CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Digitale Hochschulschriften der LMU

Fakultät für Mathematik, Informatik und Statistik
Ludwig–Maximilians–Universität
München

Mathematische Logik
Prof. Dr. Helmut Schwichtenberg

Refinement of Classical Proofs for Program Extraction

Diana Ratiu

Refinement of Classical Proofs for Program Extraction

Diana Ratiu

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
Ludwig–Maximilians–Universität
München

vorgelegt von
Diana Ratiu
April 2011

Diana Ratiu

Refinement of Classical Proofs for Program Extraction

Dissertation an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

1. Berichterstatter: Prof. Dr. Helmut Schwichtenberg
2. Berichterstatter: Prof. Dr. Wilfried Buchholz
1. Prüfer: Prof. Dr. Otto Forster
2. Prüfer: Prof. Dr. Günther Kraus
Ersatzprüfer: Prof. Dr. Martin Schottenloher
Externer Gutachter: Prof. Dr. Hajime Ishihara

Datum der Einreichung: 4. April 2011
Tag der mündlichen Prüfung: 16. August 2011

To my family

Acknowledgments

"Should every one of those who have given
of themselves to us take their share back,
we could not possibly know what would be left of us;
therefore, we owe them all our gratitude." (Teofil Pãrãian)

My gratitude goes to Prof. Helmut Schwichtenberg who has given me the opportunity to be a PhD Student in the Mathlogaps Project. I have learned from him a lot, in terms of Logics and in terms of research: his rigor has taught me how proper mathematics is done. I am grateful that he has encouraged me to visit other research groups and gave me the unique opportunity to meet leading researchers in Munich.

I am honored to have Prof. Hajime Ishihara as a second reviewer of my thesis. I would like to thank him in particular for his patience in clarifying my questions during our long discussions both in Japan and in Munich.

During my PhD study, I benefited from the high-quality lectures of Prof. Wilfried Buchholz. His talks in our group meetings were mind-provoking and his precision motivating. I am thankful to him for accepting to be my internal reviewer.

The Mathlogaps project has given me the chance to visit various research groups and get valuable advice and a critical view of my work. Among those which have contributed, I am grateful in particular to Ulrich Berger and Monika Seisenberger for the illuminating discussions during my visit in Swansea, to Roy Dyckhoff for inviting me to visit his research group in St. Andrews and raising issues that shed a new light on my research, to Paulo Oliva for sharing his perspective regarding proof interpretation; to Bruno Buchberger for challenging me with questions on the importance of classical proofs, during his Gröbner Bases Semester in Linz.

In Munich I had the opportunity to be part of a competitive and open environment: I have exchanged ideas and worked together with many people. First of all, I would like to thank Stefan Schimanski for taking the time to

read my blackboard notes on the intriguing parts of my work. On many occasions he cleverly asked me questions; his insight has helped me clarify many aspects regarding my research.

The underground seminar has been a valuable experience and I am grateful to my colleagues for their wonderful talks and challenging questions: Basil Karadice, Freiric Baral, Luca Chiarabini, Dominik Schlenker and Stefan Schimanski. Klaus Thiel was motivational with his perseverance in tackling new theorems in Minlog.

I am also thankful to those who have their share in some of the chapters of this thesis. Our long sessions with Trifon Trifonov on the example of the Infinite Pigeonhole Principle has clarified many of the otherwise cumbersome aspects of the associated program. The expertise of Josef Berger on constructive mathematics has been helpful in investigating the strength of Dickson's Lemma and has opened the way to future research. I spend long afternoons debating on the benefits and limitations of the double-negation translation with Christian Urban.

Special thanks to Daniel Ratiu, Stefan Schimanski, Bogomil Kovachev, Kenji Miyamoto, Roy McCasland and Martin Ochoa for putting aside their own research to review my thesis and provide me with valuable feedback.

I grew to become what I am today professionally thanks to my mathematics teachers Saveta Coste and Ruxanda Georgescu. Later, my BSc Thesis supervisors Prof. Viorel Negru and Prof. Tudor Jebelean have given me the first impulse towards research. My Master's Thesis supervisor Prof. Dana Petcu has been a role model, guiding me through my first research years and inspiring me with her energy, motivation and willingness to explore new ideas.

Last but not least, I dedicate this work to our little one, Stefan, to my husband Daniel and my parents. I thank in particular my husband for his support and for being an inspiration with his hard-work and perseverance. I am most grateful to my parents - their love and care for me have guided me throughout all my life. I owe them what I am today.

Abstract

The A-Translation enables us to unravel the computational information in classical proofs, by first transforming them into constructive ones, however at the cost of introducing redundancies in the extracted code. This is due to the fact that all negations inserted during translation are replaced by the computationally relevant form of the goal.

In this thesis we are concerned with eliminating such redundancies, in order to obtain better extracted programs. For this, we propose two methods: a controlled and minimal insertion of negations, such that a refinement of the A-Translation can be used and an algorithmic decoration of the proofs, in order to mark the computationally irrelevant components.

By restricting the logic to be minimal, the Double Negation Translation is no longer necessary. On this fragment of minimal logic we apply the refined A-Translation, as proposed in (Berger et al., 2002). This method identifies further selected classes of formulas for which the negations do not need to be substituted by computationally relevant formulas. However, the refinement imposes restrictions which considerably narrow the applicability domain of the A-Translation. We address this issue by proposing a controlled insertion of double negations, with the benefit that some intuitionistically valid Π^0_2-formulas become provable in minimal logic and that certain formulas are transformed to match the requirements of the refined A-Translation.

We present the outcome of applying the refined A-translation to a series of examples. Their purpose is two folded. On one hand, they serve as case studies for the role played by negations, by shedding a light on the restrictions imposed by the translation method. On the other hand, the extracted programs are characterized by a specific behaviour: they adhere to the continuation passing style and the recursion is in general in tail form.

The second improvement concerns the detection of the computationally irrelevant subformulas, such that no terms are extracted from them. In order to achieve this, we assign decorations to the implication and universal quantifier. The algorithm that we propose is shown to be optimal, correct and terminating and is applied on the examples of factorial and list reversal.

Kurzfassung

Die A-Übersetzung ermöglicht es, die rechnerische Information aus klassischen Beweisen einzuholen. Dennoch hat sie den Nachteil, dass die Programme, die man aus auf diese Weise transformierten Beweisen extrahiert, viele redundante Teile enthalten. Das liegt daran, dass die A-Übersetzung viele doppelte Negationen hinzufügt und alle diese Negationen durch die rechnerisch relevante Form der Ziel-Formel substituiert werden.

In dieser Doktorarbeit werden Methoden dargestellt, um Teile der redundante Information in den extrahierten Programen zu entfernen. Einerseits wird das Einfügen der Negationen minimal gehalten und anderseits werden die nicht rechnerischen Teile als solche indentifiziert und ausgezeichnet.

Wir bemerken zuerst, dass in der Minimallogik das Einfügen der doppelten Negationen nicht mehr nötig ist. Darüber hinaus, um das Ersetzen aller Negationen zu vermeiden, identifizieren Berger et al. (2002) diejenigen, wo die Substitution nicht nötig ist. Diese verfeinerte A-Übersetzung hat aber den Nachteil, dass sie den Anwendungsbereich begrenzt. Um das zu beseitigen, wird in dieser Dissertation eine verfeinerte Doppel-Negation angewandt, die bestimmte Formeln so umsetzt, dass die verfeinerte A-Übersetzung darauf anwendbar ist. Als Zugabe kann diese Methode auch benutzt werden, um konstruktive Beweise mancher Π^0_2-Formeln in der Minimallogik durchzuführen.

Dieses Verfahren wird durch Anwendung der verfeinerten A-Übersetzung auf eine Reihe von bedeutenden Fallstudien illustriert. Es werden das Lemma von Dickson, das unendliche Schubfachprinzip und das Erdös-Szekeres Theorem betrachtet. Dabei wird es festgestellt, dass ein Zusammenhang zu der Endrekursion und dem Rechnen mit Fortsezungen besteht.

Ferner, um möglichst viel der überflüssigen Information zu entfernen, wird ein Dekorationsalgorithmus vorgelegt. Dadurch werden die rechnerisch irrelevanten Komponenten identifiziert und entsprechend annotiert, so dass sie während der Extraktion nicht berücksichtigt werden. Es wird gezeigt, dass das vorgeschlagene Dekorationsverfahren, das auf Beweisebene eingesetzt wird, optimal, korrekt und terminierend ist.

Contents

1 Introduction **1**
- 1.1 The Problem and Proposed Solutions . . . 2
- 1.2 Contributions . . . 4
- 1.3 Road-map of the Thesis . . . 5

2 NA^ω as the Negative Fragment of HA^ω **9**
- 2.1 Language Specification . . . 10
- 2.2 Weak and Strong Operators . . . 14
- 2.3 Axioms . . . 15
  - 2.3.1 Induction and Cases . . . 15
  - 2.3.2 efq and Stab . . . 16
  - 2.3.3 Other Axioms . . . 18
- 2.4 Natural Deduction . . . 18
- 2.5 Summary of the Chapter . . . 21

3 A-Translation and its Refined Version **23**
- 3.1 The Double Negation Translation . . . 24
- 3.2 A-Translation . . . 25
  - 3.2.1 Friedman's "trick" . . . 25
  - 3.2.2 A-Translation . . . 26
- 3.3 The Refinement of the A-Translation . . . 29
  - 3.3.1 Definite and Goal Formulas . . . 31
  - 3.3.2 Properties of the Definite/Goal Formulas . . . 34
  - 3.3.3 Refined A-Translation . . . 38
- 3.4 Program Extraction . . . 39
  - 3.4.1 Modified Realizability. Extracted Terms . . . 39
  - 3.4.2 Program Extraction from A-Translated Proofs . . . 42
- 3.5 Related Work . . . 44
- 3.6 Summary of the Chapter . . . 52

4 A-Translation and Programming **55**
4.1 Tail Recursion . . . 56
4.1.1 Tail Recursion in Computer Science . . . 56
4.1.2 Connection with the Refined A-Translation . . . 57
4.2 Computing with Continuations . . . 62
4.2.1 Classical Proofs and the Notion of Control . . . 63
4.2.2 Connection with the Refined A-Translation . . . 64
4.3 Summary of the Chapter . . . 66

5 The Infinite Pigeonhole Principle **69**
5.1 The Infinite Boolean Tape . . . 69
5.1.1 Statement and Formalization . . . 70
5.1.2 The Logical Falsity $\bot$ . . . 74
5.1.3 The Extracted Program . . . 75
5.1.4 Related Work . . . 80
5.2 Generalization: IPH . . . 82
5.2.1 IPH - Statement and Proof . . . 82
5.2.2 Double Negations . . . 85
5.2.3 The Π^0_2-Corollary . . . 85
5.2.4 Results with A-Translation . . . 87
5.2.5 Related work . . . 94
5.3 Summary of the Chapter . . . 94

