhart. Practical applications of this theory are nonetheless still to be found.

In the end, the Appendix chapter presented a somewhat older joint work with Ulrich Kohlenbach on the computational complexity of the Dialectica extraction algorithms. The very low (worst-case cubic) complexity was obtained for a presentation of Dialectica in Hilbert-style systems, but the result immediately adapts to the Natural Deduction systems used in the main part of this thesis. Our Natural Deduction framework for the Dialectica interpretation is an improvement of Jørgensen's historically first such presentation of Gödel's original extraction technique.

Bibliography

[1] W. Alexi. Extraction and verification of programs by analysis of formal proofs. *Theoretical Computer Science*, 61:225–258, 1988. (pp. xxi and 181.)

[2] J. Avigad. Formalizing forcing arguments in subsystems of second-order arithmetic. *Annals of Pure and Applied Logic*, 82:165–191, 1996. (pp. 147, 181 and 235.)

[3] J. Avigad. Interpreting classical theories in constructive ones. *The Journal of Symbolic Logic*, 65:1785–1812, 2000. (pp. iv, v, 10, 11 and 147.)

[4] J. Avigad and S. Feferman. Gödel's functional ('Dialectica') interpretation. In S. Buss, editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 337–405. Elsevier, 1998. (pp. v, vii, xi, xvii, xx, 11, 16, 59, 76, 126, 135, 137, 179, 187, 224, 233, 234, 235 and 236.)

[5] F. Barbanera and S. Berardi. Extracting constructive content from classical logic via control–like reductions. In M. Bezem and J. Groote, editors, *Typed Lambda Calculi and Applications*, pages 45–59. LNCS Vol. 664, 1993. (pp. iv and 10.)

[6] 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. (p. 162.)

[7] J. L. Bates and R. L. Constable. Proofs as programs. *ACM Transactions on Programming Languages and Systems*, 7(1):113–136, January 1985. (pp. iv and 10.)

[8] A. Beckmann. Exact bounds for lengths of reductions in typed λ-calculus. *J. of Symbolic Logic*, 66(3):1277–1285, 2001. (pp. 108, 113 and 114.)

[9] U. Berger. Uniform Heyting Arithmetic. *Annals of Pure and Applied Logic*, 133(1-3):125–148, 2005. Festschrift for H. Schwichtenbergs 60th birthday. (pp. vi, ix, xii, 11, 12, 13, 20, 21, 24, 27, 29, 49, 60, 128, 136, 137 and 147.)

[10] U. Berger, W. Buchholz, and H. Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002. (pp. iv, v, vii, ix, xvii, 10, 11, 16, 17, 19, 20, 24, 29, 37, 41, 44, 127, 128, 134, 135, 136, 137, 138, 142, 180 and 181.)

[11] U. Berger, M. Eberl, and H. Schwichtenberg. Normalization by evaluation. In B. Möller and J. Tucker, editors, *Prospects for Hardware Foundations*, volume 1546 of *LNCS*, pages 117–137. Springer Verlag, 1998. (pp. 135 and 152.)

[12] U. Berger, M. Eberl, and H. Schwichtenberg. Term rewriting for normalization by evaluation. *Information and Computation*, 183(1):19–42, May 2003. International Workshop on Implicit Computational Complexity (ICC'99). (pp. 135 and 152.)

[13] U. Berger and H. Schwichtenberg. An inverse of the evaluation functional for typed λ-calculus. In R. Vemuri, editor, *Proceedings 6'th Symposium on Logic in Computer Science (LICS'91)*, pages 203–211. IEEE Computer Society Press, Los Alamitos, 1991. (p. 152.)

[14] U. Berger and H. Schwichtenberg. Program extraction from classical proofs. In D. Leivant, editor, *Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, IN, USA, October 1994*, volume 960 of *LNCS*, pages 77–97. Springer Verlag, 1995. (p. 143.)

[15] U. Berger, H. Schwichtenberg, and M. Seisenberger. The Warshall algorithm and Dickson's lemma: Two examples of realistic program extraction. *Journal of Automated Reasoning*, 26:205–221, 2001. (pp. iv, xvii, 10, 127, 143 and 147.)

[16] M. Bezem. Strongly majorizable functionals of finite type: A model for bar-recursion containing discontinuous functionals. *J. Symb. Log.*, 50(3):652–660, 1985. (pp. 83 and 149.)

[17] J. Blanck and al., editors. *Proceedings Fourth Workshop on Computability and Complexity in Analysis (CCA 2000)*, volume 2064 of *Springer LNCS*. Springer, 2001. (p. 165.)

[18] W. Burr. Functional interpretation of Aczel's constructive set theory. *Annals of Pure and Applied Logic*, 104:31–75, 2000. (p. 179.)

[19] S. Buss, editor. *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1998. (pp. 169 and 170.)

[20] R. L. Constable and C. Murthy. Finding computational content in classical proofs. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 341–362. Cambridge University Press, 1991. (pp. iv and 10.)

[21] S. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the 7th ACM Symposium on the Theory of Computation*, pages 83–97, 1975. (pp. 104, 108 and 109.)

[22] S. Cook and A. Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63:103–200, 1993. (pp. xiii, 13, 102, 104, 108, 109, 110, 113, 179, 181, 195 and 232.)

[23] 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. (pp. iv, v, xvii, 10, 11, 127, 136, 147 and 180.)

[24] O. Danvy, R. Glück, and P. Thiemann, editors. *Partial Evaluation. Dagstuhl Castle, Germany, February 1996*, volume 1110 of *Lecture Notes in Computer Science*. Springer Verlag, 1996. (p. 152.)

[25] J. Diller and W. Nahm. Eine Variante zur Dialectica Interpretation der Heyting Arithmetik endlicher Typen. *Arch. Mathem. Logik und Grundl.*, 16:49–66, 1974. (pp. xi, xiii, 59, 102 and 187.)

[26] A. Dragalin. New kinds of realisability and the Markov rule. *Dokl. Akad. Nauk. SSSR*, 251:534–537, 1980. In Russian, the English Translation is [27]. (pp. v, 11 and 180.)

[27] A. Dragalin. New kinds of realisability and the Markov rule. *Soviet Math. Dokl.*, 21:461–464, 1980. (p. 162.)

[28] S. Feferman. Theories of finite type related to mathematical practice. In [6], pages 913–972. (pp. 179 and 232.)

[29] M. Felleisen, D. P. Friedman, E. Kohlbecker, and B. Duba. A syntactic theory of sequential control. *Theoretical Computer Science*, 52:205–237, 1987. (pp. iv and 10.)

[30] M. Felleisen and R. Hieb. The revised report on the syntactic theory of sequential control and state. *Theoretical Computer Science*, 102:235–271, 1992. (pp. iv and 10.)

[31] A. Fernandes and F. Ferreira. Groundwork for Weak Analysis. *The Journal of Symbolic Logic*, 67(2):557–578, 2002. (pp. xv and 104.)

[32] F. Ferreira. A feasible theory for analysis. *The Journal of Symbolic Logic*, 59(3):1001–1011, September 1994. (pp. xiv and 103.)

[33] F. Ferreira and P. Oliva. Bounded functional interpretation. *Annals of Pure and Applied Logic*, 135(1-3):73–112, September 2005. (pp. v, xi, xv, 11, 59, 61, 76, 103 and 136.)

[34] F. Ferreira and P. Oliva. Bounded functional interpretation and feasible analysis. *Annals of Pure and Applied Logic*, 145(2):115–129, 2007. (pp. xv, 61, 103 and 104.)

[35] H. Friedman. Systems of second order arithmetic with restricted induction, I, II (abstracts). *The Journal of Symbolic Logic*, 41:557–559, 1976. (p. 234.)

[36] H. Friedman. Classical and intuitionistically provably recursive functions. In G. Müller and D. Scott, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–27. Springer Verlag, 1978. (pp. v, 11 and 180.)

[37] P. Gerhardy. Improved Complexity Analysis of Cut Elimination and Herbrand's Theorem. Master's thesis, University of Aarhus, Department of Computer Science, 2003. (pp. xx and 179.)

[38] P. Gerhardy. Refined Complexity Analysis of Cut Elimination. In M. Baaz and J. Makovsky, editors, *Proceedings of the 17th International Workshop CSL 2003*, volume 2803 of *LNCS*, pages 212–225. Springer-Verlag, Berlin, 2003. (pp. xx and 179.)

[39] P. Gerhardy. The role of quantifier alternations in Cut Elimination. *Notre Dame Journal of Formal Logic*, 46(2):165–171, 2005. (pp. xx and 179.)

[40] P. Gerhardy and U. Kohlenbach. Extracting Herbrand disjunctions by functional interpretation. *Archive for Mathematical Logic*, 44:633–644, 2005. (p. 179.)

[41] P. Gerhardy and U. Kohlenbach. Strongly uniform bounds from semiconstructive proofs. *Annals of Pure and Applied Logic*, 141:89–107, 2006. (pp. x and 59.)

[42] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérior.* PhD thesis, Université de Paris VII, 1972. (p. 179.)

[43] J.-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and Types.* Cambridge University Press, 1989. (pp. 25, 137 and 173.)

[44] K. Gödel. Zur intuitionistischen Arithmetik und Zahlentheorie. *Ergebnisse eines Mathematischen Kolloquiums*, 4:34–38, 1933. (pp. 83 and 223.)

[45] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287, 1958. (pp. v, x, xi, xiii, xvii, xx, 11, 58, 59, 76, 102, 126, 135, 137, 179, 182, 188, 197 and 232.)

[46] T. G. Griffin. A formulae–as–types notion of control. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 47–58, 1990. (pp. iv and 10.)

[47] 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. (pp. 181 and 235.)

[48] S. Hayashi and H. Nakano. *PX: A Computational Logic*. MIT Press, 1988. (p. 180.)

[49] M.-D. Hernest. The `MinLog` proof-system for Dialectica program-extraction. Free software, with full code source and documentation @ http://www.brics.dk/~danher/MinLogForDialectica. (pp. xvi, xviii, 14, 29, 135, 136, 137, 138, 140, 146, 150, 151 and 153.)

[50] 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. (pp. iv, v, 10, 11, 126, 139 and 181.)

[51] M.-D. Hernest. Light Functional Interpretation. *Lecture Notes in Computer Science*, 3634:477 – 492, July 2005. Computer Science Logic: 19th International Workshop, CSL 2005. (pp. vi, xviii, 10, 12, 63, 76, 126, 129, 136, 137, 154 and 164.)

[52] M.-D. Hernest. Technical appendix to [51]. Available in the author's web-page, June 2005. (p. 10.)

[53] M.-D. Hernest. `NdE` - Normalization during Extraction. In *Local Proceedings of CiE06 (Computability in Europe 2006)*. Computability in Europe, IT Wales - University of Wales, Swansea, 2006. Extended Abstract. Full paper in preparation. Available in the author's web-page. (pp. xvii, 79, 126 and 152.)

[54] M.-D. Hernest. Light Dialectica program extraction from a classical Fibonacci proof. In *Proceedings of DCM'06 at ICALP'06*, Electronic Notes in Theoretical Computer Science (ENTCS), page 10pp. Elsevier, 2007. Accepted for publication, Downloadable @ http://www.brics.dk/~danher/. (pp. 134 and 149.)

[55] M.-D. Hernest. Synthesis of moduli of uniform continuity by the Monotone Dialectica Interpretation in the proof-system `MinLog`. In *Proceedings of LFMTP'06 at FLoC'06*, Electronic Notes in Theoretical Computer Science (ENTCS), page 12pp. Elsevier, 2007. Accepted for publication, Downloadable @ http://www.brics.dk/~danher/. (pp. iv, vi, 10, 12, 79, 148 and 154.)

[56] 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. Published in a slightly revised form as [57]. (pp. vi, xvi, 11, 126, 129, 130 and 178.)

[57] M.-D. Hernest and U. Kohlenbach. A complexity analysis of functional interpretations. *Theoretical Computer Science*, 338(1-3):200–246, June 2005. Shortened and slightly revised version of the BRICS Technical report RS-03-12, University of Aarhus, Denmark, Feb 2003. (pp. ix, 13, 44, 74, 84, 86, 165 and 178.)

[58] W. Howard. Hereditarily majorizable functionals of finite type. In [123], pages 454–461. (pp. x, xx, 38, 39, 83, 149, 179, 181, 195, 227 and 238.)

[59] W. Howard. Ordinal analysis of simple cases of bar recursion. *Journal of Symbolic Logic*, 46(1):17–30, 1981. (pp. xiv and 103.)

[60] I. Johansson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Matematica*, 4:119–136, 1936. (pp. 20 and 150.)

[61] K. Jørgensen. Finite type arithmetic. Master's thesis, University of Roskilde, Departments of Mathematics and Philosophy, 2001. (pp. ix, xi, 13, 38, 59, 76 and 187.)

[62] U. Kohlenbach. On the computational content of the Krasnoselski and Ishikawa fixed point theorems. In [17], pages 119–145. (pp. x and 59.)

[63] U. Kohlenbach. Proof Interpretations and the Computational Content of Proofs. *Lecture Course*, latest version in the author's web page. (pp. iv, v, vi, vii, x, xi, xvi, xvii, xxi, 10, 11, 16, 38, 44, 59, 81, 83, 84, 86, 93, 95, 96, 97, 98, 101, 117, 126, 127, 134, 150, 151, 224 and 225.)

[64] 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. (pp. x, 59, 93, 117, 149, 150, 152, 228, 232, 234, 235 and 237.)

[65] U. Kohlenbach. Pointwise hereditary majorization and some applications. *Archive for Mathematical Logic*, 31:227–241, 1992. (pp. 120 and 151.)

[66] U. Kohlenbach. Effective moduli from ineffective uniqueness proofs. An unwinding of de La Vallée Poussin's proof for Chebycheff approximation. *Annals of Pure and Applied Logic*, 64:27–94, 1993. (pp. x and 59.)

[67] U. Kohlenbach. New effective moduli of uniqueness and uniform a–priori estimates for constants of strong unicity by logical analysis of known proofs in best approximation theory. *Numerical Functional Analysis and Optimization*, 14:581–606, 1993. (pp. x and 59.)

[68] 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. (pp. iv, v, vi, x, xi, xiii, xvii, 10, 11, 12, 44, 58, 59, 76, 83, 102, 126, 136, 149, 152, 181, 182, 187, 227, 229 and 232.)

[69] U. Kohlenbach. Mathematically strong subsystems of analysis with low rate of growth of provably recursive functionals. *Archive for Mathematical Logic*, 36:31–71, 1996. (pp. xiii, xv, 93, 104, 110, 112, 113, 115, 116, 117, 118, 119, 120, 152, 179 and 232.)

[70] U. Kohlenbach. Arithmetizing proofs in analysis. In J. Larrazabal, D. Lascar, and G. Mints, editors, *Logic Colloquium 1996*, volume 12 of *Lecture Notes in Logic*, pages 115–158. Springer Verlag, 1998. (pp. xv, 93, 95, 110 and 117.)

[71] U. Kohlenbach. Proof interpretations. Technical report BRICS LS-98-1, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark, June 1998. Free downloadble @ http://www.brics.dk/LS/Abs/BRICS-LS-Abs/BRICS-LS-Abs.html. These lecture notes are a polished version of notes from a BRICS PhD course given in the spring term 1998. (pp. 150 and 151.)

[72] U. Kohlenbach. Proof theory and computational analysis. *Electronic Notes in Theoretical Computer Science*, 13:124–157, 1998. Comprox III, Third Workshop on Computation and Approximation Proof theory and computational analysis. (pp. xv, 104, 110 and 117.)

[73] U. Kohlenbach. On the no-counterexample interpretation. *The Journal of Symbolic Logic*, 64:1491–1511, 1999. (pp. xx and 179.)

[74] U. Kohlenbach. A note on Spector's quantifier-free rule of extensionality. *Archive for Mathematical Logic*, 40:89–92, 2001. (p. 186.)

[75] U. Kohlenbach. A quantitative version of a theorem due to Borwein-Reich-Shafrir. *Numerical Functional Analysis and Optimization*, 22:641–656, 2001. (pp. x and 59.)

[76] U. Kohlenbach. Uniform asymptotic regularity for Mann iterates. *Journal of Mathematical Analysis and Applications*, 279:531–544, 2003. (pp. x and 59.)

[77] U. Kohlenbach. Some computational aspects of metric fixed point theory. *Nonlinear Analysis*, 61(5):823–837, 2005. (pp. x and 59.)

[78] U. Kohlenbach. Some logical metatheorems with applications in functional analysis. *Transactions American Mathematical Society*, 357(1):89–128, 2005. (pp. x, 59 and 134.)

[79] U. Kohlenbach and B. Lambov. Bounds on iterations of asymptotically quasi–nonexpansive mappings. In J. Falset, E. Fuster, and B. Sims, editors, *International Conference on Fixed Point Theory and Applications, Valencia, 2003*, pages 143–172. Yokohama Publishers, 2004. (pp. x and 59.)

[80] U. Kohlenbach and L. Leuştean. Mann iterates of directionally nonexpansive mappings in hyperbolic spaces. *Abstract and Applied Analysis*, 2003(8):449–477, 2003. (pp. x and 59.)