6 The Erdös-Szekeres Theorem **97**
6.1 The Finite Pigeonhole Principle . . . 98
6.1.1 Formalization . . . 99
6.1.2 The Program Extracted from FPH . . . 102
6.2 The Erdös-Szekeres theorem . . . 107
6.2.1 Formalization Issues . . . 109
6.2.2 The Double-negated FPH . . . 114
6.2.3 Formalization of the Erdös-Szekeres Theorem . . . 118
6.2.4 Extracted Program . . . 120
6.3 Summary of the Chapter . . . 123

7 Dickson's Lemma **125**
7.1 Terminology . . . 126
7.2 The Minimum Principle . . . 128
7.3 Equivalent Formulations . . . 129
7.4 A Π^0_2-Corollary of Dickson's Lemma . . . 135
7.4.1 A Simplified Statement. . . . 135

7.4.2 Generalization 142
7.5 Applications of Dickson's Lemma 145
7.5.1 Further Terminology 145
7.5.2 Dickson's Lemma in the Gröbner Bases Theory 147
7.6 Summary of the Chapter 149

8 Refined Double-Negation 151
8.1 Negations 152
8.2 Critical Predicate Symbols 154
8.3 Refined Double Negation 157
8.3.1 Obtaining Definite/Goal Formulas 157
8.3.2 Refined Double Negation of Proofs 160
8.4 Examples 162
8.4.1 The "Least" Element in a Well-Founded Set 162
8.4.2 The Infinite Pigeonhole Principle (Chapter 5) 164
8.4.3 Dickson's Lemma (Chapter 7) 166
8.4.4 The Erdös-Szekeres Theorem (Chapter 6) 167
8.5 Further Refinement of the Double Negation 175
8.6 Summary of the Chapter 178

9 Decoration of Proofs 181
9.1 Terminlogy 182
9.2 Decoration of Axioms 186
9.3 Decoration Algorithm 189
9.4 Examples 200
9.4.1 Decoration of Implication 200
9.4.2 Program Transformation by Decorations 200
9.4.3 List reversal 206
9.5 Summary of the Chapter 211

10 Concluding Remarks 213
10.1 Overview of this Thesis 213
10.2 Contributions 216
10.3 Future Lines of Research 217

A Technical Details 221
A.1 The Erdös-Szekeres Theorem 221
A.2 Dickson's Lemma 226

Chapter 1

Introduction

In this thesis we are concerned with the interpretation of non-constructive existence proofs, such that their computational content is extracted into computable functionals. The reason why such proofs are of interest is that in traditional mathematical textbooks no attention is payed to the distinction between classical and constructive principles and many of the proofs are indirect, i.e., they show existential statements by contradiction. However, the distinction is essential in regards to the interpretation given to the existential quantifier. Whereas intuitionistic logic requires a witness to be provided in order to validate existential statements, in classical logic the existence is seen merely as an abbreviation, in the sense: $\tilde{\exists}_x A \equiv (\forall_x A \to \bot) \to \bot$.

As a consequence, when associating programs to existence proofs, the classical components have the drawback that their computational content is most often hidden. For this reason, efforts have been invested into uncovering the computational meaning of classical[1] proofs or, in other words, into extracting the witness for y in $\vec{G}$, when the goal formula is $\forall_x \exists_y \vec{G}(x, y)$.

We read the Π^0_2-formulas $\forall_x \exists_y G(x, y)$ as:
"Assume that an input x and a property G which x should fulfill are given. Is there an algorithm which produces an output y from this input, such that the requirement G on x is met?".

It is known that the Π^0_2- formulas with arithmetical and decidable kernel are equiderivable in classical and intuitionistic logic[2], i.e.

$$\Gamma \vdash_c \forall_x \exists_y G(x, y) \text{ iff } \Gamma' \vdash_i \forall_x \exists_y G(x, y) \tag{1.1}$$

[1] By abuse of the language, we refer to *classical* proofs when in a strict sense we mean the fragment of constructive logic in which $\exists$ is not explicit.

[2] $\vdash_c$ denotes derivability in classical and $\vdash_i$ in intuitionistic logic.

This suggests that given proofs by contradiction of $\forall_x \exists_y G(x, y)$ it should be possible to obtain their constructive counterpart and thus unravel the computational content. In order to achieve this, we need to recover the constructive semantics of $\exists$ and identify the witness provided in the proof. For this we need to either assign proper interpretations to the classical principles or give some intuitionistically valid versions by adequate translations.

In the case of intuitionistic proofs the semantics associated to the logical operators allows for their interpretation as programs, via the formulas-as-types correspondence. Transforming classical proofs in their constructive counterparts would also enable us to use this interpretation. By the Curry-Howard isomorphism, it would then be possible to associate computational λ-terms to the inference rules of Natural Deduction.

Beginning as early as the 1930's with the work of Gödel, various methods have been proposed in order to transform classical proofs into constructive ones. Gödel (1933), in parallel with Gentzen (1936), developed a form of double-negation translation, which has further developed into variants. Further, Gödel (1958) introduced the so-called "Dialectica Interpretation". In this thesis, we focus on a transformation method as presented in (Friedman, 1978), which is a hybrid of the Gödel-Gentzen negative translation and Friedman-Dragalin's trick - also known as A-translation. We work with a refinement of this method, as proposed in (Berger et al., 2002) and further improved in (Schwichtenberg, 2007). The role of this refinement is to minimize the occurrences of the logical falsity, with the purpose of brushing up the associated programs of redundant computations.

1.1 The Problem and Proposed Solutions

Our work investigates under which circumstances and with what results we can synthesize programs from classical proofs of Π^0_2-formulas. More precisely, we analyze the computational potential of proofs in the fragment of the intuitionistic logic which coincides with the classical logic. From this we exclude not only $\mathsf{Stab}^{\perp}$, but also $\mathsf{efq}^{\perp}$, thus obtaining the minimal logic system NA^{ω}. The interpretation given to the weak existential quantifier, seen as an abbreviation for $(\forall_x A \to \perp) \to \perp$ delimits the system as "classical".

In order to unravel the computational content in classical proofs we use the A-Translation. However, as the PhD Thesis (Murthy, 1990) has revealed, the introduction of double negations in the process of A-translating, not only makes the proofs expand exponentially, but it consequently introduces unnecessary computations in the extracted programs. By working in

the fragment of minimal logic, the insertion of double negations is no longer necessary. Moreover, (Berger et al., 2002) identifies a class of formulas for which the number of computationally relevant negations can be further minimized. When it has no computational contribution, the logical falsity $\bot$ is substituted by a constant for "false" (F) which is of null type. The remaining $\bot$ are substituted by the strong existence goal, such that the intended semantics of $\exists$ is recovered. This is then coupled with modified realizability, in order to associate computational terms to the translated proof terms.

The classes of formulas for which the refined A-Translation is defined introduce significant constraints. In order to enhance the applicability domain of the method, we propose in this thesis to insert double negations in a controlled and restrictive manner. For this, it suffices to negate only atoms containing "critical predicate symbols" as defined in (Berger et al., 2002).

On top of these constraints, the minimal logic system NA^ω in which we are working can be seen as rather restrictive, since $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$ are not accepted. This makes the unification of the assumptions, among which we have $\forall_x(G \to \bot)$ resulting from unfolding the $\tilde{\exists}$, sometimes impossible. The versions of efq and Stab in which negations are represented as constants are provable; moreover, the double negations of $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$ are NA^ω-provable. Thus, a careful manipulation of negations could enable us to overcome the aforementioned restriction. For this purpose we identify in this thesis the so-called L-critical atoms which we double negate with the purpose of embedding some classically valid formulas into minimal logic. Because no other atoms need to be double negated in the process of A-translation as a result we minimize in the associated programs the redundant information, arising from the use of logical falsity.

We will also show in this thesis that it is possible to further improve the extracted terms, by associating so-called decorations to the logical operators. These can be seen as colorings, marking the operators according to their computational relevance, such that when we apply the modified realizability, the terms associated to the irrelevant formulas are eliminated from the extracted programs. Since in NA^ω we only have the $\forall$-quantifier and the $\to$ connector[3], we will exemplify in this work only the decorations for these operators. Further decorations for the strong $\exists$ and $\vee$ are also possible and the decoration algorithm can be applied directly to constructive proofs.

The theoretical aspects are supported by some relevant examples. These will serve a double-folded purpose: on one hand, we analyze the effects of the refined A-translation in terms of the extracted programs, in view of its

[3] $\tilde{\exists}$ and $\tilde{\vee}$ are seen as abbreviations.

relation to various programming techniques. On the other hand, based on these examples, we identify the issues arising from the use of the method, in the sense of the aforementioned restrictions. The case studies will serve as a starting point for the statements which we make regarding the use of double negation, inserted around special atoms, called L-critical, in order to overcome these restrictions.

1.2 Contributions

The *contributions* of this dissertation evolve around the refined A-Translations and can be divided in three main categories:

- the extention of the field of application of the translation method
- the application of the A-Translation to selected problems
- a decoration algorithm for the marking of quantifiers and connectors, according to their computational relevance.

The third contribution is in fact a general method, independent of the translation procedure used to extract programs from classical proofs. It turns out that the decoration procedure, applicable at the proof level, can be used also directly on constructive proofs.

More precisely, the contributions, summarized also in Figure 1.1, are:

- an analysis of the specifics of the refined A-Translation, with an extension of its field of application:
  - a new perspective on the classes of goal/definite formulas to which A-Translation is applied: we introduce these classes in Chapter 3 and compare them to similar ones from the literature in Section 3.5, in order to gain insight on the role of the imposed restrictions
  - the "refined double negation" for a controlled insertion of negations, in order to transform intuitionistic proofs into minimal logic ones and to obtain definite/goal formulas, as described in Chapter 8
  - observations regarding the correlation between A-Translation and programming techniques (tail recursion and the control operators) are made in Chapter 4
- the analysis of the programs extracted from non-constructive proofs of some Π^0_2-theorems, after transformation by the refined A-Translation. The selected case studies are:

  - the corollary of the Infinite Pigeonhole Principle, both for a simplified and the general case (Chapter 5)
  - a modified version of the Finite Pigeonhole Principle and the Erdös-Szekeres Theorem (Chapter 6)
  - a corollary of Dickson's Lemma; we have investigated also the correlations between various formulations of Dickson's Lemma in view of its applications in the Gröbner Basis Theory (Chapter 7)

  We have formalized these examples in the proof assistant (Minlog), which allows to exploit the proofs-as-programs paradigm for program development. Based on minimal logic, Minlog allows the extraction of programs from classical proofs, by providing a special package which deploys the refined A-Translation method.

- the Decoration Algorithm used for marking operators according to their computational relevance, in order to eliminate redundancies in the extracted programs. In Chapter 9 we:
  - refine the notions of computational types, proof terms, extracted terms and the Natural Deduction rules, in order to incorporate these markings
  - provide the decorated form of the uninstantiated[4] axioms
  - describe the decoration algorithm and prove it to be correct, optimal and terminating
  - apply the decoration method to some test problems

1.3 Road-map of the Thesis

We work in this thesis in a negative fragment of Heyting's arithmetic, restricted to $\rightarrow$, $\neg$ and $\forall$. We present our system NA^ω in Chapter 2 and give the interpretation for the weak operators $\tilde{\exists}$ and $\tilde{\vee}$. Negation plays a special role in our system and can be taken to be either the constant F or the predicate variable $\bot$, as presented in Section 2.2. Consequently, a special treatment concerns efq and Stab, which have two forms, depending on which from of negation is used - Section 2.3.2 provides more detail on this.

As we have already mentioned, we build our interpretation of classical proofs on a refined version of the Friedman-Dragalin (A-)Translation. In

[4]Uninstantiated in the sense that the axioms may contain predicate variables.

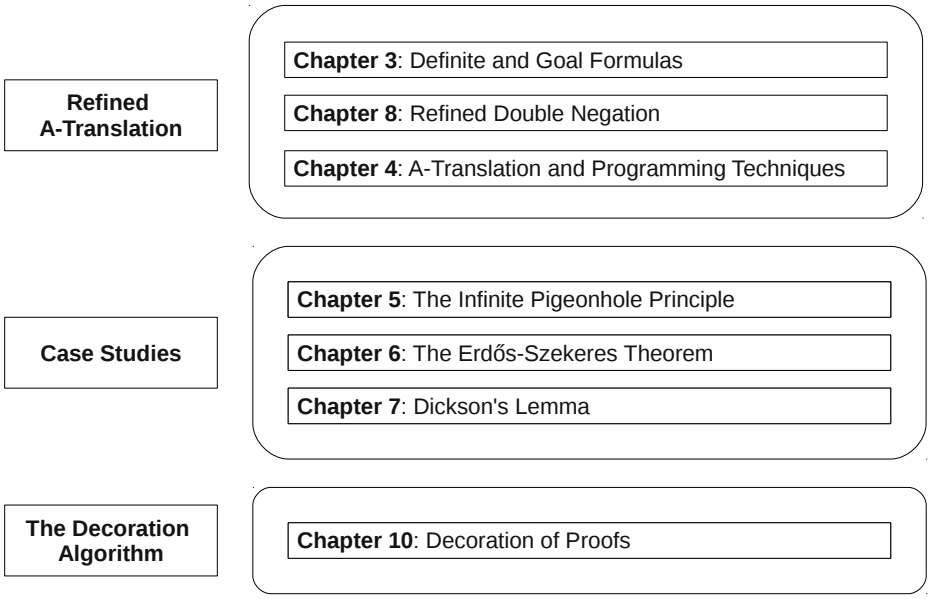

Figure 1.1: Contributions of the dissertation

Chapter 3 we overview the original method and give its refinement due to (Berger et al., 2002). As central notions, we present in Section 3.3 some special classes of formulas, called definite and goal, to which the refined A-Translation can be applied. The machinery which identifies the terms that do not contribute computationally - i.e., where $\bot$ does not occur or can be replaced by its computationally irrelevant variant F, with the purpose of discarding the irrelevant components, is presented in Section 3.3. The next Section 3.4 presents the mechanism by which programs are associated to the A-Translated proofs. In Section 3.5 we make a comparison between the definite/goal classes and a related work of (Ishihara, 2000). The aim was to identify whether an extension of these classes to which the refined A-Translation is applicable is possible - however, with a negative answer. The comparison in Section 3.5 shows the necessity of the restrictions imposed on the definite/goal formulas when proofs are carried out in minimal logic.

Chapter 4 underlines some specific programming techniques associated with the extracted programs, in view of related work. On one hand, we

establish connections with the concept of continuation passing style which has been put by (Griffin, 1990) in correspondence with the computational content of classical proofs. On the other hand, we observe that the tail form of recursion resulting from the application of some transformation techniques to constructive proofs as proposed in (Chiarabini, 2010) is in general implicit in the corresponding A-translated classical proofs.

We present in the following chapters of this thesis the outcome of applying this refined form of A-translation to a series of interesting case studies:

- starting from Stolzenberg's example, we have generalized it in Chapter 5 to the Infinite Pigeonhole Principle (IPH), stating that any infinite sequence that is colored with finitely many colors has an infinite monochromatic subsequence. We analyze the behavior of A-Translation on a corollary of this principle and present the simplified case of a boolean sequence in Section 5.1 and the generalization to r colors in Section 5.2. We carry out in Section 5.2.4 an extensive analysis of the extracted program from the corollary of (IPH).

- the finite version of the Pigeonhole Principle (FPH) is used to prove an interesting result in combinatorics due to Erdös and Szekeres, showing the existence of monotonically increasing/decreasing subsequences of a given sequence. Unlike its infinite counterpart - which is however unrelated computationally - (FPH) is a Π^0_2-formula. Since the original formulation is not appropriate in order to prove the Erdös-Szekeres Theorem in minimal logic, a double-negated form of the assumption needs to be considered. We present a classical proof of (FPH), its double negated variant and the use of the latter in the proof of the Erdös-Szekeres Theorem in Chapter 6. In each case we present the corresponding extracted programs and comment on the specifics arising from using the refined A-Translation.

- Dickson's Lemma, which plays an essential role for Buchberger's Algorithm for computing the Gröbner Basis, asserts that for any (finite) number of infinite sequences of natural numbers, there exists an unbounded set M, such that each of these sequence increases on M.

  The strong version of Dickson's Lemma is not suited for the refined A-translation, as discussed in Chapter 7. In Section 7.3 we present the correlations between the various formulations, in order to understand the strength of the variant which we have analyzed. In Section 7.4 we carry out a thorough analysis of the A-translated Π^0_2-corollary of the lemma for the case of two functions increasing in two distinct points and the general

case of arbitrary functions; we analyze the extracted program from the perspective of the computation by continuations.

Our motivation for analyzing Dickson's Lemma was its use in the Gröbner Bases Theory, so we overview in Section 7.5 its applications in this context. We propose further lines of investigation and suggest to shift the use of the refined A-Translation to these applications. We also sketch a possible relation between Dickson's Lemma and some constructive and classical principles, in order to determine the strength of the classical variant that we have investigated in this thesis.

As a result of analyzing the above listed examples (see also Section 8.4), we have observed that some parts of the proofs require $\mathsf{efq}^{\bot}$. Since the double negated form of $\mathsf{efq}^{\bot}$ is NA^{ω}-provable, in order to carry out the proofs in minimal logic, it is sufficient to modify the formulations of the theorems by introducing some explicit negations. The aim is to keep the double-negations under control, such that the extracted programs do not contain redundant terms arising from substituting each logical falsity by computational formulas during the refined A-Translation. In order to insert only the necessary double negations, we use in Chapter 8, Section 8.2, the concept of L-critical predicate symbols from Berger et al. (2002) and introduce in Section 8.3 the Refined Double Negation method. In Section 8.3.1 we show that by this method one can obtain definite/goal formulas.

In Chapter 9 we go a step further in the brush-up of extracted programs and propose an annotation of the logical operators, by which the non-computational ones are marked accordingly. Starting from the work of (Berger, 2005), which introduces $\forall^{nc}$, we propose to distinguish between relevant/irrelevant implications and call the $.^{c}/.^{nc}$ markings "decorations". The extraction procedure is adapted accordingly and the redundant parts corresponding to the nc-operators are eliminated from the associated programs. Since this procedure is parameterized by the axioms and global assumptions, we discuss the a-priori decoration of the relevant axioms in Section 9.2. We present in Section 9.3 the algorithm introduced in (Ratiu and Schwichtenberg, 2010) with the purpose of decorating the proofs in such a way, that all computationally irrelevant universal quantifiers and implications are marked as such. We illustrate in Section 9.4 the application of the decoration algorithm on some simple, but relevant examples and pointing out the improvements in the extracted programs.

Concluding remarks and an overview of problems of interest for further research are summarized in Chapter 10. Some implementation details and unfoldings for some extracted terms are gathered in Appendixes A.1, A.2.

Chapter 2

NA^ω as the Negative Fragment of HA^ω

We introduce in this chapter the language and the notations used throughout the thesis. The system described in the following is based on Gödel's System T (Gödel, 1958) and constitutes a negative fragment of Heyting's Arithmetic (Heyting, 1966) with finite types (HA^ω) (cf. (Troelstra, 1973)). We restrict the language to $\rightarrow$ and $\forall$ for reasons that should become clear in Section 2.2, in particular in the Paragraph "Weak and Strong Operators" on page 14, and refer the resulting system as the Negative Arithmetic (NA^ω). For decidable formulas this fragment of intuitionistic logic coincides with classical logic in terms of provability, but differs with respect to the semantics associated to some of the logical operators. In particular, the weak existential quantifier is viewed only as an abbreviation for the formula $(\forall_x G \rightarrow \bot) \rightarrow \bot$, as presented in Section 2.2, so it has no explicit constructive meaning.

We begin by defining in Section 2.1 the fundamental notions of *types*, *terms*, *formulas* and *proof terms*. In the process of translating the proofs in order to extract their computational content, negation will play an essential role. Thus, special care is taken in defining negation and the weak operators $\tilde{\exists}$ and $\tilde{\wedge}$. Section 2.3 overviews the axioms on which the theory is built with a special treatment in the case of efq and Stab, since they involve negations. In Section 2.4 we summarize the inference rules of the Natural Deduction system and we conclude the chapter with a brief summary in Section 2.5 justifying the choice for minimal logic.

2.1 Language Specification

We consider the following **base types**, having as domains free algebras, defined in terms of their constructors:

Natural numbers: $\mathtt{N} := \mu\alpha\,(\alpha, \alpha \to \alpha)$, given by $0^{\mathtt{N}}, \mathsf{Succ}^{\mathtt{N}\Rightarrow\mathtt{N}}$

Booleans: $\mathtt{B} := \mu\alpha\,(\alpha, \alpha)$, constructed by $\mathrm{tt}^{\mathtt{B}}, \mathrm{ff}^{\mathtt{B}}$

Lists: $\mathtt{L}(\rho) := \mu\alpha\,(\alpha, \rho \to \alpha \to \alpha)$, parameterized by the simple types of their elements and built by $\mathsf{nil}^{\rho}, \mathsf{cons}^{\rho\Rightarrow\mathtt{L}(\rho)\Rightarrow\mathtt{L}(\rho)}$.

where α and ρ are type variables.