[81] 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. (pp. iv, v, x, xvii, xxi, 10, 11, 59, 84, 86, 127, 134, 180, 187, 228 and 232.)

[82] U. Kohlenbach and P. Oliva. Proof mining in L_1-approximation. *Annals of Pure and Applied Logic*, 121:1–38, 2003. (pp. x and 59.)

[83] G. Kreisel. On the interpretation of non-finitist proofs, part I. *The Journal of Symbolic Logic*, 16:241–267, 1951. (pp. xx and 179.)

[84] G. Kreisel. On the interpretation of non-finitist proofs, part II: Interpretation of number theory. *The Journal of Symbolic Logic*, 17:43–58, 1952. (pp. xx and 179.)

[85] G. Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland Publishing Company, 1959. (pp. v, xiii, xvi, xvii, 11, 102, 126 and 135.)

[86] G. Kreisel. On weak completeness of intuitionistic predicate logic. *The Journal of Symbolic Logic*, 27:139–158, 1962. (pp. xx and 179.)

[87] J.-L. Krivine. Classical logic, storage operators and second-order lambda-calculus. *Annals of Pure and Applied Logic*, 68:53–78, 1994. (pp. iv, 10 and 147.)

[88] D. Leivant. Syntactic translations and provably recursive functions. *The Journal of Symbolic Logic*, 50(3):682–688, September 1985. (pp. iv and 10.)

[89] L. Leuştean. A quadratic rate of asymptotic regularity for CAT(0)-spaces. *Journal of Mathematical Analysis and Applications*, 325(1):386–399, 2007. (p. 134.)

[90] H. Luckhardt. *Extensional Gödel Functional Interpretation*, volume 306 of *Lecture Notes in Mathematics*. Springer Verlag, 1973. (pp. xi, 59, 84, 96, 117, 179, 186, 197, 224 and 236.)

[91] H. Luckhardt. Bounds extracted by Kreisel from ineffective proofs. In P. Odifreddi, editor, *Kreiseliana: About and around Georg Kreisel*, pages 289–300. A. K. Peters, Wellesley, MA, 1996. (pp. iv and 10.)

[92] Cadence Research Systems. Chez Scheme. http://www.scheme.com, 2006. (pp. 140, 142, 146 and 151.)

[93] C. Murthy. Extracting constructive content from classical proofs. Technical Report 90–1151, Dept. of Comp. Science of Cornell University, Ithaca, New York, 1990. PhD thesis. (pp. iv and 10.)

[94] C. Murthy. *Extracting Constructive Content from Classical Proofs*. PhD thesis, Cornell University, 1990. (p. 180.)

[95] P. Oliva. Polynomial-time algorithms from ineffective proofs. In *Proc. of the Eighteenth Annual IEEE Symposium on Logic in Computer Science LICS'03*, pages 128–137, 2003. (pp. xiv, 103, 104 and 110.)

[96] P. Oliva. On Krivine's realizability interpretation of classical second-order arithmetic. To appear in *Fundamenta Mathematicae*, available in author's home-page, September 2006. (p. 147.)

[97] P. Oliva. Understanding and using Spector's bar recursive interpretation of classical analysis. In *Proceedings of CiE'2006*, volume 3988 of *LNCS*, pages 423–434. Springer Verlag, 2006. (p. 134.)

[98] P. Oliva. Unifying functional interpretations. *Notre Dame Journal of Formal Logic*, 47:263–290, 2006. (p. 135.)

[99] 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. (pp. xx, 108 and 179.)

[100] 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. (pp. xx and 179.)

[101] G. Ostrin and S. Wainer. Elementary arithmetic. Preprint 29, University of Leeds, Department of Pure Mathematics, 2001. (pp. iv and 10.)

[102] M. Parigot. $\lambda\mu$–calculus: an algorithmic interpretation of classical natural deduction. In *Proc. of Log. Prog. and Automatic Reasoning, St. Petersburg*, volume 624 of *LNCS*, pages 190–201. Springer Verlag, 1992. (pp. iv, 10 and 147.)

[103] C. Parsons. On n-quantifier induction. *Journal of Symbolic Logic*, 37:466–482, 1972. (pp. 37, 179, 232 and 238.)

[104] C. Paulin-Mohring and B. Werner. Synthesis of ML programs in the system Coq. *Journal of Symbolic Computation*, 15(5/6):607–640, 1993. (pp. iv and 10.)

[105] P. Pudlak. The lengths of proofs. In [19], pages 547–637. (pp. xx, 108 and 179.)

[106] C. Raffalli. Getting results from programs extracted from classical proofs. *Theoretical Computer Science*, 323(1-3):49–70, 2004. (pp. iv, 10 and 147.)

[107] P. Rath. *Eine verallgemeinerte Funktionalinterpretation der Heyting Arithmetik endlicher Typen*. PhD thesis, Universität Münster, 1978. (pp. xi, 59 and 187.)

[108] B. Scarpellini. A model for bar recursion of higher types. *Compositio Mathematica*, 23:123–153, 1971. (pp. xiv and 103.)

[109] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924. (pp. 187 and 194.)

[110] H. Schwichtenberg. An arithmetic for polynomial-time computation. To be published by *Theoretical Computer Science*. Available in the author's web page. (pp. 179, 181 and 195.)

[111] H. Schwichtenberg. Minimal logic for computable functions. Lecture course on program-extraction from (classical) proofs. Available in the author's web page or in the MINLOG distribution [116]. (pp. vi, ix, 12, 17, 19, 22, 24, 25, 29, 34, 35, 37, 40, 41, 44, 84, 128, 135, 136, 137 and 138.)

[112] H. Schwichtenberg. Complexity of normalization in the pure typed lambda-calculus. In A. Troelstra and D. Van Dalen, editors, *The L.E.J. Brouwer*

Centenary Symposium, pages 453–457. North-Holland Publishing Company, 1982. (pp. 108, 113 and 114.)

[113] H. Schwichtenberg. An upper bound for reduction sequences in the typed λ-calculus. *Arch. Math. Logic*, 30:405–408, 1991. (pp. 108, 113 and 114.)

[114] H. Schwichtenberg. Monotone majorizable functionals. *Studia Logica*, 62:283–289, 1999. (p. 38.)

[115] 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. (p. 179.)

[116] H. Schwichtenberg and Others. Proof- and program-extraction system Minlog. Free code and documentation at http://www.minlog-system.de. (pp. xvi, xvii, xviii, 14, 22, 29, 41, 127, 135, 136, 137, 138, 140, 146, 150, 153, 169 and 181.)

[117] M. Seisenberger. *On the Constructive Content of Proofs*. PhD thesis, University of Munich, Faculty of Mathematics, 2003. (p. 129.)

[118] S. Simpson. *Subsystems of Second Order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, 1999. (p. 234.)

[119] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In J. Dekker, editor, *Recursive function theory*, volume 5 of *Symposia in Pure Mathematics*, pages 1–27, 1962. (pp. xiv, 103 and 179.)

[120] R. Statman. Lower bounds on Herbrand's theorem. *Proceedings of the American Mathematical Society*, 75(1):104–107, 1979. (pp. xx, 108 and 179.)

[121] M. Stein. *Interpretation der Heyting-Arithmetik endlicher Typen*. PhD thesis, Universität Münster, 1976. (pp. xi, 59 and 187.)

[122] A. Troelstra. Realisability. In [19], pages 407–473. (pp. xx and 179.)

[123] A. Troelstra, editor. *Metamathematical investigation of intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin - Heidelberg - New York, 1973. (pp. vii, xi, 16, 29, 34, 35, 59, 84, 152, 165, 195, 197, 232 and 237.)

[124] A. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2nd edition, 2000. (pp. xx and 179.)

Index of Chapters 1 and 2

Appendix A

A complexity analysis of functional interpretations

Note: This chapter is a synthesis of [57] and [56], therefore a joint work with U. Kohlenbach[1].

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.*

This chapter 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

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

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 [86] modified realizability (see [122] for a survey) and Gödel's functional interpretation (first published in [45], see [4] for a survey). The no-counterexample interpretation (*n.c.i.*) due to Kreisel [83, 84] 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 [73]).

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 [120], Orevkov [99, 100] and Pudlak [105] – see also [124] and the more recent [39, 37, 38]. 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[2] 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 [58] 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 [22, 69, 103] or – more recently – the poly-time arithmetic of [110, 115]. They were extended to analysis [28, 90, 119], type theories [42] and fragments of set theory [18]. Gödel's functional interpretation was recently adapted to yield an extraction of Herbrand terms from ordinary first-order predicate logic proofs [40].

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 trans-

[2]Normalization would bring back the aforementioned complexities.

lation 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 [26, 36] and its variants [23] [3]. Unlike realizability interpretations, functional interpretations 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 A.4). 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[4] 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 [81]).

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 chapter 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 A.2. 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

[3] See, e.g., [10, 94] 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 on the Dialectica interpretation feature full modularity: the input proofs may use arbitrary lemmas. See also [48] for applications of a form of recursive realizability.

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

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 `WKL`–elimination for Π^0_2–sentences over primitive recursive arithmetic[5] in all finite types by means of syntactic translations. 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` [116]. An experimental comparison between the performance of this and the existent refined A-translation [10] extraction module is reported in [50]. 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. [22, 110]. The issue of characterizing the complexity of provably total function(al)s of a theory is completely separate from the present chapter'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 [68]. 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[6] (some) realizers. Monotone functional interpretation extracts majorizing terms which are simpler than the actual realizers produced by functional interpretation. This

[5]This had been shown for a second–order fragment independently in [47] and [2], in the latter by means of a formalized forcing technique.

[6] Majorization is understood in the sense of Howard [58] mentioned before.

is due to the much simpler treatment of $\mathtt{CT}\wedge$, see Proposition A.37 and the paragraph following Definition A.70. Also the treatment of induction axioms is much simpler, see Section A.4. 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 A.73 .

A.0.1 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[7] 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);
- 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[8] 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 [45] and the monotone variant [68] of functional interpretation. We first consider a binary-tree representation for terms, see also Footnote 34. 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[9], see Section A.2.4 . Since their definition does not depend on the term representation, the bounds for mdg and mar still hold. From Section A.2.4 on it is tacitly assumed that terms are represented in the economic manner. A representation for types becomes necessary

[7] 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 A.2.1 for details.

[8] Proofs are represented as trees, see also the last paragraph of Section A.0.2 .

[9] 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 A.50 .

only at the moment that we are interested in the space/time overhead of the extraction algorithm, see Section A.2.5 . Let us denote by ∂ the depth and by S_i, S_c, S_m the size (in the sense of Definition A.46) 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[10] $\vee$) 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 $\mathtt{EIL}^\omega$)

- 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] We must count $\vee$ among the quantifiers because functional interpretation treats disjunction as an existential quantifier.

	usual FI	monotone FI
mdg	$O(1) + vdg_{\mathrm{o}} + id_{\mathrm{o}}$	$O(1) + vdg_{\mathrm{o}} + id_{\mathrm{o}}$
mar	$O(1) + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}}$	$O(1) + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}}$
d	$O(ld_1) + qs_{\mathrm{o}} \cdot \partial$	$O(1) + qs_{\mathrm{o}} \cdot \partial$
S	$O(S_i)\,,\, O(ls_1 \cdot qs_{\mathrm{o}}^{\partial})$	$O(S_m)\,,\, O(qs_{\mathrm{o}}^{\partial})$
∂_v	$O(ld_1 + \partial)$	$O(qs_{\mathrm{o}} + \partial)$
θ	$O(qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m)$	$O(qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m)$

where vdg_{o}, var_{o}, id_{o}, qs_{o}, ls_{o} are maxima taken over all the axioms of $\mathcal{P}$ [11] 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 A.3.1. 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	$vdg_{\mathrm{o}} + O(fid_{\mathrm{o}})$	$vdg_{\mathrm{o}} + O(fid_{\mathrm{o}})$
mar	$var_{\mathrm{o}} + O(qs_{\mathrm{o}} \cdot fid_{\mathrm{o}})$	$var_{\mathrm{o}} + O(qs_{\mathrm{o}} \cdot fid_{\mathrm{o}})$
d	$O(ls_{\mathrm{o}} \cdot \partial)$	$O(qs_{\mathrm{o}} \cdot \partial)$
S	$O(S_c)\,,\, O(ls_{\mathrm{o}} \cdot qs_{\mathrm{o}}^{k \cdot \partial})$	$O(S_m)\,,\, O(qs_{\mathrm{o}}^{k \cdot \partial})$
∂_v	$O(ls_{\mathrm{o}} + \partial)$	$O(qs_{\mathrm{o}} + \partial)$
θ	$O(qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m)$	$O(qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m)$

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 $\mathcal{P}$.

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

[11] In fact it is sufficient to consider only the axioms of the transformed proof $\mathcal{P}^{\mathrm{tr}}$, see Definition A.23. The same holds for the subsequent definitions as well, including the classical case.

terms. We rather consider them as "black boxes" and use their type and free variables information only (see Definition A.25). 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 A.41.

A.0.2 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) [12]. For a labeled tree $\triangle$ we denote by $\partial(\triangle)$ the depth of $\triangle$ plus k_0;

[12] 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.

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 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, $n \in \mathbb{N}$.

A.1 The weak base system $\mathtt{EIL}^\omega$

In the following we introduce the system $\mathtt{EIL}^\omega$ [13] 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[14], 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 A.1.3. This allows an as extensional as possible treatment of higher type equality in the context of functional interpretation[15] – see also [74].

We first carry out a full quantitative analysis for the functional interpretation of an extension $\mathtt{EIL}^\omega_+ + \mathtt{AC} + \mathtt{IP}_\forall + \mathtt{MK}$ [16] of $\mathtt{EIL}^\omega$ into the quantifier-free fragment of $\mathtt{EIL}^\omega$. Due to the modularity of functional interpretation this analysis immediately relativises to further extensions of $\mathtt{EIL}^\omega$ with certain axioms like, e.g., induction. Suppose that we consider an additional (closed) axiom

[13] Acronym for "(weakly extensional) extended intuitionistic logic in all finite types".

[14] These allow the definition of λ-terms, see Definition A.12.

[15] 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 [90] is applicable.

[16] $\mathtt{AC}$ 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 A.2.1. For $\mathtt{EIL}^\omega_+$ see Definition A.25.

A. Let us add to $\mathtt{EIL}^\omega$ new constants $\underline{c}$ of appropriate types and the axiom[17] $\forall \underline{y} A_{\mathtt{D}}(\underline{c}, \underline{y})$ 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 relativises 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 $\vdash \forall \underline{y} A_{\mathtt{D}}(\underline{s}, \underline{y})$.

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[18] 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 [109]. Our Σ provide in particular extensions of Schönfinkel's Σ to tuples (see Definition A.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[19]. 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 A.43.

[17] Here $\exists \underline{x} \forall \underline{y} A_{\mathtt{D}}(\underline{x}, \underline{y})$ is the functional interpretation of A, see also Section A.2.

[18] 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 [25] (see also [107, 121]) and Jørgensen [61]. 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 chapter can immediately be adapted to a system with λ-abstraction included as primitive construct, see Remark A.43.

[19] One example of a logical relation is Howard's majorizability which plays a key role in most applications of functional interpretation [4, 68, 81].

A.1.1 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 $\sigma_1 \dots \sigma_n o$ with $n \geq 0$. We abbreviate by:

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

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

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

A.1.2 Intuitionistic Equality Logic over FT ($\mathtt{IEL}^\omega$)

Our formalization of $\mathtt{IEL}^\omega$ below is a slight modification of the axiomatic calculus for multisorted intuitionistic predicate logic used by Gödel in his original paper on functional interpretation [45]. 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 $\mathtt{IEL}^\omega[\mathcal{C}]$ 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 $\mathtt{FT}$, 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 $\underline{x^\sigma} :\equiv x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$;
- a binary predicate constant $=_o$ for equality between objects of type o;
- logical constants $\bot,\ \wedge,\ \vee,\ \to,\ \forall x$ 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 $mdg(t) :\equiv max\{dg(s) \,|\, s \leq t\}$ and

$$mar(t) :\equiv max\{ar(s) \,|\, s \leq t\}\,.$$

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 $\bot$ according to the rule that if A and B are formulas then $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, $(\forall x A)$ and $(\exists x A)$ are formulas. Equivalence and negation of formulas are defined as $A \leftrightarrow B :\equiv ((A \to B) \wedge (B \to A))$ and respectively $\neg A :\equiv (A \to \bot)$. 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 A.0.2 for the definition of k_0)
- For $Q \in \{\forall, \exists\}$, $d_{\mathcal{S}}(Qx\,A) :\equiv \begin{cases} d_{\mathcal{S}}(A) + 1\,, & \text{if } Q \in \mathcal{S} \\ d_{\mathcal{S}}(A)\quad\;\; , & \text{if } Q \notin \mathcal{S} \end{cases}$
- For $\Box \in \{\wedge, \vee, \rightarrow\}$, $d_{\mathcal{S}}(A \Box B) :\equiv \begin{cases} max\{d_{\mathcal{S}}(A)\,,\, d_{\mathcal{S}}(B)\} + 1\,, & \text{if } \Box \in \mathcal{S} \\ max\{d_{\mathcal{S}}(A)\,,\, d_{\mathcal{S}}(B)\}\quad\;\; , & \text{if } \Box \notin \mathcal{S} \end{cases}$

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 $\mathtt{IEL}^\omega$:

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)

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.