Notation 2.1. *We abbreviate* $\mathsf{Succ}\, n^{\mathtt{N}}$ *by* $\mathsf{S}n$ *and* $\mathsf{cons}\, x^{\rho}\, l^{\mathtt{L}(\rho)}$ *by* $x :: l$.

Definition 2.1 (Types)**.** *Starting from the* base types*, further simple* types *(ρ, σ) are built by the function and product type constructors:*

$$\rho, \sigma := \mathtt{N} \mid \mathtt{B} \mid \mathtt{L}(\rho) \mid \rho \Rightarrow \sigma \mid \rho \times \sigma$$

Notation 2.2. *Terms, as given by the following definition, are considered to be always typed, but whenever clear from the context, the types will be omitted. Otherwise, we write the type α of t either as a superscript (t^{α}) or following the term ($t : \alpha$).*

Definition 2.2 (Terms)**.** Terms *are defined inductively and are formed by abstraction, application, pairing and projection from typed variables (x^{ρ}) and constants (the previously introduced constructors, the recursion and the cases operators):*

$$\begin{aligned} s, t := {} & x^{\rho} \mid (\lambda x^{\rho} t^{\sigma})^{\rho\Rightarrow\sigma} \mid (s^{\rho\Rightarrow\sigma} t^{\rho})^{\sigma} \mid \langle t^{\rho}, s^{\sigma}\rangle^{\rho\times\sigma} \mid (t^{\rho\times\sigma})_0^{\rho} \mid (t^{\rho\times\sigma})_1^{\sigma} \mid \\ & 0^{\mathtt{N}} \mid \mathsf{S}^{\mathtt{N}\to\mathtt{N}} \mid tt^{\mathtt{B}} \mid ff^{\mathtt{B}} \mid \mathsf{nil}^{\mathtt{L}(\rho)} \mid cons^{\rho\Rightarrow\mathtt{L}(\rho)\Rightarrow\mathtt{L}(\rho)} \mid \\ & \mathcal{R}_{\mathtt{B}}^{\sigma} \mid \mathcal{R}_{\mathtt{N}}^{\sigma} \mid \mathcal{R}_{\mathtt{L}(\rho)}^{\sigma} \mid \mathcal{C}_{\mathtt{N}}^{\sigma} \mid \mathcal{C}_{\mathtt{L}(\rho)}^{\sigma} \end{aligned}$$

The full typing for the recursion operators is as follows:

$$\begin{aligned} \mathcal{R}_{\mathtt{B}}^{\sigma} &: \mathtt{B} \Rightarrow \sigma \Rightarrow \sigma \Rightarrow \sigma, \\ \mathcal{R}_{\mathtt{N}}^{\sigma} &: \mathtt{N} \Rightarrow \sigma \Rightarrow (\mathtt{N} \Rightarrow \sigma \Rightarrow \sigma) \Rightarrow \sigma, \\ \mathcal{R}_{\mathtt{L}(\rho)}^{\sigma} &: \mathtt{L}(\rho) \Rightarrow \sigma \Rightarrow (\rho \Rightarrow \mathtt{L}(\rho) \Rightarrow \sigma \Rightarrow \sigma) \Rightarrow \sigma. \end{aligned}$$

The conversion rules are as known from Gödel's T

$$\begin{aligned} \mathcal{R}_{\mathtt{B}}(\mathrm{tt}, f, g) &= f, \qquad & \mathcal{R}_{\mathtt{B}}(\mathrm{ff}, f, g) &= g \\ \mathcal{R}_{\mathtt{N}}(0, f, g) &= f, \qquad & \mathcal{R}_{\mathtt{N}}(Sn, f, g) &= g(n, \mathcal{R}_{\mathtt{N}}(n, f, g)) \\ \mathcal{R}_{\mathtt{L}}(\mathsf{nil}, f, g) &= f, \qquad & \mathcal{R}_{\mathtt{L}}(n :: l, f, g) &= g(n, l, \mathcal{R}_{\mathtt{L}}(l, f, g)). \end{aligned}$$

$\mathcal{C}$ ("Cases") correspond to the $\mathsf{if}\,()\,\mathsf{then}\,\mathsf{else}$ constructs. Their typing is:

$$\begin{aligned} \mathcal{C}^{\sigma}_{\mathtt{N}} &: \mathtt{N} \Rightarrow \sigma \Rightarrow (\mathtt{N} \Rightarrow \sigma) \Rightarrow \sigma \\ \mathcal{C}^{\sigma}_{\mathtt{L}(\rho)} &: \mathtt{L}(\rho) \Rightarrow \sigma \Rightarrow (\rho \Rightarrow \mathtt{L}(\rho) \Rightarrow \sigma) \Rightarrow \sigma, \end{aligned}$$

and the conversion rules are:

$$\begin{aligned} \mathcal{C}_{\mathtt{N},\sigma}(n, f, g) &= \mathsf{if}\,(n = 0)\ \text{then}\ f\ \text{else}\ g\,(n-1), \\ \mathcal{C}_{\mathtt{L},\sigma}(l, f, g) &= \mathsf{if}\,(l = \mathsf{nil})\ \text{then}\ f\ \text{else}\ (g\ l_0\ \text{cdr}\ l), \end{aligned}$$

with (cdr l) denoting the list l without its head element.

Notation 2.3. *The element at position $m^{\mathtt{N}}$ in the list l is denoted by l_m, with l_0 the head of the list. For the length of the list l we use the notation $|l|$. Sometimes, when referring to lists over* N *we omit the type parameter.*

We allow a special unit type which we denote by ε and, by abuse of notation, take the terms of this special type to be also ε. As we will discuss in Section 3.4 on realizability, the ε is associated with computationally irrelevant formulas. We make the following conventions:

$$\begin{aligned} \varepsilon t &:= \varepsilon, & t\varepsilon &:= t, & \varepsilon\varepsilon &:= \varepsilon \\ (\rho \Rightarrow \varepsilon) &:= \varepsilon, & (\varepsilon \Rightarrow \sigma) &:= \sigma, & (\varepsilon \Rightarrow \varepsilon) &:= \varepsilon. \end{aligned} \qquad (\varepsilon\ \text{red.})$$

Definition 2.3 (Formulas)**.** *Let P be a predicate variable of arity n and signature $\rho_1 \times \ldots \times \rho_n$, $t_1, \ldots, t_n$ typed terms and r a boolean term.*

Prime formulas*, also called* atoms *are constructs of the form*

$$P(t_1^{\rho_1} \ldots, t_n^{\rho_n}) \mid \mathsf{Eq}(r, tt),$$

where Eq *is the Leibniz Equality as defined in Section 2.2 and* $\mathsf{Eq}(r, tt)$ *lifts the boolean term r to a formula.*

Formulas *are built from prime formulas by the connectors $\to$, $\wedge$ and the universal quantifier $\forall$*[1]*:*

$$A, B := P(t_1^{\rho_1}, \ldots, t_n^{\rho_n}) \mid \bot \mid A \to B \mid A \wedge B \mid \forall x^{\sigma} A.$$

[1] We refer generically to the connectors $\to$, $\wedge$ and the universal quantifier $\forall$ as *operators*.

Notation 2.4. *Formulas are also denoted by* $\psi, \phi, ...$

In addition to $\rightarrow$ and $\forall$, we use the classical (or weak) logical operators $\tilde{\exists}$ and $\tilde{\vee}$, but regard them only as abbreviations (see page 14).

Notation 2.5. *We make the following conventions:*

- $\forall$ *and* $\neg$ *bind stronger than* $\wedge$*, which binds stronger than* $\rightarrow$*.*
- $\rightarrow$ *is associative to the right:*

  $A \rightarrow B \rightarrow C$ *abbreviates* $A \rightarrow (B \rightarrow C)$ *and is equivalent to* $A \wedge B \rightarrow C$*.*

- *we use the "dot"-notation, whenever this saves on parentheses*

  $$\forall_x.\ A_1(x) \rightarrow \cdots \rightarrow A_n(x) \text{ stands for } \forall_x(A_1(x) \rightarrow \cdots \rightarrow A_n(x)).$$

  The scope of x *extends up to the closing parenthesis corresponding to the last one opened before* $\forall_x$*. If all parenthesis opened before* $\forall_x$ *are also closed before* $\forall_x$*, then* x *is bound by the universal quantifier for the remaining subformula.*

Notation 2.6. *We use the abbreviations* $\vec{A}$ *for* $A_1 ... A_n$ *and* $\vec{A} \rightarrow B$ *for* $A_1 \rightarrow ... \rightarrow A_n \rightarrow B$*.*

Definition 2.4. *A formula is said to be* arithmetical *if the quantifiers do not range over predicate variables.*

Formulas are always arithmetical in our system, so no higher order quantifiers are introduced in the language. Since they are viewed as place holders for comprehension terms, the predicate variables allow us to "simulate" part of the higher order logic. At the same time, our system is conservative over the first order logic.

Definition 2.5. (a) *Let* A *be an arithmetical formula.* A comprehension term *is an entity of the form*

$$\lambda x_1, \ldots, x_n.A,$$

also denoted by $\{\, \vec{x} \mid A \,\}$*, where* $\vec{x} = \{x_1, ..., x_n\}$ *may appear free in* A*, which may contain further free variables.*

(b) *A predicate variable of arity* n *is seen as a place holder for a comprehension term* $\lambda x_1, \ldots, x_n.A$ *as defined above.*

Remark 2.1. *In the comprehension term* $\{\,\vec{x} \mid A\,\}$*, the formula* A *may contain predicate variables.*

The capture-free substitutions $M[x/t]$ and $M[u/N]$, are defined inductively as usual (see for instance (Troelstra and Schwichtenberg, 2000)):

Definition 2.6 (Substitution)**.** *Let* $M[x/t]$ *denote the substitution of the term* t *for the variable* x *in the term/formula* M*. We have:*

$$\begin{aligned} x[x/t] &:= t \\ y[x/t] &:= y, \text{ when } y \neq x \\ (M_1\, M_2)[x/t] &:= (M_1[x/t])\,(M_2[x/t]) \\ (\lambda x.\, M)[x/t] &:= \lambda x.\, M \\ (\lambda y.\, M)[x/t] &:= \lambda y.\, M[x/t], \text{ when } y \neq x, \end{aligned}$$

where $y \notin \mathrm{FV}(t)$*.*

Notation 2.7. $B[X/\{\,\vec{x} \mid A\,\}]$*,* $B[\{\,\vec{x} \mid A\,\}]$ *or* $B[A]$ *are used to denote the substitution of the comprehension term* $\{\,\vec{x} \mid A\,\}$ *for the predicate variable* X *in the formula* B*.*

Definition 2.7. *A formula of the form* $\forall_{\vec{x}}\exists_{\vec{y}}\vec{A}(\vec{x},\vec{y})$*, with* $\vec{A} \in \mathsf{NA}^{\omega}$ *quantifier free, is referred to as a* Π^2_0*-formula.*

Definition 2.8. *We define* positive, negative, strictly positive *subformula inductively* [2]

(a) *A is a (strictly) positive subformula of itself*

(b) *if $B \wedge C$ is a (strictly) positive (or negative) subformula of A, then so are B and C*

(c) *if $\forall_x B(x)$ is a (strictly) positive (or negative) subformula of A, then so is $B(r)$, for any r free for x in B*

(d) *if $B \to C$ is a positive (negative) subformula of A, then B is a negative (positive) subformula of A and C a positive (negative) subformula of A*

(e) *if $B \to C$ is a strictly positive subformula of A, then so is C.*

Notation 2.8. *We abbreviate "Truth"given by the formula* $\mathsf{Eq}(tt, tt)$ *as* T*.*

AxT *stands for the "Truth Axiom",* Ind *for the Induction Axioms and* Cases *for the Case Distinction Axioms.*

[2] We use the formulation from (Schwichtenberg and Wainer, 2011)

Definition 2.9. Proof terms (M, N) *of* NA^ω *are defined as follows:*

$$
\begin{array}{lll}
M, N & ::= & u^A \mid (\lambda u^A M^B)^{A\to B} \mid (M^{A\to B} N^A)^B \mid \\
& (*) & (\lambda x^\rho M^{A(x)})^{\forall_{x^\rho} A(x)} \mid (M^{\forall_{x^\rho} A(x)} t^\rho)^{A(t)} \mid \\
& & \mathsf{AxT} : \mathrm{T} \mid \mathsf{Ind}^{\forall_{n^\mathtt{N}}.\, A(0)\to\forall_{n^\mathtt{N}}(A(n)\to A(\mathsf{S}n))\to A(n)}_{\mathtt{N},A(n)} \mid \\
& & \mathsf{Ind}^{\forall_{b^\mathtt{B}}.\, A(tt)\to A(ff)\to A(b)}_{\mathtt{B},A(b)} \mid \mathsf{Ind}^{\forall_{l^\mathtt{L}}.\, A(\mathsf{nil})\to\forall_{n^\mathtt{N},l'^\mathtt{L}}(A(l')\to A(n::l'))\to A(l)}_{\mathtt{L},A(l)} \mid \\
& & \mathsf{Cases}^{\forall_{n^\mathtt{N}}.\, A(0)\to\forall_n A(\mathsf{S}n)\to A(n)}_{\mathtt{N},A(n)} \mid \mathsf{Cases}^{\forall_{l^\mathtt{L}}.\, A(\mathsf{nil})\to\forall_{n^\mathtt{N},l'^\mathtt{L}} A(n::l')\to A(l)}_{\mathtt{L},A(l)},
\end{array}
$$

with $(*)$ *being the variable condition that the object variable* x *does not occur freely in any of the open assumptions of* M*.*

The sets of free variables $\mathrm{FV}(M)$ and free (open) assumption variables $\mathrm{FA}(M)$ are introduced in Section 2.4.

2.2 Weak and Strong Operators

As we will see in Chapter 3 on the refined A-Translation - in particular Lemma 3.6 -, when transforming the proofs by the Gödel-Gentzen- or the A-Translation, negations play a special role. We therefore distinguish between the arithmetical falsity F and the logical falsity $\bot$. The first is defined as a constant, while the latter is taken to be a predicate variable, which allows to substitute arbitrary formulas for $\bot$.

In order to introduce the arithmetical falsity, we consider the *Leibniz equality*, defined inductively by the predicate $\mathsf{Eq}(x, y)$ in terms of

- the introduction axiom: $\mathsf{Eq}^+ : \forall_x \mathsf{Eq}(x^\rho, x^\rho)$
- the elimination axiom: $\mathsf{Eq}^- : \forall_{x,y}\big(\mathsf{Eq}(x, y) \to \forall_z C(z, z) \to C(x, y)\big)$.

With this, we have the following forms of negation:

- F, defined as $\mathsf{Eq}(\mathrm{ff}, \mathrm{tt})$
- $\bot$, the (logical) falsity, defined as a 0-ary predicate variable and thus a (special) prime formula.

Notation 2.9. *We use the abbreviations* $\neg A := A \to F$ *and* $\tilde{\neg} A := A \to \bot$*.*

Remark 2.2. *When we do not need/want a predicate variable for "falsity", i.e. in the case when we want to inhibit substitution in a negated formula, we use the arithmetical falsity,* F*.*

The classical (or weak) logical operators are viewed only as abbreviations and we use the tilde ($\tilde{\exists}$, $\tilde{\vee}$), in order to distinguish them from the usual (strong or constructive) operators.

The existential quantifier. In the situations when we need to refer to existentially-quantified formulas in a constructive (strong) sense, we will use the standard notation, $\exists_x A$. Otherwise, $\tilde{\exists}$ has the following meaning:

$$\tilde{\exists}_x A := \tilde{\neg}(\forall_x \tilde{\neg} A) = \forall_x (A \to \bot) \to \bot.$$

If needed, the strong existential quantifier $\exists$ can be defined in terms of introduction and elimination axioms.

Disjunction. $\tilde{\vee}$ can be expressed in many ways, due to the symmetry. The interpretation is given by:

$$A \,\tilde{\vee}\, B := \tilde{\neg} A \to \tilde{\neg} B \to \bot.$$

We will consider instead the formulas $\tilde{\neg} A \to B$ or $\tilde{\neg} B \to A$ when this does not come in conflict with the logic, i.e., camouflage a use of $\mathsf{Stab}^{\bot}$.

If needed, (strong) $\vee$ can be introduced by axioms, as on page 25.

Conjunction. If the conjunction appears in the kernel of a classical existence statement, then when unfolding the formula, $\wedge$ needs to be changed to $\to$. More precisely, $\tilde{\exists}_x (A \wedge B)$ should unfold to $\tilde{\neg}\forall_x .\ A \to B \to \bot$ and not to $\tilde{\neg}\forall_x .\ A \wedge B \to \bot$. To make this explicit, we use the notation $\tilde{\wedge}$:

$$\tilde{\exists}_x (A \,\tilde{\wedge}\, B) := \forall_x (A \to B \to \bot) \to \bot.$$

Otherwise, the conjunction has the known semantics and is given by the Natural Deduction rules on page 20.

2.3 Axioms

2.3.1 Induction and Cases

We use the following induction schemes

$$\begin{aligned}
&\mathsf{Ind}_{b,A} \colon \forall_{b^{\mathbb{B}}} .\ A(\mathrm{tt}) \to A(\mathrm{ff}) \to A(b),\\
&\mathsf{Ind}_{n,A} \colon \forall_{m^{\mathbb{N}}} .\ A(0) \to \forall_{n^{\mathbb{N}}} (A(n) \to A(\mathsf{S}n)) \to A(m),\\
&\mathsf{Ind}_{l,A} \colon \forall_{l^{\mathbb{L}(\rho)}} .\ A(\mathsf{nil}) \to \forall_{x^{\mathbb{N}}, l'^{\mathbb{L}(\rho)}} (A(l') \to A(x :: l')) \to A(l),
\end{aligned}$$

and the case distinction

$$\begin{aligned}\mathsf{Cases}_{m,A} &: \forall_{m^{\mathbb{N}}}.\ A(0) \to \forall_n A(\mathsf{S}n) \to A(m)\\ \mathsf{Cases}_{l,A} &: \forall_{l^{\mathbb{L}(\rho)}}.\ A(\mathsf{nil}) \to \forall_{n,l'} A(n :: l') \to A(l)\end{aligned}$$

2.3.2 efq and Stab

As a consequence of distinguishing between F and $\bot$, the axioms involving negation have two forms, as is the case with "Ex falso quodlibet" (efq) and the "Law of Stability" (Stab) :

$$\begin{aligned}\mathsf{efq}_A &: \mathrm{F} \to A & \mathsf{efq}^\bot{}_A &: \bot \to A\\ \mathsf{Stab}_A &: \neg\neg A \to A & \mathsf{Stab}^\bot{}_A &: \tilde\neg\,\tilde\neg A \to A,\end{aligned}$$

For reasons that will become clear once we introduce the refined A-Translation, we choose to work in minimal logic, hence we do not allow $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$. We will explain this in more detail in Section 8.1. However, for formulas A which do not have free predicate variables we can prove in minimal logic both efq_A and Stab_A.

Lemma 2.1. *For any formula A with no free predicate variables,*

$$\vdash_m \mathsf{efq}_A \text{ and } \vdash_m \mathsf{Stab}_A.$$

Proof. We show the two principles by structural induction on A. In the derivation trees IH stands for Induction Hypothesis. The proofs are presented in Natural Deduction style (see Section 2.4 for the rules).

$\vdash \mathsf{efq}_A$

Case A atomic. We use AxT and Ind_B.

By $\mathsf{Ind}_{b,\mathsf{efq}}$, it suffices to show efq_F and efq_T. We have

$$\dfrac{\mathsf{Ind}_{b,\mathsf{efq}_A} \qquad \dfrac{u\colon \mathrm{F}}{\mathrm{F}\to\mathrm{F}}\to^+ u \qquad \dfrac{\mathsf{AxT}}{\mathrm{F}\to\mathrm{T}}\to^+}{\mathsf{efq}_A}\to^-$$

Case $A \to B$

$$\dfrac{\dfrac{\text{IH: } \mathrm{F}\to B \qquad u:\mathrm{F}}{B}\to^-}{\dfrac{A\to B}{\mathrm{F}\to(A\to B)}\to^+ u}$$

Case $A \wedge B$

$$\dfrac{\dfrac{\text{IH: } \mathrm{F} \to A \qquad u : \mathrm{F}}{A} \to^- \qquad \dfrac{\text{IH: } \mathrm{F} \to B \qquad u : \mathrm{F}}{B} \to^-}{\dfrac{A \wedge B}{F \to A \wedge B} \to^+ u} \wedge^+$$

Case $\forall_x A$

$$\dfrac{\dfrac{\dfrac{\text{IH: } \mathrm{F} \to A \qquad u : \mathrm{F}}{A} \to^-}{\forall_x A} \forall^+ x}{F \to \forall_x A} \to^+ u$$

$\vdash \mathsf{Stab}_A$

Case A atomic. We use AxT and $\mathsf{Ind}_{\mathsf{B}}$.