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

Remark A.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 $(\mathtt{IEL}^\omega[\mathtt{Rl}])[\mathtt{Rl}'] :\equiv \mathtt{IEL}^\omega[\mathtt{Rl} \cup \mathtt{Rl}']$.

A.1.3 $\mathtt{EIL}^\omega$ - Extended Intuitionistic Equality Logic

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
  $\mathtt{Ax}\nu$: $\quad (x = 0 \wedge y = 0) \leftrightarrow \nu\, x\, y = 0$
  $\mathtt{AxI}$: $\quad (x = 0 \to y = 0) \leftrightarrow \mathtt{I}\, x\, y = 0$
  $\mathtt{AxE}$: $\quad x = y \leftrightarrow \mathtt{E}\, x\, y = 0$

- 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 $\Sigma^{\underline{\sigma},\underline{\delta},\tau,m}_{\underline{p},\overline{p},\underline{n},\overline{n}}$ which is defined by the following

$\mathtt{Ax}\Sigma$: $\quad \Sigma^{\underline{\sigma},\underline{\delta},\tau,m}_{\underline{p},\overline{p},\underline{n},\overline{n}}(x, \underline{y}, \underline{z}) = x(\underline{z}_0, y_1(\underline{z}^1), \underline{z}_1, \ldots, y_m(\underline{z}^m), \underline{z}_m)$

The type of Σ 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 $\{\underline{\sigma}_j :\equiv \sigma_{(p_j)_1}, \ldots, \sigma_{(p_j)_{n_j}}\}^m_{j=0}$ and $\{\underline{\sigma}^j :\equiv \sigma_{(p^j)_1}, \ldots, \sigma_{(p^j)_{nj}}\}^m_{j=1}$.

- 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

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

  Recall from Section A.0.2 that $\underline{\sigma}^p$ and $\underline{z}^p$ are 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 $\Pi^{\underline{\sigma}}_i$ of type $\underline{\sigma}\sigma_i$ which is defined by the axiom

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

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 A.3 The constants P and Π are definable in terms of Σ and also $\mathtt{0}_{\underline{\sigma}o} = \Pi^{o,\underline{\sigma}}_1 0$. We nevertheless chose to define them separately since they play a particular rôle.

Definition A.4 As particular cases of Σ we distinguish the *tuple-Schönfinkel combinators* with defining axioms of shape

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

These are generalizations of the usual[20] Schönfinkel combinators Σ to tuples and will be used in the λ-abstraction Definition A.12. The usual *Schönfinkel*

[20] For the original definition of Schönfinkel's Σ and Π see [109]. See also the last paragraph before Section A.1.1.

combinators Σ are in fact particular cases of our Σ of shape $\Sigma^{\underline{\sigma},(\delta),\tau,1}_{(1_1,1_1),(1_1),(1,0),(1)}$ with defining axioms

$$\Sigma^{\underline{\sigma},(\delta),\tau,1}_{(1_1,1_1),(1_1),(1,0),(1)}(x,y,z) \;=\; x(z,y(z))\,.$$

Also the usual *Schönfinkel projectors* Π are particular cases of our Π of shape $\Pi_1^{(\sigma_1,\sigma_2)}$ with defining axioms $\Pi_1^{(\sigma_1,\sigma_2)}(z_1,z_2)=z_1$.

Remark A.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 $\mathcal{D},\nu,\mathtt{I}$ 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 [22] and the poly-time arithmetic $\mathtt{LHA}$ of [110] [21] .

Remark A.6 The extensionality axiom

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

is derivable in $\mathtt{EIL}^\omega$ for $\underline{\sigma}\equiv o,\dots,o$, 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 [123] and [58]). 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 A.7 These hold in $\mathtt{EIL}^\omega$: $\vdash \bot \leftrightarrow 1=0$ and $\vdash x\neq 0 \leftrightarrow \mathtt{I}x1=0$.

Remark A.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:

$$\underline{s}=\underline{t} \;\vdash_k\; B_0(\underline{s})\to B_0(\underline{t}) \tag{A.1}$$

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

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

Proposition A.9 The following equalities hold:

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

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

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 A.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,

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

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

Corollary A.11 (TND and Stability for quantifier-free formulas) There exists $k \in \mathbb{N}$ constant such that for all quantifier–free A_0,

$$\begin{aligned} &\vdash_{k\cdot ld(A_0)} \quad A_0(\underline{a}) \ \vee \ \neg\, A_0(\underline{a}) \\ &\vdash_{k\cdot ld(A_0)} \quad \neg\neg\, A_0(\underline{a}) \ \to \ A_0(\underline{a}) \ . \end{aligned} \tag{A.3}$$

Proof: The principle $\mathtt{STAB_0}: \quad \neg\neg\, x = 0 \ \to \ x = 0$ follows immediately with constant-depth proof from $\mathtt{TND_0}$. Both (A.3) and (A.4) follow immediately from $\mathtt{TND_0}$ and $\mathtt{STAB_0}$ respectively by (A.3) and (A.1) .$\Box$

Definition A.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:

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

Proposition A.13 (β-reduction) There exists $k \in \mathbb{N}$ constant such that for all t and $\underline{r}$ the following holds:

$$\vdash_{k\cdot d(t)} \ (\, \lambda \underline{x}^{\underline{\sigma}}.\, t[\underline{x}]\,)\, \underline{r}^{\underline{\sigma}} =_\tau \ t[\underline{r}] \quad .$$

Proof: By straightforward induction on $d(t)$, using (A.2) when the induction step falls under (A.4). $\Box$

Proposition A.14 The following inequalities hold:

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

Proof: By structural induction on t, following Definition A.12. $\square$

Remark A.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.

A.2 A quantitative analysis of the Dialectica interpretation

Gödel's functional (*Dialectica*) interpretation/translation was first introduced in [45] and is also presented in [90](4) and [123](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 A.16 (Gödel's functional interpretation of formulas)

$$
\begin{aligned}
A^{\mathtt{D}} &:\equiv (A_{\mathtt{D}} :\equiv A) \text{ for prime formulas } A \\
(A \wedge B)^{\mathtt{D}} &:\equiv \exists \underline{x}, \underline{u} \forall \underline{y}, \underline{v}\, [(A \wedge B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}) \wedge B_{\mathtt{D}}(\underline{u}; \underline{v})] \\
(\exists z A(\underline{a}, z))^{\mathtt{D}} &:\equiv \exists z, \underline{x} \forall \underline{y}\, [(\exists z A(\underline{a}, z))_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}, z)] \\
(\forall z A(\underline{a}, z))^{\mathtt{D}} &:\equiv \exists \underline{X} \forall z, \underline{y}\, (\forall z A(\underline{a}, z))_{\mathtt{D}} \qquad \text{(A.4)} \\
&\qquad\qquad (\forall z A(\underline{a}, z))_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{X}(z); \underline{y}; \underline{a}, z) \\
(A \to B)^{\mathtt{D}} &:\equiv \exists \underline{Y}, \underline{U} \forall \underline{x}, \underline{v}\, (A \to B)_{\mathtt{D}} \qquad \text{(A.5)} \\
&\qquad\qquad (A \to B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v}) \\
(A \vee B)^{\mathtt{D}} &:\equiv \exists z^o, \underline{x}, \underline{u} \forall \underline{y}, \underline{v}\, (A \vee B)_{\mathtt{D}} \\
&\qquad (A \vee B)_{\mathtt{D}} :\equiv (z = 0 \to A_{\mathtt{D}}(\underline{x}; \underline{y})) \wedge (Iz1 = 0 \to B_{\mathtt{D}}(\underline{u}; \underline{v}))
\end{aligned}
$$

Remark A.17 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 A.18 It can be easily proved by induction on the structure of the formula A [22] that

$$qs(A^{\mathtt{D}}) \;=\; |\underline{x},\underline{y},\underline{a}| \;=\; qs(A) \tag{A.6}$$

Lemma A.19 The following hold (for k_0 see Section A.0.2 and Footnote 12):

$$\begin{aligned} dg(\mathcal{V}_{\mathrm{b}}(C^{\mathtt{D}})) &\leq vdg(C) + id(C) - k_0 + 1 \qquad (A.7)\\ ar(\mathcal{V}_{\mathrm{b}}(C^{\mathtt{D}})) &\leq var(C) + qs(C)\cdot(id(C) - k_0 + 1) \end{aligned}$$

Proof: The proof is by recursion on the structure of the formula C, following the Definition A.16. We simply notice that

- $dg(\mathcal{V}_{\mathrm{b}}(\cdot^D))$ may increase only at (A.5), with the quantity 1; (A.4) 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 (A.4) and with at most $|\underline{x},\underline{v}| \leq qs(A\to B) \leq qs(C)$ at (A.5), hence

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

□

Definition A.20 Let $\mathtt{Ax}$ be an arbitrary but fixed[23] 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 $PR[\mathrm{Tm},\mathrm{Fm}] :\equiv \{(\underline{t}, A(\underline{a})) \subseteq \mathrm{Tm}^{\leq\omega} \times \mathrm{Fm} \mid |\underline{t}| = |\underline{x}|,\ \{\underline{a}\} = \mathcal{V}_{\mathrm{f}}(A) \text{ and } \mathit{typ}(\underline{t}(\underline{a})) = \mathit{typ}(\underline{x})\}$. 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

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

- 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 $\mathrm{Tm}^{\leq\omega}$.

[22] Recall from Section A.1.3 that qs also counts the free variables.
[23] See also Definition A.25 and especially Remark A.26.

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 A.21 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 A.22 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 MP instances. Indeed, if the conclusion in any of the rules QR, EXP, IMP is non-realizer-free then also the premise must be non-realizer-free. For the MP rule, if the conclusion is non-realizer-free then also the $A \to B$ premise must be non-realizer-free.

Definition A.23 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 A.24 The proofs we consider in the sequel are realizer-free-normal if not otherwise specified. See also Remark A.45.

A.2.1 Axiom extensions of $\mathtt{EIL}^\omega$. The system $\mathtt{EIL}^\omega_+ + \mathtt{AC} + \mathtt{IP}_\forall + \mathtt{MK}$

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[24] 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 (A.4) 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 $\underline{y} \notin \mathcal{V}_\mathrm{f}(\forall \underline{x} A_0(\underline{x}))$]

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

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 A.2.3). 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

[24] 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.

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 A.25 Let $\mathtt{Ax}$ be an arbitrary but fixed set of axioms and $\mathtt{Th}_{\mathtt{rf}}$ an arbitrary but fixed set of realizer-free theorems of $\mathtt{EIL}^\omega[\mathtt{Ax}]$. We define below two extensions $\mathtt{EIL}^\omega_+$ and $\mathtt{EIL}^\omega_{\mathtt{v}}$ of $\mathtt{EIL}^\omega[\mathtt{Ax}]$. The system $\mathtt{EIL}^\omega_+$ is obtained by simply adding $\mathtt{Th}_{\mathtt{rf}}$ 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

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

together with the defining axiom

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

Let 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 $t \in \mathrm{Tm}$ by (A.8).

Remark A.26 All *arbitrary but fixed* items in the above definition will be implicitly given by their context if not explicitly described.

A.2.2 The treatment of $\mathtt{EIL}^\omega$ rules

Remark A.27 Remember that the formal proofs below are by default in $\mathtt{EIL}^\omega$. See Section A.0.2 for the definitions of Vt, Lv and ∂. The definitions of qs, ls and the other quantitative measures of terms or formulas are given in Section A.1.3. Recall that qs also counts the free variables of its argument. For the meaning of the relations *PR*, *RR* and *RTS* below see Definition A.20 .

Lemma A.28 The following hold for any proof $\mathcal{P}$:

$$\begin{aligned} qs(\mathrm{Vt}(\mathcal{P})) \;&=\; qs(\mathrm{Lv}(\mathcal{P})) \quad \text{and} \quad ls(\mathrm{Vt}(\mathcal{P})) \;=\; ls(\mathrm{Lv}(\mathcal{P})) \\ \mathcal{V}(\mathrm{Vt}(\mathcal{P})) \;&=\; \mathcal{V}(\mathrm{Lv}(\mathcal{P})) \quad \text{and} \quad \mathcal{C}(\mathrm{Vt}(\mathcal{P})) \;=\; \mathcal{C}(\mathrm{Lv}(\mathcal{P})) \end{aligned} \tag{A.8}$$

Proof: The following (in)equalities are immediate:

$$\begin{array}{rclcrcl} qs(A \to \forall \underline{z} B(\underline{z})) & = & qs(A \to B(\underline{z})) & \quad & qs(\exists \underline{z} A(\underline{z}) \to B) & = & qs(A(\underline{z}) \to B) \\ qs(A \wedge B \to C) & = & qs(A \to B \to C) & & qs(B) & \leq & qs(A \to B) \end{array}$$

It follows by structural induction on $\mathcal{P}$ that $qs(A) \leq qs(\mathrm{Lv}(\mathcal{P}))$ for any formula $A \in \mathrm{Vt}(\mathcal{P})$ and then $qs(\mathrm{Vt}(\mathcal{P})) = qs(\mathrm{Lv}(\mathcal{P}))$ is immediate. The argument for ls is identical and (A.9) has a similar proof, with $\subseteq$ instead of $\leq$. $\square$

Lemma A.29 (MP) Let $\mathfrak{A}_{\mathtt{MP}}$ be the algorithm which produces $(\underline{t}_4, B(\underline{a}')) \in PR$ from the input $(\underline{t}_1, A(\underline{a}))$, $(\underline{t}_2, \underline{t}_3, (A \to B)(\widetilde{\underline{a}})) \in PR$, where $\{\underline{a}_1\} = \{\underline{a}\} - \{\underline{a}'\}$, $\{\widetilde{\underline{a}}\} = \{\underline{a}\} \cup \{\underline{a}'\}$ and t_4 is obtained from $t_4' :\equiv \Sigma(t_3, \underline{t}_1, \underline{a}_1) = \lambda \underline{a}'.\, t_3(\widetilde{\underline{a}}, \underline{t}_1(\underline{a}))$ 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:

$$d(\underline{t}_4) \quad \leq \quad qs(A \to B) + d(\underline{t}_1, \underline{t}_3) \tag{A.9}$$

$$S(\underline{t}_4) \quad \leq \quad 1 + qs(A \to B) \cdot S(\underline{t}_1, \underline{t}_3) \tag{A.10}$$

$$dg(\underline{t}_4) \leq dg(\underline{t}_3) \quad \text{and} \quad ar(\underline{t}_4) \leq ar(\underline{t}_3)$$

$$mdg(\underline{t}_4) \quad \leq \quad max\{mdg(\underline{t}_1, \underline{t}_3)\,,\, dg(\underline{t}_3) + 1\}$$

$$mar(\underline{t}_4) \quad \leq \quad max\{mar(\underline{t}_1, \underline{t}_3)\,,\, ar(\underline{t}_3) + 1\,,\, ar(\underline{a}_1)\}$$

$$\{\!|\, \underline{t}_1\,,\, A \,|\!\} \quad , \quad \{\!|\, \underline{t}_2, \underline{t}_3\,,\, A \to B \,|\!\} \quad \vdash_k \quad \{\!|\, \underline{t}_4\,,\, B \,|\!\}$$

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:

$$\underline{y} :\equiv \underline{t}_2(\widetilde{\underline{a}}, \underline{t}_1(\underline{a}), \underline{v}) \quad \frac{\forall \underline{y}\, A_{\mathtt{D}}(\underline{t}_1(\underline{a}); \underline{y})}{A_{\mathtt{D}}(\underline{t}_1(\underline{a}); \underline{t}_2(\widetilde{\underline{a}}, \underline{t}_1(\underline{a}), \underline{v}))}\ k$$

$$\underline{x} :\equiv \underline{t}_1(\underline{a}) \quad \frac{\forall \underline{x}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{t}_2(\widetilde{\underline{a}}, \underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{t}_3(\widetilde{\underline{a}}, \underline{x}); \underline{v}))}{A_{\mathtt{D}}(\underline{t}_1(\underline{a}); \underline{t}_2(\widetilde{\underline{a}}, \underline{t}_1(\underline{a}), \underline{v})) \to B_{\mathtt{D}}(\underline{t}_3(\widetilde{\underline{a}}, \underline{t}_1(\underline{a})); \underline{v})}\ k$$

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:

$$\frac{\forall \underline{y}\, A_{\mathtt{D}}(\underline{t}_1(\underline{a}); \underline{y}) \qquad \forall \underline{x}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{t}_2(\widetilde{\underline{a}}, \underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{t}_3(\widetilde{\underline{a}}, \underline{x}); \underline{v}))}{B_{\mathtt{D}}(\underline{t}_3(\widetilde{\underline{a}}, \underline{t}_1(\underline{a})); \underline{v})}\ k$$