By $\mathsf{Ind}_{b,\mathsf{Stab}}$, it suffices to show $\mathsf{Stab}_{\mathrm{F}}$ and $\mathsf{Stab}_{\mathrm{T}}$. We have

$$\dfrac{\mathsf{Ind}_{b,\mathsf{efq}_A} \qquad \dfrac{u_2 \colon \neg\neg\mathrm{F} \qquad \dfrac{\dfrac{u_1 \colon \mathrm{F}}{\mathrm{F} \to \mathrm{F}} \to^+ u_1}{\mathrm{F}} \to^-}{\neg\neg\mathrm{F} \to \mathrm{F}} \to^+ u_2 \qquad \dfrac{\mathsf{AxT}}{\neg\neg\mathrm{T} \to \mathrm{T}} \to^+}{\mathsf{Stab}_A} \to^-$$

Case $A \to B$

We assume $u_1 \colon \neg\neg(A \to B)$ and $u_2 : A$ and show B.

$$\dfrac{\text{IH: } \neg\neg B \to B \qquad \dfrac{u_1 \qquad \dfrac{\dfrac{u_4 \colon \neg B \qquad \dfrac{u_3 \colon A \to B \qquad u_2}{B} \to^-}{\mathrm{F}} \to^+ u_3}{\neg(A \to B)} \to^-}{\mathrm{F}} \to^+ u_4}{\neg\neg B} \to^-}{B}$$

Case $A \wedge B$.

We assume $u_1 \colon \neg\neg(A \wedge B)$ and show $A \wedge B$. We give only the derivation of A, since B can be inferred in a very symmetrical way. $A \wedge B$ can be then obtained by $\wedge^+$.

$$\dfrac{\text{IH: } \neg\neg A \to A \qquad \dfrac{\dfrac{u_1 \qquad \dfrac{\dfrac{u_2\colon \neg A \qquad \dfrac{u_3\colon A \wedge B}{A}\wedge^-}{F}\to^- }{\neg(A\wedge B)}\to^+ u_3}{F}\to^-}{\neg\neg A}\to^+ u_2}{A}\to^-$$

Case $\forall_x A$.

We assume $u_1\colon \neg\neg\forall_x A$, fix an arbitrary x and show A.

$$\dfrac{\text{IH: } \neg\neg A \to A \qquad \dfrac{\dfrac{u_1 \qquad \dfrac{\dfrac{u_3\colon \neg A \qquad \dfrac{u_2\colon \forall_x A \qquad x}{A}\forall^-}{F}\to^- }{\neg\forall_x A}\to^+ u_2}{F}\to^-}{\neg\neg A}\to^+ u_3}{A}\to^-$$

□

Remark 2.3. *In general, one can show in minimal logic* $\mathsf{Stab}^\perp_{\tilde\neg A}$*, for any* A*. In particular, due to the interpretation given to* $\tilde\exists$ $\mathsf{Stab}^\perp_{\tilde\exists_x A}$ *is provable in minimal logic, as follows*

$$\dfrac{w\colon\ \tilde\neg\,\tilde\neg(\,\tilde\neg(\forall_x\,\tilde\neg A)) \qquad \dfrac{\dfrac{u\colon\ \tilde\neg(\forall_x\,\tilde\neg A) \qquad v\colon \forall_x\,\tilde\neg A}{\perp}\to^-}{\tilde\neg(\,\tilde\neg(\forall_x\,\tilde\neg A))}\to^+ u}{\perp}\to^-}{\tilde\neg(\forall_x\,\tilde\neg A)}\to^+ v$$

2.3.3 Other Axioms

We also allow in NA^ω the Axiom of Choice (AC) and the Axiom of Dependent Choice (DC):

$$\text{AC}\colon \forall_x\tilde\exists_y P(x,y) \to \tilde\exists_f\forall_x P(x,fx)$$
$$\text{DC}\colon \forall_{n,x}\tilde\exists_y P(n,x,y) \to \forall_x\tilde\exists_f.\ f(0)=x\ \tilde\wedge\ \forall_n P(n,fn,f(\mathsf{S}n))$$

2.4 Natural Deduction

We work with proofs in Natural Deduction-style. Introduced by (Gentzen, 1934), the system expresses the mathematical reasoning in terms of infer-

ence rules. The set of rules come in pairs - introduction and elimination, respectively.

We summarize in Table 2.1 the rules for the connectors that we have defined. For the $\forall^+$-rule, the (Eigen-) variable condition that the variable x is free in the open/free assumptions of the derivation M[3]:

$$(*)x \notin \mathsf{FV}(\mathsf{FA}(M^A))$$

needs to be fulfilled.

Definition 2.10. *The set of* free variables *of a derivation M, denoted by $\mathsf{FV}(M)$, is defined inductively in the standard way:*

$$\begin{aligned}\mathsf{FV}(x) &= \{x\}\\ \mathsf{FV}(M\,N) &= \mathsf{FV}(M) \cup \mathsf{FV}(N)\\ \mathsf{FV}(\lambda x.M) &= \mathsf{FV}(M) \setminus \{x\}\\ \mathsf{FV}(M\,t) &= \mathsf{FV}(M) \cup \{t\}\end{aligned}$$

As can be seen in the table, Gentzen's proof calculus allows us to illustrate the Curry-Howard correspondence, by which each inference rule has an associated λ-term. By viewing formulas as types we are able to extract programs from the intuitionistic proofs. In what follows, we use alternatively the representation of proofs as natural deduction trees and as *lambda*-terms, having this correspondence in mind.

Notation 2.10. *When writing proofs in natural deduction style we sometimes write $[u : A]$ to mark that the assumption A is discharged in the proof.*

Notation 2.11. *We will sometimes omit the $\forall$-quantification, when it is clear from the context that the terms appearing free should be universally quantified. In such cases quantification is understood as ranging over the entire formula.*

Definition 2.11. *The set of* free assumptions *of a derivation M, denoted by $FA(M)$, is defined inductively, according to the inference rules:*

$$\begin{aligned}FA(u) &= \{u\}\\ FA((\lambda u^A.M^B)^{A\to B}) &= FA(M) \setminus \{u\}\\ FA((M^{A\to B}N^A)^B) &= FA(M) \cup FA(N)\\ FA((\lambda x.M^A)^{\forall_x A}) &= FA(M)\\ FA((M^{\forall_x A}t)^{A(t)}) &= FA(M)\end{aligned}$$

[3]The sets FV and FA are given by Definitions 2.10 and 2.11 and $\mathsf{FV}(\mathsf{FA}(M))$ is taken to be colection of the free variables in the free assumptions from the derivation M

Inference rule	*λ-term*
$u\colon A$	u^A
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B} \to^+ u$	$(\lambda u^A M^B)^{A\to B}$
$\dfrac{\begin{array}{cc}\mid M & \mid N\\ A \to B & A\end{array}}{B} \to^-$	$(M^{A\to B} N^A)^B$
$\dfrac{\begin{array}{c}\mid M\\ A\end{array}}{\forall_x A} \forall^+ x \quad (*)$	$(\lambda x M^A)^{\forall_x A} \quad (*)$
$\dfrac{\begin{array}{cc}\mid M & \\ \forall_x A & r\end{array}}{A(r)} \forall^-$	$(M^{\forall_x A} r)^{A(r)}$
$\dfrac{\begin{array}{cc}\mid M & \mid N\\ A & B\end{array}}{A \wedge B} \wedge^+$	$\langle M^A, N^B\rangle^{A\wedge B}$
$\dfrac{\begin{array}{ccc} & [u\colon A] \quad [v\colon B] & \\ \mid M & & \mid N\\ A \wedge B & & C\end{array}}{C} \wedge^- u, v$	$(M^{A\wedge B}(u^A, v^B, N^C))^C$

Table 2.1: Derivation terms for $\to$, $\forall$ and $\wedge$

2.5 Summary of the Chapter

The Logic In this chapter we introduce the system NA^ω, which builds on Gödel's System T and is a negative fragment of Heyting's Arithmetic with finite types. We carry out the proofs in *minimal logic*, but when not clear from the context or when we want to emphasize in which system the formulas are valid, we use the following notation:

$\vdash_m$ for derivability in minimal logic

$\vdash_i$ for derivability in intuitionistic logic, i.e., minimal logic + $\mathsf{efq}^\bot$

$\vdash_c$ for derivability in classical logic, i.e., intuitionistic logic + $\mathsf{Stab}^\bot$

A discussion at length on ${\mathsf{efq}^\bot}_A$ and the reason why we are excluding it in addition to $\mathsf{Stab}^\bot$, in connection with the observations made in Subsection 2.3.2, is carried out in Section 8.1.

In our (minimal) setting, the distinction between classical and constructive proofs relies (only) in the semantics that we fix a-priori for the existential quantifier. As seen in this chapter, the classical existential quantifier is viewed as an abbreviation and thus proofs of $\tilde{\exists}$-formulas consist in deriving $\bot$ and therefore they can be regarded as proofs by contradiction. To recover the computational (or constructive) content in the classical proofs, we rely on the fact that the Π^0_2-formulas are equiderivable in both systems and use the translation mechanism introduced in the following chapter.

Summary of this chapter. Among the key notions used throughout this thesis, we define in Section 2.1 the types, terms, formulas and proof terms. Since the terms are built using also the recursion operators which will be associated to the induction axioms when extracting programs from proofs, we have presented in this chapter the corresponding reduction rules. We have given the main notations and abbreviations used in this thesis.

Special care is payed to the weak operators $\tilde{\forall}$, $\tilde{\exists}$, $\tilde{\wedge}$ and $\tilde{\neg}$, which we see only as abbreviation and treat in Section 2.2. We distinguish among the logical falsity $\bot$ and the arithmetical F, since we want to allow substitution of the former when A-translating the proofs. Consequently, the axioms efq and $\mathsf{efq}^\bot$ require special treatment, as presented in Section 2.3.2.

Proofs are presented both as derivation trees in natural deduction style and as λ-terms. In Section 2.4 we have presented the connection between the two.

Chapter 3

A-Translation and its Refined Version

By working in a fragment of classical logic in which $\exists$ is not explicit, but rather an abbreviation, the proofs of Π^0_2-formulas do not provide a witness in an explicit manner. For this reason, recovering the computational content of existential goals cannot be done in a direct way by the Curry-Howard correspondence. Therefore, we need to apply first a transformation method on the proofs, in order to lift them to a form in which the computational content becomes explicit. In order to do this, we have to consider a special treatment of the negation, which plays in this process the role of a place holder for "strong" instances of the existential goal formula.

This chapter is organized as follows: Section 3.1 gives a brief exposition of the Gödel-Gentzen translation which, combined with Friedman's trick, results in the so-called "A-Translation" presented in Section 3.2. In this dissertation we work with a refinement due to (Berger et al., 2002), which we overview in detail in Section 3.3. The key idea evolves around the concept of definite/goal formulas, which are restricted classes introduced in order to improve the original A-Translation. We will present them in Section 3.3.1 in a manner close to (Ishihara, 2000)'s approach, since we believe that this illustrates better their structure. Once the proofs are lifted to their constructive counterparts, we are able to unravel their computational content by the extraction rules summarized in Section 3.4. We conclude this chapter by comparing in Section 3.5 the classes defined for the purpose of refining the A-Translation and the ones introduced by (Ishihara, 2000) in order to determine which classical formulas are provable in constructive logic.

3.1 The Double Negation Translation

A first step towards identifying the computational content in classical proofs consists in translating them into constructively valid ones. In order to achieve this, variants of the *Double-Negation Translation* have been proposed in the literature. The first such translation, also called the *Negative Translation*, has been introduced (independently) by (Gödel, 1933) and (Gentzen, 1936). The method consists in double negating all atomic formulas, a process by which also the translation of classical principles become constructively valid.

Definition 3.1. *Gödel-Gentzen's Negative Translation* $\cdot^g$ *is given by*

$$\begin{array}{ll} \bot^g & := \bot, \\ \psi^g & := \neg\neg\psi, \psi \text{ atomic} \\ (\psi_1 \wedge \psi_2)^g & := \psi_1^g \wedge \psi_2^g, \\ (\psi_1 \to \psi_2)^g & := \psi_1^g \to \psi_2^g, \\ (\forall_x \psi)^g & := \forall_x \psi^g. \end{array}$$

Lemma 3.1. *For any formula* A *of* NA^ω*,*

$$\vdash_m (\mathsf{Stab}^\bot{}_A)^g \text{ and } \vdash_m (\mathsf{efq}^\bot{}_A)^g.$$

Proof. By induction on A. Details in (Schwichtenberg and Wainer, 2011). □

This translation preserves the classical validity of the formulas, as shown by the following theorem. More importantly, as a consequence of Lemma 3.1 the double-negated formulas become NA^ω-valid.

Theorem 3.1. *For any formula* A *of* NA^ω *we have*

(a) $\vdash_c A \leftrightarrow A^g$,

(b) $\Gamma \vdash_c A$ *iff* $\Gamma^g \vdash_m A^g$, *where* $\Gamma^g := \{ B^g \mid \text{for all } B \in \Gamma \}$.

Proof. By structural induction on A for (a) and induction on the derivation for (b). The latter follows for each rule from the induction hypothesis using again the same rule, because the Negative Translation acts as a homomorphism for the connectives of NA^ω (Troelstra and Schwichtenberg, 2000). □

However, in the case when A is a Π^0_2-formula, the translation does not help us recover the constructive meaning of the existential quantifier. As pointed out in the previous chapter (see Section 2.2 for an explanation on the weak operators), a classical proof of $\tilde{\exists}_x A$ consists in deriving a contradiction from the assumption $\forall_x.A \to \bot$. It is in general not easy to read of the means for constructing x, so the aim is to translate this proof in such a way that the information on how to obtain the witness for x is recovered. The following section presents a solution to this problem.

3.2 A-Translation

3.2.1 Friedman's "trick"

The idea of A-translation is due (independently) to Friedman and Dragalin (Dragalin, 1979; Friedman, 1978) and is based on what is known as "Friedman's trick": "Define the A-Translation φ_A of φ to be the result of simultaneously replacing each atomic subformula ψ of φ by $(\psi \vee A)$", where A is an arbitrary formula ((Friedman, 1978)).

Before we give a more formal definition of Friedman's trick, we need to introduce the (strong) disjunction, which we otherwise use only in its weak form, as an abbreviation (see Section 2.2, page 15). The disjunction is given by the following introduction and elimination axioms:

$$\vee^{+,r} : A \to A \vee B \qquad \vee^{+,l} : B \to A \vee B$$
$$\vee^{-} : A \vee B \to (A \to C) \to (B \to C) \to C$$

Remark 3.1. *It is an easy exercise to show in intuitionistic logic by the axioms $\vee^{+,l}$, $\vee^{-}$ and $\mathsf{efq}^{\bot}{}_A$ that $\bot \vee A \leftrightarrow A$, for A arbitrary. By this, when $\psi := \bot$ Friedman's "trick" amounts to substituting $\bot$ by A.*

The following inductive definition rephrases Friedman's "trick" formally:

Definition 3.2 (Friedman's "trick")**.** *Let A be any formula from our language NA^ω. For a formula $\psi \in \mathsf{NA}^\omega$, we define its translation ψ_A to be:*

$$\begin{aligned}
\psi_A &:= \psi \vee A, \text{ for } \psi \text{ atomic} \\
(\psi_1 \wedge \psi_2)_A &:= (\psi_1)_A \wedge (\psi_2)_A, \\
(\psi_1 \to \psi_2)_A &:= (\psi_1)_A \to (\psi_2)_A \\
(\forall_x \psi)_A &:= \forall_x \psi_A.
\end{aligned}$$

By Remark 3.1 $\bot_A := A$.

Friedman's trick preserves provability, so it is an easy exercise to show by induction on the formula ψ that:

Theorem 3.2. *(Friedman, 1978) For φ, ψ formulas and φ_A, ψ_A their corresponding translations by Friedman's trick, we have that*

$$\textit{if } \varphi \vdash_i \psi \textit{ then } \varphi_A \vdash_i \psi_A.$$

We point out that Friedman's trick is applied to intuitionistic proofs. Therefore, if one works in the classical setting, the Gödel-Gentzen translation[1] is necessary in order to first transform the classical proofs into intuitionistic ones, as depicted in Figure 3.1.

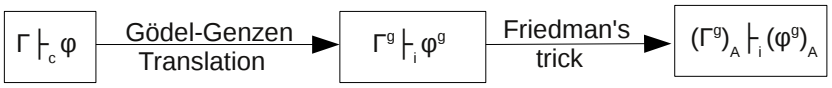

Figure 3.1: Proof translation: from classical to intuitionistic proofs

The achievement of combining the two translation methods is that not only classical proofs are transposed in the constructive setting, but that furthermore, proofs of $\bot$ can be turned into proofs of A (by Remark 3.1). Since A is arbitrary, we can take it to be $\exists_x G$. By this we turn a classical proof of a $\tilde{\exists}_x G$, so of $\bot$ from $\forall_x \tilde{\neg} G$, into a constructive proof of $\exists_x G$.

3.2.2 A-Translation

As Figure 3.1 shows, the "A-Translation" is given by combination of Friedman's trick and the Gödel-Gentzen translation.

Notation 3.1. *We use in what follows the abbreviation $\neg_A \psi := \psi \to A$.*

Definition 3.3. *Let A be some arbitrary formula. The Friedman-Dragalin*

[1] We are not limited to the Gödel-Gentzen translation, but can use some equivalent form of a double-negation translation

A-Translation $\cdot^{fd}$ *is given inductively by:*

$$\begin{aligned}
\bot^{fd} &:= A,\\
\psi^{fd} &:= \neg_A\neg_A\psi,\ \psi \textit{ atomic},\\
(\psi_1 \wedge \psi_2)^{fd} &:= \psi_1^{fd} \wedge \psi_2^{fd},\\
(\psi_1 \to \psi_2)^{fd} &:= \psi_1^{fd} \to \psi_2^{fd},\\
(\forall_x \psi)^{fd} &:= \forall_x \psi^{fd}
\end{aligned}$$

We show in what follows that the A-Translation is indeed a combination of the Double-Negation Translation and Friedman's trick.

Lemma 3.2. *For any formula* $\psi \in \mathsf{NA}^\omega$, $\vdash (\psi^g)_A \leftrightarrow \psi^{fd}$.

Proof. By induction on the structure of ψ, using Definitions 3.1 and 3.2.

Case $\bot$ $(\bot^g)_A = \bot_A = A$, by Remark 3.1.

Case ψ **atomic**

$$(\psi^g)_A := (\neg\neg\psi)_A = \neg_A\neg_A\psi_A = \neg_A\neg_A(\psi \vee A) \text{ and } \psi^{fd} = \neg_A\neg_A\psi.$$

"$\to$": $\neg_A\neg_A(\psi \vee A) \to \neg_A\neg_A\psi$

$$\dfrac{w : \neg_A\neg_A(\psi \vee A) \qquad \dfrac{\dfrac{v : \psi \vee A \qquad [A] \qquad \dfrac{u : \psi \to A \qquad [\psi]}{A}\to^-}{A}\vee^-}{(\psi \vee A) \to A}\to^+ v}{\dfrac{A}{\psi^{fd}}\to^+ u}\to^-$$

"$\leftarrow$": $\neg_A\neg_A\psi \to \neg_A\neg_A(\psi \vee A)$

$$\dfrac{v : \neg_A\neg_A\psi \qquad \dfrac{\dfrac{w : \neg_A(\psi \vee A) \qquad \dfrac{u : \psi}{\psi \vee A}\vee^+}{A}\to^-}{\psi \to A}\to^+ u}{\dfrac{A}{(\psi^g)_A}\to^+ w}\to^-$$

Cases $\psi_1 \circ \psi_2$, with $\circ \in \{\wedge, \to\}$. On one hand, $(\psi_1 \circ \psi_2)^g_A = (\psi_1^g)_A \circ (\psi_2^g)_A$ and on the other, $(\psi_1 \circ \psi_2)^{fd} = \psi_1^{fd} \circ \psi_2^{fd}$. The equivalence follows therefore easily by the (IH)[2].

[2]We denote the induction hypothesis by (IH)

Case $\forall_x \psi$ This follows also directly by the (IH), since $(\forall_x \psi)^g_A = \forall_x (\psi^g)_A$ and $(\forall_x \psi)^{fd} = \forall_x (\psi)^{fd}$.

□

The A-translated versions of classical axioms become intuitionistically provable, as it is the case with $\mathsf{Stab}^{\perp}$. Based on a result from (Friedman, 1978) we have:

Lemma 3.3. *For any formula* ψ, $\vdash_i \neg_A \neg_A \psi^{fd} \to \psi^{fd}$.

A-Translation hence allows the transition from classical proofs to their constructive equivalents, as depicted in Figure 3.2 and stated by the following theorem.

Theorem 3.3. *(Friedman, 1978) If* $\psi_1, ..., \psi_k \vdash_c \psi$ *then* $\psi_1^{fd}, ..., \psi_k^{fd} \vdash_i \psi^{fd}$.