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, $\vdash_k t_3(\widetilde{\underline{a}}, \underline{t}_1(\underline{a})) = t_4'(\underline{a}')$ holds. Since $B_{\mathtt{D}}$ is quantifier–free, we obtain from (A.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 $B_{\mathtt{D}}(\underline{t}_3(\widetilde{\underline{a}}, \underline{t}_1(\underline{a})); \underline{v}) \vdash_k B_{\mathtt{D}}(\underline{t}_4'(\underline{a}'); \underline{v})$ 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:

$$\frac{\forall \underline{y}\, A_{\mathtt{D}}(\underline{t}_1(\underline{a}); \underline{y}) \qquad \forall \underline{x}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{t}_2(\widetilde{\underline{a}}, \underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{t}_3(\widetilde{\underline{a}}, \underline{x}); \underline{v}))}{\forall \underline{v}\, B_{\mathtt{D}}(\underline{t}_4'(\underline{a}'); \underline{v})}\ k$$

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 (A.9) and (A.10) follow from

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

The remaining inequalities follow immediately from

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

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

Lemma A.30 ($\mathtt{QR}\forall, \mathtt{QR}\exists$) 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 input $(\underline{t_1}, \underline{t_2}, A(\underline{a}) \to B(\underline{a'}, \underline{z})) \in PR$, where $t_3 :\equiv P\, t_1 = \lambda \widetilde{\underline{a}}, \underline{x}, \underline{z}.\, t_1(\widetilde{\underline{a}}, \underline{z}, \underline{x})$ and $t_4 :\equiv P\, t_2 = \lambda \widetilde{\underline{a}}, \underline{x}, \underline{z}.\, t_2(\widetilde{\underline{a}}, \underline{z}, \underline{x})$ with $\{\widetilde{\underline{a}}\} = \{\underline{a}\} \cup \{\underline{a'}\}$. There exists $k \in \mathbb{N}$ constant such that the following hold:

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

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

Proof: By definition,

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

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

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

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 (A.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,

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

Further, there exists $k \in \mathbb{N}$ constant such that for all $(\underline{t}_1, \underline{t}_2, A \to B(\underline{z})) \in PR$,

$$\frac{\forall \underline{x}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{t}_1(\widetilde{a}, \underline{z}, \underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{t}_2(\widetilde{a}, \underline{z}, \underline{x}); \underline{v}; \underline{a}', \underline{z}))}{\forall \underline{x}, \underline{z}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{t}_3(\widetilde{a}, \underline{x}, \underline{z}, \underline{v})) \to B_{\mathtt{D}}(\underline{t}_4(\widetilde{a}, \underline{x}, \underline{z}); \underline{v}; \underline{a}', \underline{z}))}\, k\ .$$

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 A.31 (EXP, IMP) The following holds:

$$\{\!|\, \underline{t}_1, \underline{t}_2, \underline{t}_3\,,\ A \to (B \to C)\, |\!\}\quad =\quad \{\!|\, \underline{t}_1, \underline{t}_2, \underline{t}_3\,,\ A \wedge B \to C\, |\!\}\ .$$

Proof: By definition,

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

□

Theorem A.32 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[25] $\partial_{\mathtt{MP}} :\equiv \partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}} :\equiv \partial_{\mathtt{QR}}(\mathcal{P})$ 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:

$$d(\underline{t}) \quad \leq \quad d_{\mathrm{o}} + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot (\partial_{\mathtt{MP}} - k_0) \tag{A.11}$$

$$S(\underline{t}) \quad \leq \quad (S_{\mathrm{o}} + \partial_{\mathtt{QR}} - k_0 + 1) \cdot qs_{\mathrm{o}}^{(\partial_{\mathtt{MP}} - k_{\mathrm{o}})} \tag{A.12}$$

$$dg(\underline{t}) \leq dg_{\mathrm{o}} \quad \text{and} \quad mdg(\underline{t}) \leq mdg_{\mathrm{o}} + 1 \tag{A.13}$$

$$ar(\underline{t}) \leq ar_{\mathrm{o}} \quad \text{and} \quad mar(\underline{t}) \leq max\{var_{\mathrm{o}}\,,\, mar_{\mathrm{o}} + 1\} \tag{A.14}$$

$$\mathtt{EIL}^{\omega}_{+} \quad \vdash_{\partial_{\mathrm{o}} + k\,\partial} \quad \{\!|\, \underline{t}\,,\ A\, |\!\}$$

[25] See Section A.0.2 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.

Proof: The algorithm proceeds by recursion on the structure of $\mathcal{P}$, using the algorithms in Lemmas A.29 and A.30 as subprocedures at the MP, respectively QR recursion steps; (A.15) follows immediately. We notice that dg and ar do not increase in the recursion, hence (A.13) and (A.14) are clear. Let $\underline{e} \equiv e_1 \dots e_n$ 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 $(d^{\underline{e}}_0\ ,\ S^{\underline{e}}_0) :\equiv (d_0\ ,\ S_0)$ and let

$$(d^{\underline{e}}_i\ ,\ S^{\underline{e}}_i) \quad :\equiv \quad \begin{cases} (d^{\underline{e}}_{i-1} + qs_0\ ,\ qs_0 \cdot S^{\underline{e}}_{i-1} + 1)\,, & \text{if } L(e_i) = \texttt{MP} \\ (d^{\underline{e}}_{i-1} + 1\ ,\ S^{\underline{e}}_{i-1} + 1) \qquad\quad , & \text{if } L(e_i) \in \texttt{QR} \\ (d^{\underline{e}}_{i-1}\ ,\ S^{\underline{e}}_{i-1}) \qquad\quad , & \text{otherwise} \end{cases} .$$

with $i \in \overline{1,n}$. Using (A.8) 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 (A.11) and (A.12) follow now immediately[26].□

Remark A.33 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_0. In consequence, (A.15) becomes $\texttt{EIL}^{\omega}_{+} \vdash_{\partial_0 + k\cdot(qs_0+\partial)} \{\!|\, \underline{t}\,,\, A \,|\!\}$.

A.2.3 Bounds for realizing terms for $\texttt{EIL}^{\omega}_{+}+\texttt{AC}+\texttt{IP}_{\forall}+\texttt{MK}$ axioms

Remark A.34 Recall that the formal proofs below are by default in $\texttt{EIL}^{\omega}$.

Proposition A.35 There exists $k \in \mathbb{N}$ constant such that for any instance A of $\texttt{CT}\vee$, $\texttt{WK}\vee$, $\texttt{WK}\wedge$, $\texttt{PM}\vee$, $\texttt{PM}\wedge$, SYL, EPN, EFQ, $\texttt{TND}_0$, MK, $\texttt{IP}_{\forall}$, AC, there exists a realizing tuple $\underline{t}$ for $A^{\mathtt{D}}$ such that:

$$\begin{aligned} d(\underline{t}) &\leq k && \text{(A.15)} \\ S(\underline{t}) &\leq k && \text{(A.16)} \\ mdg(\underline{t}) &\leq k + vdg(A) + id(A) && \text{(A.17)} \\ mar(\underline{t}) &\leq k + var(A) + qs(A)\cdot(id(A) - k_0 + 2) && \text{(A.18)} \\ \texttt{EIL}^{\omega} &\vdash_k \{\!|\, \underline{t}\,,\, A \,|\!\} \end{aligned}$$

[26] At (A.12) an intermediate upper bound is $(S_0 + \partial_{\texttt{QR}} - k_0)\cdot qs_0^{(\partial_{\texttt{MP}} - k_0)} + \sum_{i=0}^{(\partial_{\texttt{MP}} - k_0)-1} qs_0^i$.

Proof: Let below $\{\underline{a}\} \equiv \mathcal{V}_{\mathrm{f}}(\textit{the corresp. axiom instance})$. We first treat $\mathtt{SYL}$, because it is the most complex among the above listed axioms. We have:

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

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

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

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

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

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

The proofs of (A.15) and (A.16) are immediate, for (A.17) and (A.18) we use the results in Proposition A.9 plus (A.7), respectively ($A.6$, $A.8$) and for (A.19) we use $\mathtt{Ax}\Sigma$, $\mathtt{Ax}P$, $\mathtt{Ax}\Pi$, $\mathtt{Ax}\mathcal{D}$ and (A.1). We now treat the other axioms:

$\mathtt{CT}\vee$: By definition,

$$(A \vee A \to A)^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{Y'}, \underline{X''}\, \forall z, \underline{x}, \underline{x'}, \underline{y''}$$
$$\left(\begin{array}{c} \left(\begin{array}{c} z = 0 \to A_{\mathtt{D}}(\underline{x}; \underline{Y}(z, \underline{x}, \underline{x'}, \underline{y''})) \\ \wedge \\ z \neq 0 \to A_{\mathtt{D}}(\underline{x'}; \underline{Y'}(z, \underline{x}, \underline{x'}, \underline{y''})) \end{array} \right) \\ \longrightarrow \\ A_{\mathtt{D}}(\underline{X''}(z, \underline{x}, \underline{x'}); \underline{y''}) \end{array} \right)$$

and we can take $\left\{ \begin{array}{lcl} t_Y & :\equiv & t_{Y'} :\equiv \Pi = \lambda \underline{a}, z, \underline{x}, \underline{x'}, \underline{y''}.\, y'' \\ t_{X''} & :\equiv & \Pi\, \mathcal{D} = \lambda \underline{a}.\, \mathcal{D} \end{array} \right.$

$\mathtt{WK}\vee$: By definition,

$$(A \to A \vee B)^{\mathtt{D}} \equiv \exists \underline{Y}, Z, \underline{X'}, \underline{U}\, \forall \underline{x}, \underline{y'}, \underline{v}$$
$$\left(\begin{array}{c} A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{y'}, \underline{v})) \\ \longrightarrow \\ \left(\begin{array}{c} Z(\underline{x}) = 0 \to A_{\mathtt{D}}(\underline{X'}(\underline{x}); \underline{y'}) \\ \wedge \\ Z(\underline{x}) \neq 0 \to B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v}) \end{array} \right) \end{array} \right)$$

and we can take $\left\{ \begin{array}{lcl} t_Y & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{y'}, \underline{v}.\, y' \\ t_Z & :\equiv & \mathtt{0} = \lambda \underline{a}, \underline{x}.\, 0 \\ t_{X'} & :\equiv & \Pi = \lambda \underline{a}, \underline{x}.\, x \\ t_U & :\equiv & \mathtt{0} \end{array} \right.$

$\mathtt{WK}\wedge$: By definition,

$$(A \wedge B \to A)^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{V}, \underline{X'}\, \forall \underline{x}, \underline{u}, \underline{y'}$$
$$\left(\begin{array}{c} A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{u}, \underline{y'})) \wedge B_{\mathtt{D}}(\underline{u}; \underline{V}(\underline{x}, \underline{u}, \underline{y'})) \\ \longrightarrow \\ A_{\mathtt{D}}(\underline{X'}(\underline{x}, \underline{u}); \underline{y'}) \end{array} \right)$$

and we can take $\left\{ \begin{array}{lcl} t_Y & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{u}, \underline{y'}.\, y' \\ t_V & :\equiv & \mathtt{0} \\ t_{X'} & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{u}.\, x \end{array} \right.$

$\mathtt{PM}\vee$: By definition,

$$(A \vee B \to B \vee A)^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{V}, Z', \underline{U'}, \underline{X'}\ \forall z, \underline{x}, \underline{u}, \underline{v'}, \underline{y'}$$
$$\left(\begin{array}{c} \left(\begin{array}{c} z = 0 \to A_{\mathtt{D}}(\underline{x}; \underline{Y}(z, \underline{x}, \underline{u}, \underline{v'}, \underline{y'})) \\ \wedge \\ z \neq 0 \to B_{\mathtt{D}}(\underline{u}; \underline{V}(z, \underline{x}, \underline{u}, \underline{v'}, \underline{y'})) \end{array} \right) \\ \longrightarrow \\ \left(\begin{array}{c} \underline{Z'}(z, \underline{x}, \underline{u}) = 0 \to B_{\mathtt{D}}(\underline{U'}(z, \underline{x}, \underline{u}); \underline{v'}) \\ \wedge \\ \underline{Z'}(z, \underline{x}, \underline{u}) \neq 0 \to A_{\mathtt{D}}(\underline{X'}(z, \underline{x}, \underline{u}); \underline{y'}) \end{array} \right) \end{array} \right)$$

and we can take
$$\left\{ \begin{array}{lcl} t_Y & :\equiv & \Pi = \lambda \underline{a}, z, \underline{x}, \underline{u}, \underline{v'}, \underline{y'}.\, y' \\ t_V & :\equiv & \Pi = \lambda \underline{a}, z, \underline{x}, \underline{u}, \underline{v'}, \underline{y'}.\, v' \\ t_{Z'} & :\equiv & \Sigma\,(\Sigma\,(\Pi\,\mathtt{I})\,\Pi)\,(\Pi\,1) = \lambda \underline{a}, z, \underline{x}, \underline{u}.\,(\mathtt{I}\,z\,1) \\ t_{U'} & :\equiv & \Pi = \lambda \underline{a}, z, \underline{x}, \underline{u}.\, u \\ t_{X'} & :\equiv & \Pi = \lambda \underline{a}, z, \underline{x}, \underline{u}.\, x \end{array} \right.$$

$\mathtt{PM}\wedge$: By definition,

$$(A \wedge B \to B \wedge A)^{\mathtt{D}} \equiv \exists \underline{Y}, \underline{V}, \underline{U'}, \underline{X'}\ \forall \underline{x}, \underline{u}, \underline{y'}, \underline{v'}$$
$$\left(\begin{array}{c} A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{u}, \underline{y'}, \underline{v'})) \wedge B_{\mathtt{D}}(\underline{u}; \underline{V}(\underline{x}, \underline{u}, \underline{y'}, \underline{v'})) \\ \longrightarrow \\ B_{\mathtt{D}}(\underline{U'}(\underline{x}, \underline{u}); \underline{v'}) \wedge A_{\mathtt{D}}(\underline{X'}(\underline{x}, \underline{u}); \underline{y'}) \end{array} \right)$$

and we can take
$$\left\{ \begin{array}{lcl} t_Y & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{u}, \underline{y'}, \underline{v'}.\, y' \\ t_V & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{u}, \underline{y'}, \underline{v'}.\, v' \\ t_{U'} & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{u}.\, u \\ t_{X'} & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{u}.\, x \end{array} \right.$$

$\mathtt{EFQ}$: By definition,

$$(1 = 0 \to A)^{\mathtt{D}} \equiv \exists \underline{x}\, \forall \underline{y}\ (1 = 0 \to A_{\mathtt{D}}(\underline{x}; \underline{y}))$$

and we can take $t_x :\equiv \mathtt{0}$.

EPN : By definition,

$$\begin{array}{rcl}(A \to B)^{\mathtt{D}} & \equiv & \exists \underline{Y}, \underline{U}\, \forall \underline{x}, \underline{v}\, (A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v})) \\ (C \vee A)^{\mathtt{D}} & \equiv & \exists z, \underline{g}, \underline{x}\, \forall \underline{h}, \underline{y}\, ((z = 0 \to C_{\mathtt{D}}(\underline{g}; \underline{h})) \wedge (z \neq 0 \to A_{\mathtt{D}}(\underline{x}; \underline{y}))) \\ (C \vee B)^{\mathtt{D}} & \equiv & \exists z', \underline{g'}, \underline{u}\, \forall \underline{h'}, \underline{v}\, ((z' = 0 \to C_{\mathtt{D}}(\underline{g'}; \underline{h'})) \wedge (z' \neq 0 \to B_{\mathtt{D}}(\underline{u}; \underline{v})))\end{array}$$

hence

$$(C \vee A \to C \vee B)^{\mathtt{D}} \equiv \exists \underline{H}, \underline{Y}, Z', \underline{G'}, \underline{U'}\, \forall z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}$$
$$\left(\begin{array}{c} \left(\begin{array}{c} z = 0 \to C_{\mathtt{D}}(\underline{g}; \underline{H}(z, \underline{g}, \underline{x}, \underline{h'}, \underline{v})) \\ \wedge \\ z \neq 0 \to A_{\mathtt{D}}(\underline{x}; \underline{Y}(z, \underline{g}, \underline{x}, \underline{h'}, \underline{v})) \end{array} \right) \\ \longrightarrow \\ \left(\begin{array}{c} Z'(z, \underline{g}, \underline{x}) = 0 \to C_{\mathtt{D}}(\underline{G'}(z, \underline{g}, \underline{x}); h') \\ \wedge \\ Z'(z, \underline{g}, \underline{x}) \neq 0 \to B_{\mathtt{D}}(\underline{U'}(z, \underline{g}, \underline{x}); \underline{v}) \end{array} \right) \end{array} \right)$$

and further

$$((A \to B) \to (C \vee A \to C \vee B))^{\mathtt{D}} \equiv \exists \underline{X}, \underline{V}, \underline{H}, \underline{Y}, Z', \underline{G'}, \underline{U'}\, \forall \underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}$$
$$\left(\begin{array}{c} \left(\begin{array}{c} A_{\mathtt{D}}(\underline{X}(\underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}); \underline{y}(\underline{X}(\underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}), \underline{V}(\underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}))) \\ \to \\ B_{\mathtt{D}}(\underline{u}(\underline{X}(\underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v})); \underline{V}(\underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v})) \end{array} \right) \\ \longrightarrow \\ \left(\begin{array}{c} z = 0 \to C_{\mathtt{D}}(\underline{g}; \underline{H}(\underline{y}, \underline{u})(z, \underline{g}, \underline{x}, \underline{h'}, \underline{v})) \\ \wedge \\ z \neq 0 \to A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{y}, \underline{u})(z, \underline{g}, \underline{x}, \underline{h'}, \underline{v})) \end{array} \right) \\ \longrightarrow \\ \left(\begin{array}{c} Z'(\underline{y}, \underline{u})(z, \underline{g}, \underline{x}) = 0 \to C_{\mathtt{D}}(\underline{G'}(\underline{y}, \underline{u})(z, \underline{g}, \underline{x}); \underline{h'}) \\ \wedge \\ Z'(\underline{y}, \underline{u})(z, \underline{g}, \underline{x}) \neq 0 \to B_{\mathtt{D}}(\underline{U'}(\underline{y}, \underline{u})(z, \underline{g}, \underline{x}); \underline{v}) \end{array} \right) \end{array} \right)$$

and we can take
$$\left\{ \begin{array}{rcl} t_X & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}.\, x \\ t_V & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}.\, v \\ t_H & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}.\, h' \\ t_{Z'} & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, z, \underline{g}, \underline{x}.\, z \\ t_Y & :\equiv & P\, \Sigma = \lambda \underline{a}, \underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}.\, y(\underline{x}) \\ t_{G'} & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, z, \underline{g}, \underline{x}, \underline{h'}, \underline{v}.\, g \\ t_{U'} & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, z, \underline{g}.\, u \end{array} \right.$$

$\mathtt{TND}_0$: By definition,

$$[\, x = 0 \,\vee\, \neg(x = 0)\,]^{\mathtt{D}} \equiv \exists z\,[\,(z = 0 \to x = 0) \,\wedge\, (\mathtt{I}z1 = 0 \to \neg(x = 0))\,]$$

and we can take $t_z :\equiv \lambda x.\, x$.

The remaining axioms are of shape $A \to B$ such that $A^{\mathtt{D}} \equiv B^{\mathtt{D}}$ and therefore are immediately seen to be realized with projectors Π. For the reader's convenience we nevertheless give below the full details.

MK: We have $[\,\neg\neg\,\exists\underline{x}\,A_0(\underline{x})\,]^{\mathtt{D}} \equiv \exists\underline{x}\,\neg\neg\,A_0(\underline{x})$ hence

$$[\,\neg\neg\,\exists\underline{x}\,A_0(\underline{x}) \,\to\, \exists\underline{x}\,\neg\neg\,A_0(\underline{x})\,]^{\mathtt{D}} \equiv \exists\underline{X}\,\forall\underline{x}\,[\,\neg\neg\,A_0(\underline{x}) \,\to\, \neg\neg\,A_0(\underline{X}(\underline{x}))\,]$$

and we can take $t_X :\equiv \Pi = \lambda\underline{a}, \underline{x}.\, x$.

$\mathtt{IP}_\forall$: We have

$$\begin{array}{rcl}
[\,\forall\underline{x}\,A_0(\underline{x}) \,\to\, \exists\underline{y}\,B(\underline{y})\,]^{\mathtt{D}} & \equiv & \exists\underline{X},\underline{y},\underline{u}\,\forall\underline{v}\,[\,A_0(\underline{X}(\underline{v})) \,\to\, B_{\mathtt{D}}(\underline{u};\underline{v};\underline{y})\,] \\
(\,\exists\underline{y}\,[\,\forall\underline{x}\,A_0(\underline{x}) \,\to\, B(\underline{y})\,]\,)^{\mathtt{D}} & \equiv & \exists\underline{y},\underline{X},\underline{u}\,\forall\underline{v}\,[\,A_0(\underline{X}(\underline{v})) \,\to\, B_{\mathtt{D}}(\underline{u};\underline{v};\underline{y})\,]
\end{array}$$

hence $(\,[\,\forall\underline{x}\,A_0(\underline{x}) \,\to\, \exists\underline{y}\,B(\underline{y})\,] \,\to\, \exists\underline{y}\,[\,\forall\underline{x}\,A_0(\underline{x}) \,\to\, B(\underline{y})\,]\,)^{\mathtt{D}}$ is

$$\exists V, Y, X, U\,\forall x, y, u, v$$

$$\left(\begin{array}{c}
A_0(x(V(x,y,u,v))) \,\to\, B_{\mathtt{D}}(\underline{u}; V(x,y,u,v); y) \\
\longrightarrow \\
A_0(X(x,y,u,v)) \,\to\, B_{\mathtt{D}}(U(x,y,u); v; Y(x,y,u))
\end{array}\right)$$

and we can take $\left\{\begin{array}{rcl}
t_V & :\equiv & \Pi = \lambda\underline{a}, \underline{x}, \underline{y}, \underline{u}, \underline{v}.\, v \\
t_Y & :\equiv & \Pi = \lambda\underline{a}, \underline{x}, \underline{y}, \underline{u}.\, y \\
t_X & :\equiv & \Pi = \lambda\underline{a}, \underline{x}, \underline{y}, \underline{u}.\, x \\
t_U & :\equiv & \Pi = \lambda\underline{a}, \underline{x}, \underline{y}, \underline{u}.\, u
\end{array}\right.$

AC: We have

$$[\,\forall x\,\exists y\,B(\underline{x},\underline{y})\,]^{\mathtt{D}} \equiv \exists Y, \underline{U}\,\forall x, v\,B_{\mathtt{D}}(\underline{U}(x); v; x, Y(x)) \equiv [\,\exists Y\,\forall x\,B(x, Y(x))\,]^{\mathtt{D}}$$

hence $[\,\forall\underline{x}\,\exists\underline{y}\,B(\underline{x},\underline{y}) \,\to\, \exists\underline{Y}\,\forall\underline{x}\,B(\underline{x},\underline{Y}(\underline{x}))\,]^{\mathtt{D}}$ is

$$\exists X, V, Y, U\,\forall y, u, x, v$$

$$\left(\begin{array}{c}
B_{\mathtt{D}}(u(X(y,u,x,v)); V(y,u,x,v); X(y,u,x,v), y(X(y,u,x,v))) \\
\longrightarrow \\
B_{\mathtt{D}}(U(y,u,x); v; x, Y(y,u,x))
\end{array}\right)$$

and we can take $\begin{cases} t_X & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{x}, \underline{v}.\, x \\ t_V & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}, \underline{x}, \underline{v}.\, v \\ t_Y & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}.\, y \\ t_U & :\equiv & \Pi = \lambda \underline{a}, \underline{y}, \underline{u}.\, u \end{cases}$ □

Proposition A.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 A.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 A.0.1, Definition A.20 and its preceding comment. We therefore give an association of closed terms to all $\mathtt{EIL}^{\omega}_{+}$ formulas such that $\widetilde{\cdot}$ *constants* are used instead of the original building terms.

Proposition A.36 (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 (here $\{\underline{a}\} = \mathcal{V}_{\mathrm{f}}(A)$ and the $\widetilde{\cdot}$ constants in (A.23) are only those corresponding to terms occurring in A.)

$$d(t^{\mathtt{D}}_A) \quad \leq \quad k \cdot ld(A) \tag{A.19}$$

$$S(t^{\mathtt{D}}_A) \quad \leq \quad k \cdot ls(A) \tag{A.20}$$

$$mdg(t^{\mathtt{D}}_A) \quad \leq \quad k + vdg(A) + id(A) \tag{A.21}$$

$$mar(t^{\mathtt{D}}_A) \quad \leq \quad k + var(A) + qs(A) \cdot (id(A) - k_0 + 2) \tag{A.22}$$

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \quad \vdash_{k \cdot ld(A)} \quad A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) \;\leftrightarrow\; t^{\mathtt{D}}_A(\underline{x}, \underline{y}, \underline{a}) = 0 \tag{A.23}$$

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

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

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

$$\begin{array}{rcl} t^{\mathtt{D}}_{A \wedge B} & :\equiv & \Sigma\, \nu\, t^{\mathtt{D}}_A\, t^{\mathtt{D}}_B = \lambda \underline{x}, \underline{u}, \underline{y}, \underline{v}, \widetilde{\underline{a}}.\, \nu\, t^{\mathtt{D}}_A(\underline{x}, \underline{y}, \underline{a})\, t^{\mathtt{D}}_B(\underline{u}, \underline{v}, \underline{a'}) \\ t^{\mathtt{D}}_{\exists z A(\underline{a}, z)} & :\equiv & P\, t^{\mathtt{D}}_{A(\underline{a}, z)} = \lambda z, \underline{x}, \underline{y}, \underline{a}.\, t^{\mathtt{D}}_{A(\underline{a}, z)}(\underline{x}, \underline{y}, \underline{a}, z) \\ t^{\mathtt{D}}_{\forall z A(\underline{a}, z)} & :\equiv & \Sigma\, t^{\mathtt{D}}_{A(\underline{a}, z)} = \lambda \underline{X}, z, \underline{y}, \underline{a}.\, t^{\mathtt{D}}_{A(\underline{a}, z)}(\underline{X}(z), \underline{y}, \underline{a}, z) \end{array}$$

$$\begin{array}{rcl} t^{\mathtt{D}}_{A\to B} & :\equiv & \Sigma\,\Sigma\,\mathtt{I}\,t^{\mathtt{D}}_A\,t^{\mathtt{D}}_B = \lambda\underline{Y},\underline{U},\underline{x},\underline{v},\widetilde{\underline{a}}.\,\mathtt{I}\,t^{\mathtt{D}}_A(\underline{x},\underline{Y}(\underline{x},\underline{v}),\underline{a})\,t^{\mathtt{D}}_B(\underline{U}(\underline{x}),\underline{v},\underline{a}') \\ t^{\mathtt{D}}_{A\vee B} & :\equiv & \Sigma\,\Sigma\,\nu\,\mathtt{I}\,t^{\mathtt{D}}_A\,t^{\mathtt{D}}_B\,\mathtt{I}\,1 = \\ & & \quad = \lambda z,\underline{x},\underline{u},\underline{y},\underline{v},\widetilde{\underline{a}}.\,\nu\,(\mathtt{I}\,z\,t^{\mathtt{D}}_A(\underline{x},\underline{y},\underline{a}))\,(\mathtt{I}\,(\mathtt{I}\,z\,1)\,t^{\mathtt{D}}_B(\underline{u},\underline{v},\underline{a}')) \end{array}$$

The inequalities (A.19) and (A.20) are immediate, (A.21) and (A.22) follow from (A.7), respectively ($A.6, A.8$) and (A.23) follows using the axioms $\mathtt{Ax}\Sigma$, $\mathtt{AxI}$, $\mathtt{Ax}\nu$, $\mathtt{AxE}$. $\Box$

Proposition A.37 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 (below the $\widetilde{\cdot}$ constants in (A.24) are only those corresponding to terms occurring in A)

$$\begin{array}{rcl} d(\underline{t}) & \leq & k\cdot ld(A) \\ S(\underline{t}) & \leq & k\cdot ls(A) \\ mdg(\underline{t}) & \leq & k + vdg(A) + id(A) \\ mar(\underline{t}) & \leq & k + var(A) + qs(A)\cdot(id(A) - k_0 + 3) \\ \multicolumn{3}{c}{\mathtt{EIL}^{\omega}_{\mathtt{v}} \vdash_{k\cdot ld(A)} \{\!|\,\underline{t}\,,\,A\,|\!\}} \end{array} \tag{A.24}$$

Proof: We have

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

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

$$\begin{array}{rcl} t_{X'} & :\equiv & t_{X''} :\equiv \Pi = \lambda\underline{a},\underline{x}.\,x \\ t_Y & :\equiv & \Sigma\,\mathcal{D}\,t^{\mathtt{D}}_B = \lambda\underline{a},\underline{x},\underline{y'},\underline{y''}.\,\mathcal{D}(t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}),\underline{y''},\underline{y'}) \end{array}$$

We first prove (A.24). By $\mathtt{Ax}\mathcal{D}$, there exists $k \in \mathbb{N}$ constant such that for all B,

$$\left\{\begin{array}{ll} \vdash_k & t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}) \;=\; 0 \;\to\; \mathcal{D}(t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}),\underline{y''},\underline{y'}) = y'' \\ \vdash_k & \mathtt{I}\,t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a})\,1 \;=\; 0 \;\to \mathcal{D}(t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}),\underline{y''},\underline{y'}) = y' \end{array}\right.$$

and by using $\mathtt{ER}_{\mathtt{0}}$, there exists $k \in \mathbb{N}$ constant such that for all B,

$$\left\{\begin{array}{ll} \vdash_k & t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}) = 0 \;\to\; B_{\mathtt{D}}(\underline{x};\underline{\mathcal{D}}(t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}),\underline{y''},\underline{y'})) \;\to\; B_{\mathtt{D}}(\underline{x};\underline{y''}) \\ \vdash_k & \mathtt{I}\,t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a})\,1 = 0 \;\to\; B_{\mathtt{D}}(\underline{x};\underline{\mathcal{D}}(t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}),\underline{y''},\underline{y'})) \;\to\; B_{\mathtt{D}}(\underline{x};\underline{y'})\;. \end{array}\right.$$

By $\mathtt{TND}_{\mathtt{0}}$ and $\mathtt{AxI}$, there exists $k \in \mathbb{N}$ constant such that for all B,

$$\vdash_k \quad t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a}) = 0 \;\vee\; \mathtt{I}\,t^{\mathtt{D}}_B(\underline{x},\underline{y'},\underline{a})\,1 \;=\; 0\;.$$

From (A.23), there exists $k \in \mathbb{N}$ constant such that for all B,

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

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

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

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,

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

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

$$\begin{aligned} \vdash_k \quad & t_Y(\underline{a},\underline{x},\underline{y'},\underline{y''}) = \mathcal{D}(t_B^{\mathtt{D}}(\underline{x},\underline{y'},\underline{a}),\underline{y''},\underline{y'}) \\ \vdash_k \quad & t_{X'}(\underline{a},\underline{x}) = x \\ \vdash_k \quad & t_{X''}(\underline{a},\underline{x}) = x \,, \end{aligned}$$

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

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

This gives (A.24). The other inequalities follow directly from Proposition A.36. □

Proposition A.38 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 (below the $\widetilde{\cdot}$ constants in (A.29) are only those corresponding to terms occurring in A)

$$\begin{aligned} d(\underline{t}) \quad &\leq \quad k + fd(A), \text{ when } A(\underline{s}) \in \mathtt{QA}\forall \text{ and} && \text{(A.25)} \\ d(\underline{t}) \quad &\leq \quad k, \text{ when } A(\underline{s}) \in \mathtt{QA}\exists \\ S(\underline{t}) \quad &\leq \quad k + fd(A), \text{ when } A(\underline{s}) \in \mathtt{QA}\forall \text{ and} && \text{(A.26)} \\ S(\underline{t}) \quad &\leq \quad k, \text{ when } A(\underline{s}) \in \mathtt{QA}\exists \\ mdg(\underline{t}) \quad &\leq \quad k + vdg(A) + id(A) && \text{(A.27)} \\ mar(\underline{t}) \quad &\leq \quad k + var(A) + qs(A) \cdot (id(A) - k_0 + 2) && \text{(A.28)} \\ \mathtt{EIL}_{\mathtt{v}}^{\omega} \quad &\vdash_k \quad \{\!|\, \underline{t} \,,\, A \,|\!\} \end{aligned}$$

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 $\underline{a} :\equiv \underline{a'}, \underline{a''} = \mathcal{V}_{\mathrm{f}}(A(\underline{s}))$. 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 A.25 that $\widetilde{s_i}(\underline{a_i}) = s_i$

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

From Proposition A.9, $\mathit{typ}(u_i) = \mathit{typ}(\widetilde{s_i})$ and (A.8) it immediately follows that

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

and (A.27), (A.28) now follow immediately from (A.7), respectively ($A.6, A.8$), also using that $|\underline{z}| = |\underline{s}|$ and $\mathit{typ}(z_i) = \mathit{typ}(s_i)$. The inequalities (A.25) and (A.26) are immediate from $|\underline{s}| \leq fd(A)$. The proof of (A.29) uses the fact (which follows from (A.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 A.25 that $\widetilde{s_i}(\underline{a_i}) = s_i$)

$$\begin{array}{rcl} t_Y & :\equiv & \Pi = \lambda \underline{a}, \underline{x}, \underline{y}.\, y \\ t_{Z_i} & :\equiv & \Sigma\, \widetilde{s_i} = (\lambda u_i, \underline{a}, \underline{x}.\, u_i(\underline{a_i}))\widetilde{s_i} = \lambda \underline{a}, \underline{x}.\, \widetilde{s_i}(\underline{a_i}) \\ t_X & :\equiv & \Pi = \lambda \underline{a}, \underline{x}.\, x \end{array}$$

The inequalities (A.25) and (A.26) are trivial, (A.27), (A.28) follow with an argument similar to the one in the $\mathtt{QA}\forall$ case. For (A.29) we use the fact (which follows from (A.1)) that there exists $k \in \mathbb{N}$ constant such that for all $A(\underline{s})$,

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

□

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

$$\begin{array}{rclrcl}
vdg_{\mathrm{o}}(\mathcal{P}) & :\equiv & vdg(\mathrm{Lv}(\mathcal{P})) & var_{\mathrm{o}}(\mathcal{P}) & :\equiv & var(\mathrm{Lv}(\mathcal{P})) \\
fd_1(\mathcal{P}) & :\equiv & fd(\mathtt{QA}\forall \cap \mathrm{Lv}(\mathcal{P})) & id_{\mathrm{o}}(\mathcal{P}) & :\equiv & id(\mathrm{Lv}(\mathcal{P})) \\
ld_1(\mathcal{P}) & :\equiv & ld(\mathtt{CT}\wedge \cap \mathrm{Lv}(\mathcal{P})) & ls_1(\mathcal{P}) & :\equiv & ls(\mathtt{CT}\wedge \cap \mathrm{Lv}(\mathcal{P})) \\
fid_{\mathrm{o}}(\mathcal{P}) & :\equiv & fid(\mathrm{Vt}(\mathcal{P})) & ls_{\mathrm{o}}(\mathcal{P}) & :\equiv & ls(\mathrm{Lv}(\mathcal{P}))
\end{array}$$

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

Theorem A.39 There exists $k \in \mathbb{N}$ constant such that for any proof $\mathcal{P}$ in $\mathtt{EIL}^{\omega}_{+}+\mathtt{AC}+\mathtt{IP}_{\forall}+\mathtt{MK}$ 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 $(\mathtt{CT}\wedge, \mathtt{QA}\forall)$ then

$$\left.\begin{array}{rcl}
d(\underline{t}_A) & \leq & k \\
S(\underline{t}_A) & \leq & k \\
mdg(\underline{t}_A) & \leq & k + vdg_{\mathrm{o}} + id_{\mathrm{o}} \\
mar(\underline{t}_A) & \leq & k + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}} \\
\mathtt{EIL}^{\omega}_{\mathtt{v}} \ \vdash_k & \{\!|\, \underline{t}_A\,,\, A \,|\!\}
\end{array}\right\}$$

- if A is an instance of $\mathtt{CT}\wedge$, (A.29) 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 $\mathtt{QA}\forall$, (A.29) holds except that $d(\underline{t}_A) \leq k + fd_1$ and $S(\underline{t}_A) \leq k + fd_1$.

The $\widetilde{\cdot}$ constants of $\mathtt{EIL}^{\omega}_{\mathtt{v}}$ above[27] 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 A.35, A.37, A.38 and $k_0 \geq 10$.
□

[27] See also Definition A.25 and Remark A.26.

Theorem A.40 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 and, with the notations 2 and abbreviations[28] $\partial_{\mathtt{MP}} :\equiv \partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}} :\equiv \partial_{\mathtt{QR}}(\mathcal{P})$ and $\partial :\equiv \partial(\mathcal{P})$, the following hold:

$$\begin{aligned} d(\underline{t}) &\leq k \cdot ld_{1} + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot \partial_{\mathtt{MP}} && \text{(A.29)} \\ S(\underline{t}) &\leq (k \cdot ls_{1} + \partial_{\mathtt{QR}}) \cdot qs_{\mathrm{o}}^{\partial_{\mathtt{MP}}} && \text{(A.30)} \\ mdg(\underline{t}) &\leq k + vdg_{\mathrm{o}} + id_{\mathrm{o}} && \text{(A.31)} \\ mar(\underline{t}) &\leq k + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}} && \text{(A.32)} \end{aligned}$$

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \vdash_{k\cdot(ld_{1}+\partial)} \{\!|\, \underline{t}\,,\, A \,|\!\}$$

The $\widetilde{\cdot}$ constants of $\mathtt{EIL}^{\omega}_{\mathtt{v}}$ in (A.33) are among those corresponding to terms occurring in the leaves of $\mathcal{P}$.

Proof: Just a synthesis of the results in Theorems A.32 and A.39. For (A.29) and (A.30) we use that $fd_{1} \leq qs_{\mathrm{o}}$ and $k_{0} \geq 10$, hence

$$\begin{aligned} &max\{k \cdot ld_{1}\,,\, k + fd_{1}\} + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot (\partial_{\mathtt{MP}} - k_{0}) \;\leq k \cdot ld_{1} + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot \partial_{\mathtt{MP}} \\ &(max\{k \cdot ls_{1}\,,\, k + fd_{1}\} + qs_{\mathrm{o}} + \partial_{\mathtt{QR}} - k_{0} + 1) \cdot qs_{\mathrm{o}}^{(\partial_{\mathtt{MP}} - k_{\mathrm{o}})} \;\leq\; (k \cdot ls_{1} + \partial_{\mathtt{QR}}) \cdot qs_{\mathrm{o}}^{\partial_{\mathtt{MP}}} \end{aligned}$$

□

Notation. 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 A.41 Theorem A.40 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 (A.31), (A.32) must be replaced with

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

[28] See Footnote 25 for the meaning of $\partial_{\mathtt{MP}}(\mathcal{P})$, $\partial_{\mathtt{QR}}(\mathcal{P})$ and $\partial(\mathcal{P})$.

Corollary A.42 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

$$\mathtt{EIL}^{\omega}_{\mathtt{v}}+\mathtt{AC}+\mathtt{IP}_{\forall}+\mathtt{MK} \quad \vdash_{k' \cdot max\{ld_1+\partial\,,\, ld(B)\}} \quad \forall \underline{x}\; B(\underline{x}, \underline{t_Y}(\underline{x}))$$

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 A.40 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

$$\mathtt{EIL}^{\omega}_{\mathtt{v}} \quad \vdash_{k\cdot(ld_1+\partial)} \quad \exists \underline{u}\, \forall \underline{v}\; B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{x}, \underline{t_Y}(\underline{x})) \quad [\, \equiv B^{\mathtt{D}}(\underline{x}, \underline{t_Y}(\underline{x}))\,]$$

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

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

The conclusion now follows immediately by combining (A.33) and (A.33) . □

Remark A.43 If λ-abstraction were treated as primitive and Σ, P, Π were defined in terms of it then (A.30) 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 $|x, \underline{y}, \underline{z}| \leq qs_{\mathrm{o}}$ for all Σ which appear in $\underline{t}$. Similarly (A.30) would still hold if only Schönfinkel Σ and Π were allowed[29]. This follows from the λ-abstraction Definition A.12 . There exists $k \in \mathbb{N}$ constant such that at most $k \cdot |x, \underline{y}, \underline{z}|^{\,2} \leq k \cdot {qs_{\mathrm{o}}}^2$ tuple-Schönfinkel Σ and Π are needed to simulate our Σ and any of these tuple-Schönfinkel Σ and Π can be defined[30] in terms of at most $k \cdot |x, \underline{y}, \underline{z}| \leq k \cdot qs_{\mathrm{o}}$ usual Schönfinkel Σ and Π .

Remark A.44 If we allowed only unary (see Remark A.33) $\mathtt{ER}_{\mathtt{0}}$ in the verifying proof then (A.33) would become $\mathtt{EIL}^{\omega}_{\mathtt{v}} \vdash_{k\cdot(ld_1+qs_{\mathrm{o}}+\partial)} \{\!|\, \underline{t}\,,\, A \,|\!\}$.

[29] See Definition A.4 for the notions of "tuple-Schönfinkel" and "Schönfinkel" combinators Σ . Also for "Schönfinkel" projectors Π .

[30] For Σ the proof is by induction on $|\underline{z}|$ of Definition A.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 $\lambda z_1.\, (\ldots \lambda z_n.\, z_i)$.

Remark A.45 The algorithm of Theorem A.40 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 A.23. Since $\mathtt{IEL}^{\omega} \vdash A \leftrightarrow A^{\mathtt{D}}$ 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.

A.2.4 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 A.40 remain valid. The simplification is based on the observation that all terms t_4 produced by $\mathtt{MP}$ (see Lemma A.29) 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, $(\underline{t_2}, \underline{t_3})$ Dr $(A \to B)$ and $(\underline{t_5}, \underline{t_6})$ Dr $(B \to C)$. The algorithm in Lemma A.29 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 A.46 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 A.3 below). The measure $S_m(\mathcal{P})$ will be used also for the time upper bounds (see Section A.2.5) 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 $S_m(\mathcal{P}) \leq S_i(\mathcal{P}) \leq S_c(\mathcal{P})$, 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 A.74).

Definition A.47 For the tuples $\underline{t} \equiv t_1, \ldots, t_n$ extracted by the algorithm of Theorem A.40 we define a *size* measure, denoted $Sz'(\underline{t})$ in the following way. There exists $m \geq 0$ 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

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

Lemma A.48 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 A.49 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 A.48 and transitivity of equality. □

Theorem A.50 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 (A.33) and (below $\#_{\mathtt{MP}}$ denotes the number of MP instances in $\mathcal{P}$)

$$\begin{aligned} Sz'(\underline{t}) &\leq k \cdot S_i(\mathcal{P}) && \text{(A.33)}\\ 3\,\#_{\mathtt{MP}} &\leq d(\underline{t}) && \text{(A.34)}\\ \partial_{\mathtt{MP}} - k_0 &\leq mdg(\underline{t}) && \text{(A.35)}\\ 3\,\#_{\mathtt{MP}} &\leq mar(\underline{t}) \end{aligned}$$

Proof: The proof of (A.33) 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 $\mathtt{QA}\forall$, (A.33) follows from $|\underline{t}| \leq qs(A)$. If $A \equiv B \to B \wedge B$ then we notice that $t^{\mathtt{D}}_B$ of Proposition A.37 is common to all realizing t_Y, hence using (A.6) and (A.20),

$$Sz'(\underline{t_{X'}}, \underline{t_{X''}}, \underline{t_Y}) \leq k' \cdot |\underline{Y}, \underline{X'}, \underline{X''}| + S(t^{\mathtt{D}}_B) \leq k' \cdot qs(A) + k'' \cdot ls(A) \leq k \cdot ls(A)$$

If $A \equiv \forall \underline{z} B(\underline{z}, \underline{a''}) \to B(\underline{s}, \underline{a''})$ then the tuple $\widetilde{\underline{s}}$ of Proposition A.38 is common to all realizing t_X, hence $Sz'(\underline{t_Z}, \underline{t_Y}, \underline{t_X}) \leq k' \cdot |\underline{Z}, \underline{Y}, \underline{X}| + |\widetilde{\underline{s}}| \leq k \cdot qs(A)$. There is nothing to prove for instances of EXP and IMP, see Lemma A.31. For QR instances the proof is trivial using Lemma A.30. For instances of MP we use Lemma A.29 and further improve the result by applying a number of Σ definitional equations. The algorithm in Lemma A.29 is presented with the tuples $\underline{t_3}$ and $\underline{t_1}$, represented [31] as

$$\left.\begin{array}{rcccl} t_3 & \equiv & P'_1(\ldots P'_{m'}(t_0(\underline{t'}))) & = & P'(t_0(\underline{t'}))\\ t^i_1 & \equiv & P^i_1(\ldots P^i_m(t^i(\underline{t}))) & = & P_i(t^i(\underline{t})) \end{array}\right\} \text{ Using Lemma A.49}$$

[31] 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 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 A.47. 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 (A.36) below is nevertheless not affected by this technical exception. After an MP all terms of the realizing tuple share a common tuple.

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

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

hence we can actually take

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

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

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

where for the last inequality we used that $|\underline{t_4}| + max\{1, |\underline{0}|\} \leq qs(A \to B)$. 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 $|\mathtt{cp}(\underline{t_1})| + |\mathtt{cp}(\underline{t_3})| + 1 \leq |\mathtt{cp}(\underline{t_4})|$ 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 $\#_{\mathtt{MP}} \leq |\mathtt{cp}(\underline{t})|$. Now (A.34) and (A.36) are immediate because $|\mathtt{cp}(\underline{t})| \leq d(\underline{t})$ and $|\mathtt{cp}(\underline{t})| \leq mar(\underline{t})$. Also (A.35) 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 $dg(t^i) \geq dg(\underline{t}) + 1$.

The proof that (A.33) 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 A.51 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, $Sz(\underline{t}) \leq k \cdot S_i(\mathcal{P})$ and the inequalities of Theorem A.40 all hold.

Remark A.52 The following inequalities are immediate:

$$\begin{array}{rcl} S(t) & \leq & Sz(\underline{t}) \\ S_i(\mathcal{P}) & \leq & (ls_{_1}(\mathcal{P}) + qs_{_0}(\mathcal{P})) \cdot 2^{\partial(\mathcal{P})} \ \leq \ 3 \cdot ls_{_1}(\mathcal{P}) \cdot qs_{_0}(\mathcal{P})^{\partial(\mathcal{P})} \ . \end{array}$$

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

Remark A.53 We will tacitly assume in the sequel that terms are represented with pointers in the manner described above.

A.2.5 Space and time complexity of the functional interpretation algorithm

In a real-world implementation of the algorithm of Theorem A.40 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[32] 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* [33]. We choose dags instead of normal binary trees[34] 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 $\mathcal{V}_{\mathrm{b}}(C^{\mathtt{D}})$ is at most $k' \cdot qs(C)^2 \cdot ls(C)$. Hence the number of new type-nodes required by the

[32] The space complexity follows immediately by the principle that the space overhead of an algorithm is always less than its time overhead.

[33] Here "dag" is the usual acronym for "directed acyclic graph".

[34] The representation with binary trees is in fact equivalent to the usual parenthesized-strings representation.

new variables created in the interpretation of the leaves, right MP–premises and QR–conclusions of $\mathcal{P}$ is at most $k' \cdot qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m(\mathcal{P})$. Then we can immediately see that whenever a new constant c of type $\underline{\sigma}\tau$ is created by the algorithm of Theorem A.40, 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 MP–right–premise C or instance of 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 $k'' \cdot qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m(\mathcal{P})$ newly created type-nodes in this category. We can now state the following theorem.

Theorem A.54 There exists $k \in \mathbb{N}$ constant such that the time overhead of the algorithm in Theorem A.51 is upper bounded by $k \cdot qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m(\mathcal{P})$.

A.3 Immediate extensions of the quantitative analysis

A.3.1 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 [35]. Negative translations were initially produced by Gödel [44], 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 A.55 (Kuroda's N-translation, adapted) To a formula A one associates $A^{\mathtt{N}} \equiv \neg\neg A^*$, where A^* is so defined by structural induction on A :

- $A^* :\equiv A$, if A is a prime formula

[35] 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.

- $(A \Box B)^* \;:\equiv\; A^* \Box B^*$, 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 A.56 $A^{\mathtt{N}}$ is realizer-free iff A is realizer-free.

N-translation followed by functional interpretation gives a proof interpretation for theories based on classical logic [36]. Remark A.56 implies that given a (complete) proof $\mathcal{P}$ in some classical system the following are equivalent:

- carry out the composed[37] interpretation to $\mathcal{P}^{\mathrm{tr}}$;
- first do the N-translation of $\mathcal{P}$, then apply the realizer-free-elimination algorithm of Definition A.23 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.

Let $\mathtt{ECL}^{\omega}, \mathtt{ECL}^{\omega}_{+}$ be the classical versions[38] 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

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

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

Remark A.57 The proof-size measure S_c is introduced in Definition A.46.

Proposition A.58 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. $\partial(\mathcal{P}^{\mathtt{N}}) \;\le\; k \cdot \partial(\mathcal{P})$ and $S_i(\mathcal{P}^{\mathtt{N}}) \;\le\; k \cdot S_c(\mathcal{P})$;
2. $$\begin{cases} qs_{\mathtt{o}}(\mathcal{P}^{\mathtt{N}}) \;\le\; qs(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;\le\; k \cdot qs(\mathrm{Vt}(\mathcal{P})) \;\overset{(A.8)}{=}\; k \cdot qs_{\mathtt{o}}(\mathcal{P}) \\ ld_{\mathtt{1}}(\mathcal{P}^{\mathtt{N}}) \;\le\; ls_{\mathtt{1}}(\mathcal{P}^{\mathtt{N}}) \;\le\; ls(\mathrm{Vt}(\mathcal{P}^{\mathtt{N}})) \;\le\; k \cdot ls(\mathrm{Vt}(\mathcal{P})) \;\overset{(A.8)}{=}\; k \cdot ls_{\mathtt{o}}(\mathcal{P}) \end{cases}$$

[36] Details of the use of negative translation in combination with functional interpretation may be found, e.g., in [4, 63, 90].

[37] In fact parts which are produced by N-translation also need to be transformed.

[38] 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 A.56 it should be obvious that A is a realizer-free axiom from $\mathtt{Th}_{\mathtt{rf}}$ of $\mathtt{ECL}^{\omega}_{+}$ (see Definition A.25) if and only if $A^{\mathtt{N}}$ is a realizer-free axiom from $\mathtt{Th}_{\mathtt{rf}}$ of $\mathtt{EIL}^{\omega}_{+}$.

3. $id_{\mathrm{o}}(\mathcal{P}^{\mathtt{N}}) \leq k \cdot fid_{\mathrm{o}}(\mathcal{P})$; we must use $fid_{\mathrm{o}}(\mathcal{P})$ 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 (A.9))

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

Proof: The algorithm proceeds by recursion on the structure of $\mathcal{P}$, see [63] for details. The proof of its correctness makes use of the following schemata of intuitionistic logic :

$$\neg\neg(A \rightarrow B) \leftrightarrow (A \rightarrow \neg\neg B) \leftrightarrow (\neg\neg A \rightarrow \neg\neg B) \tag{A.36}$$

$$\neg\neg\forall \underline{x} \neg\neg A(\underline{x}) \leftrightarrow \forall \underline{x} \neg\neg A(\underline{x}) \tag{A.37}$$

$$A \rightarrow \neg\neg A$$

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 $\mathtt{ECL}^{\omega}_{+}+\mathtt{AC}_{\mathtt{0}}$ is a theorem in $\mathtt{EIL}^{\omega}_{+}+\mathtt{AC}_{\mathtt{0}}+\mathtt{MK}$ 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 ;

- any rule $A_1\,[,\,A_2] \vdash B$ of $\mathtt{ECL}^{\omega}_{+}+\mathtt{AC}_{\mathtt{0}}$ 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 $\mathtt{AC}_{\mathtt{0}}$ and $\mathtt{QR}\forall$. The other axioms and rules are even easier.

Case $\mathtt{AC}_{\mathtt{0}}$: We prove that there exists $k' \in \mathbb{N}$ constant such that for all A_0 ,

$$\mathtt{EIL}^{\omega}_{+}+\mathtt{AC}_{\mathtt{0}}+\mathtt{MK} \quad \vdash_{k'} \quad [\,\forall \underline{x}\,\exists \underline{y}\,A_0(\underline{x},\underline{y}) \rightarrow \exists \underline{Y}\,\forall \underline{x}\,A_0(\underline{x},\underline{Y}(\underline{x}))\,]^{\mathtt{N}}$$

By (A.38), the conclusion of (A.38) 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 (A.36) 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 (A.37) 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 (A.38)). Hence overall the deduction of $(B \rightarrow \forall \underline{z} A)^{\mathtt{N}}$ from $(B \rightarrow A)^{\mathtt{N}}$ has constant depth. $\Box$

Remark A.59 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 Π instead of the terms t_Z of Proposition A.38.

Remark A.60 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

$$(\neg\neg A \;\rightarrow\; \neg\neg B) \;\rightarrow\; \neg\neg(A \;\rightarrow\; B)$$

of (A.36). The use of (A.38) can be avoided in the case of $\mathtt{IMP}$, $\mathtt{EXP}$ by using axiom versions of these rules[39], the non-critical converse of (A.38) and $\mathtt{MP}$.

Remark A.61 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 A.58. In a second phase $(\mathcal{P}^{\mathrm{tr}})^{\mathtt{N}}$ is transformed[40] to $((\mathcal{P}^{\mathrm{tr}})^{\mathtt{N}})^{\mathrm{tr}}$ and the algorithm of Theorem A.40 is applied to it. Using Proposition A.58, Theorems A.51 and A.54, Notation 2 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.

[39] The axiom versions of $\mathtt{IMP}$ and $\mathtt{EXP}$ are simply realized with projectors Π. 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 A.31.

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

Theorem A.62 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:

$$d(\underline{t}) \leq k \cdot (ls_{\mathrm{o}} + qs_{\mathrm{o}} \cdot \partial) \tag{A.38}$$

$$S(\underline{t}) \leq Sz(\underline{t}) \leq k \cdot S_c \leq k \cdot (ls_{\mathrm{o}} + \partial) \cdot (k \cdot qs_{\mathrm{o}})^{k \cdot \partial} \tag{A.39}$$

$$mdg(\underline{t}) \leq vdg_{\mathrm{o}} + k \cdot fid_{\mathrm{o}} \tag{A.40}$$

$$mar(\underline{t}) \leq var_{\mathrm{o}} + k \cdot qs_{\mathrm{o}} \cdot fid_{\mathrm{o}} \tag{A.41}$$

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

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 $\mathtt{EIL}^{\omega}_{\mathtt{v}}$ in (A.42) are among those corresponding to terms occurring in the leaves of $\mathcal{P}^{\mathtt{N}}$.

Remark A.63 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 A.40. 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 A.64 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 (A.38), (A.39), (A.40), (A.41) 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 (A.37) and (A.4),

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

From (A.8) it follows that the quantity $ld(A_0)$ gets absorbed into ls_{o}. □

A.3.2 A quantitative analysis of the monotone Dialectica

The second author realized in [68] 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

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

Here $\underline{y}\ maj\ \underline{x} :\equiv \wedge (y\ maj\ x)$ and maj is W.A. Howard's majorization relation (see [58]). 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 [64]) to the proof mining of concrete proofs [81]. We noticed in Section A.2 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 A.65 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

$$x \geq_{\underline{\sigma o}} y \;:\equiv\; \forall \underline{z^\sigma}\,(x\,\underline{z} \;\geq_o\; x\,\underline{z})\,.$$

  The *majorization* relation is defined by

$$x^*\ maj_{\underline{\sigma o}}\ x \;:\equiv\; \forall \underline{z^\sigma}, \underline{y^\sigma}\ (\underline{z}\ maj_{\underline{\sigma}}\ \underline{y} \;\to\; x^*\,\underline{z}\ maj_o\ x\,\underline{y})\,,$$

  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
  $\mathtt{Ax}\mathcal{M}:\quad \mathcal{M}_o\,x\,y \;\geq_o\; x \qquad \mathcal{M}_o\,x\,y \;\geq_o\; y \qquad \mathcal{M}_o\ maj\ \mathcal{M}_o\ .$
  Maximum constants for higher types are defined by

$$\mathcal{M}_{\underline{\sigma o}} \quad :\equiv \quad \Sigma\,\mathcal{M}_o = \lambda x^{\underline{\sigma o}}, y^{\underline{\sigma o}}, \underline{z^\sigma}.\,\mathcal{M}_o\,(x\,\underline{z})\,(y\,\underline{z})\ .$$

- 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 A.66 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 A.65 become provable (see also Remark A.5).

Remark A.67 The formulas $\Sigma\ maj\ \Sigma$, $\Pi\ maj\ \Pi$ and $P\ maj\ P$ hold in $\mathtt{EIL}^\omega_\mathtt{M}$ with proofs of depths proportional with the arities of Σ, Π 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 A.68 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

$$\mathtt{EIL}^{\omega}_{\mathtt{M}} \quad \vdash \quad \underline{x^*}\ maj\ \underline{x} \ \rightarrow\ \underline{s^*}\ maj\ \underline{s}\ .$$

Proof: The constants $\mathtt{0}$ and $\mathtt{S}$ trivially majorize themselves by the last clause of Definition A.65. On the other hand, $\Sigma\mathcal{M} = \lambda z, \underline{x}, \underline{x'}.\, \mathcal{M}\, x\, x'$ majorizes $\mathcal{D}$ and $\Pi 1 = \lambda x^o, y^o.\, 1$ majorizes $\mathtt{I}$, ν and $\mathtt{E}$. Using Remark A.67 we have that Σ, Π, P and $\mathcal{M}$ majorize themselves. The conclusion follows immediately by induction on $d(\underline{s})$. $\square$

Corollary A.69 Let $\widetilde{s}, \widetilde{s^*}$ be constants associated to terms s, s^* like in Definition A.25. From (A.42) it immediately follows that

$$\vdash \quad \underline{\widetilde{s^*}}\ maj\ \underline{\widetilde{s}}$$

Definition A.70 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 (A.42) is not created by monotone functional interpretation. Also let $\mathtt{EIL}^{\omega}_{\mathtt{M},\mathtt{v}}$ be the corresponding $(\mathtt{EIL}^{\omega}_{\mathtt{M}})_{\mathtt{v}}$.

In [68] 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

- $A \rightarrow A \wedge A$ 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 A.37;
- $A \vee A \rightarrow A$ is realized by $\Sigma\mathcal{M} = \lambda\underline{a}, z, \underline{x}, \underline{x'}.\, \mathcal{M}\, x\, x'$ and Π;
- $A \vee B \rightarrow B \vee A$ is realized by terms Π and $\Pi\, 1$;
- the schema $\mathtt{ED}$ itself is trivially realized by $\Pi\, 1 = \lambda\underline{a}.\, 1$;
- $\forall\underline{z} A(\underline{z}) \rightarrow A(\underline{s})$ 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 A.68.

Using Remark A.67 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 $\mathtt{ED}$.

Remark A.71 The proof-size measure S_m is introduced in Definition A.46 . The proof-depth measures $\partial_{\mathtt{MP}}$, $\partial_{\mathtt{QR}}$ and ∂ are introduced in Section A.0.2 . 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 A.51 holds.

Theorem A.72 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 2, the following hold :

$$\begin{array}{rcl} d(\underline{t}) & \leq & k + \partial_{\mathtt{QR}} + qs_{\mathrm{o}} \cdot \partial_{\mathtt{MP}} \\ S(\underline{t}) \leq Sz(\underline{t}) & \leq & min\{k \cdot S_m \, , \, k \cdot \partial_{\mathtt{QR}} \cdot qs_{\mathrm{o}}^{\partial_{\mathtt{MP}}}\} \leq k \cdot qs_{\mathrm{o}}^{\partial} \\ mdg(\underline{t}) & \leq & k + vdg_{\mathrm{o}} + id_{\mathrm{o}} \\ mar(\underline{t}) & \leq & k + var_{\mathrm{o}} + qs_{\mathrm{o}} \cdot id_{\mathrm{o}} \end{array} \tag{A.42}$$

$$\mathtt{EIL}^{\omega}_{\mathtt{M},\mathtt{v}} \vdash_{k\cdot(qs_{\mathrm{o}}+\partial)} \exists \underline{x} \left(\underline{t^*} \; maj \; \underline{x} \wedge \forall \underline{a}, \underline{y} \, A_{\mathtt{D}}(\underline{x}(\underline{a}), \underline{y}, \underline{a}) \right)$$

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 $\mathtt{EIL}^{\omega}_{\mathtt{M},\mathtt{v}}$ in (A.43) are among those corresponding to terms occurring in the leaves of $\mathcal{P}$.

Proof: The rightmost inequality of (A.42) follows from a suitable adaptation of Remark A.52 to the monotone case. We now only need to comment on (A.43). In order to build the verifying proof for $\mathtt{MP}$ we need to use the following lemma :

$$(\underline{y_1} \; maj \; \underline{x_1}) \wedge (\underline{y_3} \; maj \; \underline{x_3}) \quad \rightarrow \quad \wedge [\, \Sigma(y_3, \underline{y_1}, \underline{0}) \; maj \; \Sigma(x_3, \underline{x_1}, \underline{0}) \,]$$

Using Remark A.67 it follows that there exists $k' \in \mathbb{N}$ such that for all its instances, lemma (A.43) has a proof of depth at most $k' \cdot |y_3, \underline{y_1}, \underline{0}|$. When used for verifying $\mathtt{MP}$, we have $|y_3, \underline{y_1}, \underline{0}| \leq qs_{\mathrm{o}}$, hence (A.43) follows immediately. □

Remark A.73 If (A.43) 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 (A.43), $\Sigma \; maj \; \Sigma$, $\Pi \; maj \; \Pi$, $P \; maj \; P$ 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 (A.43) could be replaced with

$$\mathtt{EIL}^{\omega}_{\mathtt{M,v}} \vdash_{k\cdot\partial} \exists \underline{x} \left(\underline{t^*}\ maj\ \underline{x} \wedge \forall \underline{a}, \underline{y}\, A_{\mathtt{D}}(\underline{x}(\underline{a}), \underline{y}, \underline{a}) \right).$$

Hence the bound on verifying proof depth would be better than in the usual functional interpretation case, see (A.33). 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 $\mathtt{ER}_{\mathtt{0}}$.

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 $\mathtt{ECL}^{\omega}_{\mathtt{M},+}+\mathtt{AC}_{\mathtt{0}}$ and the upper bounds on size and proof depth are smaller than in the functional interpretation case. The following analogue of Theorem A.62 + Corollary A.64 holds.

Theorem A.74 There exists $k \in \mathbb{N}$ constant and an algorithm which does the following. Given as input a proof $\mathcal{P}$ of A in $\mathtt{ECL}^{\omega}_{\mathtt{M},+}+\mathtt{AC}_{\mathtt{0}}$, it produces as output $\underline{t^*}$ such that, with the notations 2 and the abbreviations $\partial :\equiv \partial(\mathcal{P})$ and $S_m :\equiv S_m(\mathcal{P})$ the following hold:

$$\begin{aligned}
d(\underline{t^*}) &\leq k \cdot qs_{\mathrm{o}} \cdot \partial \\
S(\underline{t^*}) &\leq Sz(\underline{t^*}) \leq k \cdot S_m \leq (k \cdot qs_{\mathrm{o}})^{k\cdot\partial} \\
mdg(\underline{t^*}) &\leq vdg_{\mathrm{o}} + k \cdot fid_{\mathrm{o}} \\
mar(\underline{t^*}) &\leq var_{\mathrm{o}} + k \cdot qs_{\mathrm{o}} \cdot fid_{\mathrm{o}} \\
\mathtt{EIL}^{\omega}_{\mathtt{M,v}} &\vdash_{k\cdot(qs_{\mathrm{o}}+\partial)} \exists \underline{x}\, [\, \underline{t^*}\ maj\ \underline{x} \wedge \forall \underline{a}, \underline{y}\, (A^{\mathtt{N}})_{\mathtt{D}}(\underline{x}(\underline{a}), \underline{y}, \underline{a})\,]
\end{aligned}$$

For $A \equiv \forall \underline{x}\, \exists \underline{y}\, A_0(\underline{x}, \underline{y})$ with A_0 quantifier-free and $\{\underline{x}, \underline{y}\} = \mathcal{V}_{\mathrm{f}}(A_0)$, (A.43) can be replaced with

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

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 $(A.43, A.43)$ 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 A.4 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 (A.43) 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 [64] for more details. This feature of monotone functional interpretation is of crucial importance in applications to numerical analysis [81] where $\{z \mid z \leq_1 s\}$ is used to represent compact Polish spaces. Since $A_0(x, z, y)$ is monotone (i.e., $A_0(x, z, y_1) \wedge y_2 \geq y_1 \to A_0(x, z, y_2)$) 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 [68] for more on this).

A.4 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 [45, 69, 123] (even in finite types) and fragments thereof [22, 69, 103].

A.4.1 Treatment of Primitive Recursive Arithmetic $\mathtt{PRA}^\omega$

Let us first consider Feferman's system [28] $\mathtt{PRA}^\omega$ (and its intuitionistic variant $\mathtt{PRA}^\omega_{\mathtt{i}}$) of primitive recursive arithmetic in all finite types, where only quantifier-free induction and ordinary Kleene-primitive recursive functionals are included.

Definition A.75 Let $\mathtt{PRA}^\omega_{\mathtt{i}}$ be an extension of $\mathtt{EIL}^\omega$ with the following:

- Kleene recursor[41] constants $\widehat{R}_\rho$ with axioms

[41] For all ρ the recursor $\widehat{R}_\rho$ can be defined from $\widehat{R}_o$ using λ-abstraction and hence the $\mathtt{EIL}^\omega$

$$\mathtt{Ax}\widehat{R}: \quad \begin{cases} \widehat{R}_\rho(0,y,z,\underline{v}) &=_o z(\underline{v}) \\ \widehat{R}_\rho(Sx,y,z,\underline{v}) &=_o y(\widehat{R}_\rho(x,y,z,\underline{v}),x,\underline{v}) \end{cases} .$$

- Axioms (the usual primitive recursive $+$, $*$, $\overline{sg}$, $|\cdot|$ are defined by $\widehat{R}_o$)

$$\left.\begin{array}{l} x =_o y \leftrightarrow |x-y| =_o 0 \\ x =_o 0 \wedge y =_o 0 \leftrightarrow x+y =_o 0 \\ x =_o 0 \vee y =_o 0 \leftrightarrow x*y =_o 0 \\ (x =_o 0 \to y =_o 0) \leftrightarrow \overline{sg}(x)*y =_o 0 \\ x \neq_o 0 \leftrightarrow \overline{sg}(x) =_o 0 \end{array}\right\} \tag{A.43}$$

- An axiom of quantifier-free-induction (below "$y < x$" is the usual primitive recursively definable strict order relation on natural numbers)

$$\mathtt{IA}'_0: \quad \forall f^1, x^o (f(0) =_o 0 \wedge \forall y < x (f(y) =_o 0 \to f(Sy) =_o 0) \to f(x) =_o 0) .$$

The $\mathtt{EIL}^\omega$-constants ν, $\mathtt{I}$ and $\mathtt{E}$ are immediately definable in $\mathtt{PRA}^\omega_\mathtt{i}$ from (A.43). Also the (here primitive-recursive) *closed* terms associated to *quantifier-free*-formulas A_0 (which here may also contain $\vee$) like in Proposition A.36 are immediately provided in $\mathtt{PRA}^\omega_\mathtt{i}$ with $\mathtt{PRA}^\omega_\mathtt{i} \vdash t_{A_0}(\underline{a}) =_o 0 \leftrightarrow A_0(\underline{a})$. Because of this, $\mathtt{IA}'_0$ implies the following schema of quantifier-free-induction (below A_0 are quantifier-free-formulas which here may contain $\vee$):

$$\mathtt{IA}_0: \quad A_0(0) \wedge \forall x (A_0(x) \to A_0(Sx)) \to \forall x A_0(x) .$$

The $\mathtt{EIL}^\omega$-axiom $\mathtt{TND}_0: \ x =_o 0 \vee x \neq_o 0$ can be immediately derived from $\mathtt{IA}_0$. The $\mathtt{EIL}^\omega$-constant $\mathcal{D}$ can now easily be defined from $\widehat{R}$ in $\mathtt{PRA}^\omega_\mathtt{i}$.

The axioms $\mathtt{Ax}\widehat{R}$ and (A.43) are realizer-free except for the implication

$$x \cdot y =_o 0 \ \to \ (\, x =_o 0 \ \vee \ y =_o 0 \,)$$

whose functional interpretation is realized by $\Pi = \lambda x, y.\, x$. It follows that functional interpretation is immediately available for $\mathtt{PRA}^\omega_\mathtt{i}$ once realizing terms are provided for $\mathtt{IA}'_0$. Such terms of constant size can be built using Kleene recursors $\widehat{R}$ and are equivalent to

$$\lambda f, x.\, min_{\ y<x} \ t_{[\,f(0)=0 \wedge (\,f(y)=0 \to f(Sy)=0\,) \to f(x)=0\,]}(f,x,y) .$$

Since $\geq$ and $\mathcal{M}$ are $\mathtt{PRA}^\omega_\mathtt{i}$-definable as well, the axioms added to $\mathtt{EIL}^\omega$ in Definition A.65 become derivable in $\mathtt{PRA}^\omega_\mathtt{i}$. It follows that also monotone

combinators in view of Definition A.12. This property no longer holds for Gödel recursor R_ρ to be introduced in Section A.4.3. See also Footnote 10 of [4].

functional interpretation is available for $\mathtt{PRA}^{\omega}_{\mathtt{i}}$. In this case $\mathtt{IA}'_0$ is much simpler realized by projectors $\Pi = \lambda f, x.\, x$ (no recursors are needed).

Theorem A.76 All the quantitative results proved above in Theorems A.40, A.72 and Theorems A.62, A.74 carry on to $\mathtt{PRA}^{\omega}_{\mathtt{i}}$, respectively $\mathtt{PRA}^{\omega}$ in the obvious way.

A.4.2 Extension to the analytical system $\mathtt{PRA}^{\omega}+\mathtt{AC}_0+\mathtt{WKL}$.

The analogue of Theorem A.76 for the classical system $\mathtt{PRA}^{\omega}+\mathtt{AC}_0$ holds as well. The system $\mathtt{PRA}^{\omega}+\mathtt{AC}_0$ allows to derive the schemas of Σ^0_1-induction and Δ^0_1-comprehension (see [64]) and therefore contains the system $\mathtt{RCA}_0$ known from reverse mathematics (see [118]).

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 [64] 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 [35, 118]) is Π^0_2-conservative over $\mathtt{PRA}^{\omega}_{\mathtt{i}}$. Moreover, a witnessing term can be provided. We give below a quantitative version of this result. We follow closely the proof in Section 7 of [4] which is a simplification of the more general method of [64].

Let $\mathtt{PRA}^{\omega}$ be formulated over $\mathtt{ECL}^{\omega}_{\mathtt{M}}$. We use the following convenient formulations of the binary König's lemma:

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

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

where (see Section 7 of [4] 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 $\mathit{Bin}(b)$;

[42] A comprehensive discussion of the vast mathematical applicability of $\mathtt{WKL}$ is in[118].

[43] Theorem I.10.3 of [118] gives a summary of important mathematical statements which are theorems of $\mathtt{WKL}_0$. We only mention here the Heine-Borel covering lemma, the maximum principle, the separable Hahn-Banach theorem and Brouwer's fixed point theorem.

- *Bnd* is a primitive recursive predicate which expresses that the given binary tree $BTr(f)$ has depth at most k.

Remark A.77 Below A_0 is quantifier-free, $\underline{x}, \underline{y}$ are type o and $\{\underline{x}, \underline{y}\} = \mathcal{V}_{\mathrm{f}}(A_0)$.

The following theorem expresses the fact that the WKL–elimination and term extraction procedure from WKL–based proofs as developed in [64] 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 [47] and [2] for fragments of second-order arithmetic, the techniques employed there do not provide any term extraction procedure.

Theorem A.78 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

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

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

$$\mathtt{PRA}^{\omega}_{\mathtt{i}} \quad \vdash_{k\cdot(ls_{\mathrm{o}}+\partial)} \quad \forall \underline{x}\, A_0(\underline{x}, \underline{t}(\underline{x}))\ .$$

The time overhead of the algorithm is upper bounded by $k \cdot qs_{\mathrm{o}} \cdot ls_{\mathrm{o}} \cdot S_m(\mathcal{P})$.

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 statements on $\mathcal{P}^{\mathtt{N}}$ in Proposition A.58 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 [4].

The second step is to transform $\mathcal{P}^{\mathtt{N}}$ to the proof in (A.44) via a technique based on functional interpretation and majorization. This technique is described in Lemmas 7.4.1 and 7.4.2 of [4] and is an adaptation of the more general technique of [64]. 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 [4] where a realizer for $\exists y$ is provided as well. □

Corollary A.79 (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

$$\mathcal{P}' : \ \mathtt{PRA}^\omega_{\mathtt{i}} \ \vdash_{k \cdot (ls_{\mathrm{o}} + \partial)} \ \forall \underline{x} \, \exists \underline{y} \, A_0(\underline{x}, \underline{y}) \ .$$

Remark A.80 We could alternatively use a monotone functional interpretation version of Lemma 7.4.1 from [4] in the lines of our Theorem A.74 . 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 A.78 would now hold with (A.44) replaced by

$$\mathtt{PRA}^\omega_{\mathtt{i}} \vdash_{k \cdot (ld(A_0) + qs_{\mathrm{o}} + \partial)} \forall \underline{x} \, A_0(\underline{x}, \underline{t}(\underline{x})) \ .$$

In many cases A_0 is monotone in $\underline{y}$ and therefore bounded search is actually not needed – see also the remarks following Theorem A.74 . 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 $k \cdot fid_{\mathrm{o}} \cdot qs_{\mathrm{o}} \cdot S_m(\mathcal{P})$ and

$$\mathtt{PRA}^\omega_{\mathtt{i}} \quad \vdash_{k \cdot (qs_{\mathrm{o}} + \partial)} \quad \forall \underline{x} \, A_0(\underline{x}, \underline{t}(\underline{x}))$$

hence a full better performance than the algorithm of Theorem A.78 .

Remark A.81 There are three ways to produce a variant of Theorem A.78 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 [90] . 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 A.78 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 (A.44), where

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

[44] Under this type restriction we can allow the use of (full) extensionality axiom EA, see also Remark A.6 . Hence in this setting we work with the fully extensional $\mathtt{PRA}^\omega$ which features the deduction theorem.

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 preprocessing 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 [64], p. 1246 for the meaning of $\oplus$ in this context). For this system the deduction theorem holds w.r.t. $\oplus$ and we obtain (A.44) 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 also Remark A.45.

Remark A.82 Even though the term extraction procedure of Theorem A.78 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.

A.4.3 The case of Peano Arithmetic $\mathtt{PA}^\omega$ and $\mathtt{PA}^\omega + \mathtt{AC}_0 + \mathtt{WKL}$

Already Gödel showed that the functional interpretation of full induction can be realized by his impredicative recursors R for (simultaneous[45]) primitive recursion in finite types, where (below $i \in \overline{1, |\underline{\sigma}|}$ with $|\underline{\sigma}|$ the length of $\underline{\sigma}$)

$$\mathtt{Ax}R: \quad \begin{cases} R^i_{\underline{\sigma}}(\underline{x}, \underline{y}, 0) &=_{\sigma_i} x_i \\ R^i_{\underline{\sigma}}(\underline{x}, \underline{y}, Sz) &=_{\sigma_i} y_i(R^1_{\underline{\sigma}}(\underline{x}, \underline{y}, z), \ldots, R^{|\underline{\sigma}|}_{\underline{\sigma}}(\underline{x}, \underline{y}, z), z) \end{cases} .$$

Gödel's objective was to reduce the consistency of Peano arithmetic PA to that of a quantifier-free calculus (called $\mathbf{T}$) based on these R. In order to achieve this he had to give a verifying proof for the functional interpretation of induction which used only quantifier-free induction. For the applied purpose of program extraction this is not required. We may use full induction in the verifying proof as well. This simplifies matters substantially as was already observed in [123](3.5.5.(iii)). Things are particularly simple if induction is formulated as a rule

$$\mathtt{IR}: \quad \frac{A(0)\,,\ A(z) \to A(Sz)}{A(z)}$$

which nevertheless allows to derive the axiom schema of induction

$$\mathtt{IA}: \quad A(0) \wedge \forall z\,(\,A(z) \to A(Sz)\,) \ \to\ \forall z A(z)\ .$$

[45] Simultaneous primitive recursion even in higher types can be reduced to ordinary primitive recursion in higher types. This is particularly simple in the presence of $\mathtt{ER}_0$, see [123].

The treatment of IR under functional interpretation is fairly similar to that of modus ponens MP. Recursors R have to be used in addition to Σ. Given that $\underline{t_1}$ Dr $A(z,\underline{a})$ and $(\underline{t_2},\underline{t_3})$ Dr $(A(z,\underline{a}) \to A(Sz,\underline{a}))$ one can prove[46] in a constant number of steps that $\underline{t_4}$ Dr $\forall z A(z,\underline{a})$. Here $t_4 \equiv \Sigma\, R\, \underline{t_1}\, (P\, t_3^1) \ldots (P\, t_3^{|t_3|})$. The monotone functional interpretation of IR is much the same as the usual one if we use recursors R^* which can easily be defined from R by a minor modification (see also [58] for similar R^+ recursors). Negative translation applies to IR just as it did for MP. Let Peano arithmetic in all finite types $\mathtt{PA}^\omega$ be formulated over $\mathtt{ECL}^\omega$ plus IR and recursors R. The remarks above imply that we can state the following consequence of (Corollary A.64) of Theorem A.62.

Theorem A.83 There exists $k \in \mathbb{N}$ constant and a functional-interpretation-based algorithm which does the following. Given as input a proof

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

it produces a realizing tuple $\underline{t}$ such that

$$\mathtt{PA}_\mathtt{i}^\omega \vdash_{k\cdot(ls_\mathrm{o}+\partial)} \forall \underline{x}\, A_0(\underline{x},\underline{t}(\underline{x}))$$

and $Sz(\underline{t}) \leq k \cdot S_c(\mathcal{P})$. The time overhead of the algorithm is upper bounded by $k \cdot qs_\mathrm{o} \cdot ls_\mathrm{o} \cdot S_m(\mathcal{P})$.

In contrast to IR, the treatment of IA under usual functional interpretation results in complexity issues similar to those of CT$\wedge$ in Proposition A.37. This is hinted by the fact that the derivation of IA from IR apparently needs contraction $A \to A \wedge A$. Let A be the induction formula; in order to realize IA we need bounded search along the predicate $t_{A_\mathtt{D}}(\underline{x},y) =_o 0$. Here $t_{A_\mathtt{D}}$ is a characteristic term for $A_\mathtt{D}$, see [103]. The monotone functional interpretation of IA avoids this altogether (like before in the case of $A \to A \wedge A$). Now only a majorizing term for $\mu y \leq z\, [\, t_{A_\mathtt{D}}(\underline{x},y) =_o 0\,]$ needs to be constructed and we can simply use $t^* :\equiv \Pi = \lambda \underline{x}, z.\, z$. Let $\mathtt{PA}^\omega$ be formulated over $\mathtt{ECL}_\mathtt{M}^\omega$ plus IA and the recursors R. The remarks above imply that we can state the following result.

Theorem A.84 All the quantitative results of Theorem A.78 and the subsequent considerations carry on to the corresponding $\mathtt{PA}^\omega$-based systems in the obvious way.

[46] Obviously using IR.

Association of Definitions, Notations or (Sub)Sections to Symbols

Name	Defined in	Name	Defined in	Name	Defined in
$\vdash_n$	Section A.0.2	$\{\!\{\cdot,\cdot\}\!\}$	Definition A.20	$\lvert\cdot\rvert$	Section A.0.2
$\widetilde{\cdot}$	Definition A.25	$\geq_\sigma$	Definition A.65	$=_\sigma, \neq$	Section A.1.2
$ar(\cdot)$	Definition A.1 + Section A.1.2	$car(\cdot)$	Section A.1.2	$car_1(\cdot)$	Notation 3
$cdg(\cdot)$	Section A.1.2	$cdg_1(\cdot)$	Notation 3	$\mathcal{C}(\cdot)$	Section A.1.2
$d(\cdot)$	Section A.1.2	$d_{\mathcal{S}}(\cdot)$	Section A.1.2 $\mathcal{S}$ meta-var.	$dg(\cdot)$	Definition A.1 + Section A.1.2
$\mathcal{D}$	Section A.1.3	$\cdot^{\mathtt{D}}, \cdot_{\mathtt{D}}$	Definition A.16	$\cdot$ Dr $\cdot$	Definition A.20
$\partial(\cdot)$	Section A.0.2	$\partial_L(\cdot)$	Section A.0.2 L meta-var.	$\mathtt{E}$	Section A.1.3
$fd(\cdot)$	Section A.1.2	$fd_1(\cdot)$	Notation 2	$fid(\cdot)$	Section A.1.2
$id(\cdot)$	Section A.1.2	$id_0(\cdot)$	Notation 2	$\mathtt{I}$	Section A.1.3
k_0	Section A.0.2	$\lambda x.\, t$	Definition A.12	$L(\cdot)$	Section A.0.2
$ld(\cdot)$	Section A.1.2	$ld_1(\cdot)$	Notation 2	$ls(\cdot)$	Section A.1.2
$ls_0(\cdot)$	Notation 2	$ls_1(\cdot)$	Notation 2	$\mathrm{Lv}(\cdot)$	Section A.0.2
$mdg(\cdot)$	Section A.1.2	$mar(\cdot)$	Section A.1.2	$\cdot\, maj_\sigma\, \cdot$	Definition A.65
$\mathcal{M}_\sigma\ \cdot\ \cdot$	Definition A.65	$\cdot^{\mathtt{N}}$	Definition A.55	$\mathtt{0}_\rho$	Section A.1.3
Σ, Π	Definition A.4 + Section A.1.3	P	Section A.1.3	$PR[\cdot]$	Definition A.20
$qs(\cdot)$	Section A.1.2	$qs_0(\cdot)$	Notation 2	$RR[\cdot]$	Definition A.20
$RTS[\cdot]$	Definition A.20	$S(\cdot)$	Section A.1.2	$\mathtt{S}$	Section A.1.3
$S_i(\cdot)$	Definition A.46	$S_c(\cdot)$	Definition A.46	$S_m(\cdot)$	Definition A.46
$Sz(\cdot)$	Theorem A.51	$Sz'(\cdot)$	Definition A.47	$\not{typ}(\cdot)$	Section A.1.2
$td(\cdot)$	Section A.1.2	$ts(\cdot)$	Section A.1.2	ν	Section A.1.3
$var(\cdot)$	Section A.1.2	$var_0(\cdot)$	Notation 2	$vdg(\cdot)$	Section A.1.2
$vdg_0(\cdot)$	Notation 2	$\mathcal{V}(\cdot)$	Section A.1.2	$\mathcal{V}_{\mathrm{b}}(\cdot)$	Section A.1.2
$\mathcal{V}_{\mathrm{f}}(\cdot)$	Section A.1.2	$\mathrm{Vt}(\cdot)$	Section A.0.2	$wd(\cdot)$	Section A.1.2
$wd_1(\cdot)$	Notation 3	$ws(\cdot)$	Section A.1.2	$ws_1(\cdot)$	Notation 3

Distribution of Definitions and Notations in (Sub)Sections

Def. Not.	Section	Def. Not.	Section	Def. Not.	Section	Def. Not.	Section
A.1	A.1	A.4	A.1.3	A.12	A.1.3	A.16	A.2
A.20	A.2	A.25	A.2.1	2	A.2.3	3	A.2.3
A.46	A.2.4	A.47	A.2.4	A.51	A.2.4	A.65	A.3.2