Proof. Either by induction on the structure of ψ or immediate by Theorems 3.1 and 3.2 together with Lemma 3.2. □

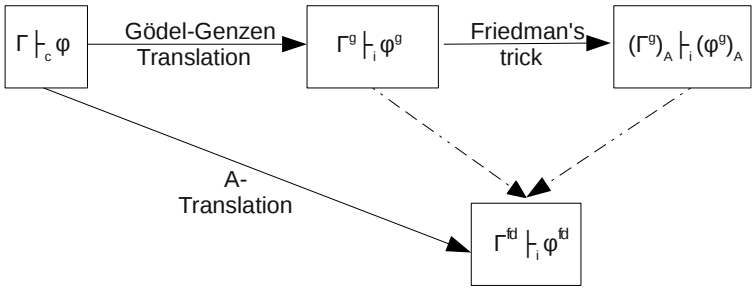

Figure 3.2: A-Translation

Remark 3.2. *Since in general the A-Translation is applied to proof systems involving strong quantifiers, it can be also defined for* $\vee$ *and* $\exists$*:*

$$(\psi_1 \vee \psi_2)^{fd} := \neg_A \neg_A (\psi_1^{fd} \vee \psi_2^{fd})$$
$$(\exists_x \psi)^{fd} \quad := \neg_A \neg_A (\exists_x \psi^{fd}).$$

Example 1. *(Peirce's formula.) Let* P *and* Q *be atomic formulas.*

$$F(P, Q) := ((P \to Q) \to P) \to P$$

can be derived classically, but is not true intuitionistically.

The proof of $F(P,Q)$ needs $\mathsf{Stab}^{\perp}$ and $\mathsf{efq}^{\perp}$, as Figure 3.3 shows.

By Theorem 3.1, $((P \to Q) \to P) \to P$ is classically trivially equivalent to $((P^g \to Q^g) \to P^g) \to P^g$ and the derivation above remains classically valid for P^g and Q^g. Moreover, since $\mathsf{Stab}^{\perp g}$ is derivable constructively by the same theorem, the double negation of the above derivation is intuitionistically (and even NA^{ω}-) valid. Hence, we can completely eliminate $\mathsf{Stab}^{\perp}$ and $\mathsf{efq}^{\perp}$ from the proof of $F^g(P,Q)$ and the minimal proof in Figure 3.4(a).

Since $\mathsf{Stab}^{\perp fd}$ is constructively valid by Lemma 3.3, we can derive the A-translated Peirce's formula by the minimal proof in Figure 3.4(b).

$$\dfrac{\dfrac{v:\neg P \quad \dfrac{\dfrac{\dfrac{v:\neg P \quad \dfrac{w:(P\to Q)\to P \quad \dfrac{\dfrac{v:\neg P \quad u:P}{\perp}\to^{-}}{Q}\,\mathsf{efq}^{\perp}}{P\to Q}\to^{+} u}{P}\to^{-}}{\perp}\to^{+} v}{\neg\neg P}}{P}\,\mathsf{Stab}^{\perp}}{((P\to Q)\to P)\to P}\to^{+} w$$

Figure 3.3: The classical proof of Peirce's formula

This example, however, does not exploit the full power of A-Translation, since the negative translation suffices in order to obtain the constructive proof. Friedman's transformation mainly serves the purpose of transforming the proof of a weak existential formula into a strong (constructive) existence proof. In Chapters 5-7 we will present more elaborated examples, where A-Translation is meaningful and necessary.

3.3 The Refinement of the A-Translation

The A-Translation presented in the previous section, resulting from the composition of the double negation translation and Friedman's "trick", introduces many $\perp$'s, some of which are unnecessary. Each of these $\perp$'s is to be substituted by some formula A. Therefore, when taking $\exists_x G$ for A, all these negations gain computational meaning and as a consequence, the programs extracted from the A-translated proofs are rather cumbersome and contain unnecessary computations.

If we restrict the logic to be minimal, however, double negation is superfluous. We still need Friedman's "trick", since $\tilde{\exists}$ does not have the computa-

$$\dfrac{\dfrac{w : (P^g \to Q^g) \to P^g \qquad \dfrac{\dfrac{u : \neg\neg P \qquad v : \neg P}{\dfrac{\bot}{Q^g} \to^+ \tilde{\neg} Q}\to^-}{P^g \to Q^g} \to^+ u}{\neg\neg P}\to^- \qquad v : \neg P}{\dfrac{\dfrac{\bot}{\neg\neg P} \to^+ v}{((P^g \to Q^g) \to P^g) \to P^g} \to^+ w}\to^-$$

(a) The negative translated proof of Peirce's formula

$$\dfrac{\dfrac{w : (P^{fd} \to Q^{fd}) \to P^{fd} \qquad \dfrac{\dfrac{u : \neg_A\neg_A P \qquad v : \neg_A P}{\dfrac{A}{Q^{fd}} \to^+}\to^-}{P^{fd} \to Q^{fd}} \to^+ u}{\neg_A\neg_A P}\to^- \qquad v : \neg_A P}{\dfrac{\dfrac{A}{P^{fd}} \to^+ v}{((P^{fd} \to Q^{fd}) \to P^{fd}) \to P^{fd}} \to^+ w}\to^-$$

(b) The A-translated proof of Peirce's formula

Figure 3.4: Proofs of Peirce's Formula and its translations

tional content that we want. On the other hand, $\tilde{\exists}$ rewrites into negations, which trigger the use of further negations in the proof, so that the issue of minimizing the number of $\bot$-occurrences remains. Furthermore, in some situations, as we will see in the case studies from Chapters 5-7, further insertions of negations are necessary, if we want to limit ourselves to NA^ω. In Section 8.1 we present an analysis of the distinction between $\bot$ and F and in Chapter 8 we propose a method by which double negations are inserted in a minimal way, in order to obtain for some cases NA^ω-provable formulas.

We follow closely the work of Berger et al. (2002), where a refinement of the A-Translation is proposed, with the purpose of eliminating unnecessary occurrences of $\bot$ from the proofs. The paper presents a mechanism for identifying the relevant occurrences of $\bot$, which are substituted by the strong existential goal. In all other cases, $\bot$ is replaced by the arithmetical falsity F which is a constant[3], so it does not carry any computational meaning.

Remark 3.3. *As discussed in Section 2.2* $\tilde{\exists}$ *and* $\tilde{\vee}$ *are seen in* NA^ω *as abbreviations.* $\wedge$ *can be regarded as a special case of an inductively defined*

[3]See Section 2.2 and Section 8.1 for explanations on the distinction between $\bot$ and F.

predicate and can be given in terms of introduction and elimination schemes. For these reasons, we limit in the following the treatement of the logical operators to $\rightarrow$ *and* $\forall$*.*

We work in this thesis with the following notion of "decidable formulas" slightly extending the one from Seisenberger (2003):

Definition 3.4 (Decidable formulas)**.** *A formula is said to be* decidable *if it is built from prime formulas only by propositional connectives and by quantifiers which are either boolean or range over finite sets of naturals.*

Lemma 3.4 (Case Distinction)**.** *Let* B *be an arbitrary formula and* C *a decidable formula. Then we have*

$$\vdash (\neg C \rightarrow B) \rightarrow (C \rightarrow B) \rightarrow B \qquad \text{(Cases)}$$

Proof. By boolean induction on C. □

3.3.1 Definite and Goal Formulas

We identify in what follows the classes of formulas for which the substitution of $\bot$ by F is possible, as in Berger et al. (2002). We define these classes in the same style as in the study presented in (Ishihara, 2000). We point out that they are distinct classes of formulas, since (Ishihara, 2000) does not restrict the logic to be minimal. We will give a detailed comparison in Section 3.5.

Definition 3.5. *Let* $P \neq \bot$ *range over prime* $\mathcal{L}$*-formulas, and* $\bot$ *be a unary predicate variable. Let* D*,* G*,* R *and* I *range over* $\mathcal{D}, \mathcal{G}, \mathcal{R}_D$ *and* $\mathcal{I}_G$*, respectively, and* D_0 *be a decidable formula from* $\mathcal{D}$*.*

The classes $\mathcal{D}, \mathcal{G}, \mathcal{R}_D$ *and* $\mathcal{I}_G$ *are simultaneously generated by the clauses*

$$\begin{aligned}
&P, \forall_x D, R, I \rightarrow D \ \in \ \mathcal{D}\\
&\bot, I, R \rightarrow G, D_0 \rightarrow G \ \in \ \mathcal{G}\\
&\bot, \forall_x R, G \rightarrow R \ \in \ \mathcal{R}_D\\
&P, \forall_x I, D \rightarrow I \ \in \ \mathcal{I}_G
\end{aligned}$$

Lemma 3.5. *We can make the following generalizations:*

- $\vec{G} \rightarrow R \in \mathcal{R}_D$
- $\vec{I} \rightarrow D \in \mathcal{D}$
- $\vec{R} \rightarrow G \in \mathcal{G}$

- $\vec{D_0} \to G \in \mathcal{G}$
- $\vec{D} \to I \in \mathcal{I}_G$

Proof. Follows by induction on the number of formulas in the premise from the following observations

- if $\vec{G} \to R \in \mathcal{R}_D$ and $G' \in \mathcal{G}$ then $G' \to (\vec{G} \to R) \in \mathcal{R}_D$.
- if $\vec{I} \to D \in \mathcal{D}$ and $I' \in \mathcal{I}_G$ then $I' \to (\vec{I} \to D) \in \mathcal{D}$.
- if $\vec{R} \to G \in \mathcal{G}$ and $R' \in \mathcal{R}_D$ then $R' \to (\vec{R} \to G) \in \mathcal{G}$.
- if $\vec{D_0} \to G \in \mathcal{G}$ an $D'_0 \in \mathcal{D}$ decidable then $D'_0 \to (\vec{D_0} \to G) \in \mathcal{G}$.
- if $\vec{D} \to I \in \mathcal{I}_G$ and $D' \in \mathcal{D}$ then $D' \to (\vec{D} \to I) \in \mathcal{I}_G$.

□

Remark 3.4. *Clearly, $\mathcal{R}_D \subset \mathcal{D}$ and $\mathcal{I}_G \subset \mathcal{G}$.*

Comparison with Berger et al. (2002) In order to bridge Definition 3.5 and the one of the definite and goal formulas from Berger et al. (2002), we recall the latter:

Definition 3.6. (a) *If a formula "ends" with $\bot$, it is said to be* relevant *and otherwise it is called* irrelevant*. Thus, C is a relevant formula iff*

$$C := \bot \mid B \to C \mid \forall_x C$$

(b) *Let P^* range over atomic formulas (including $\bot$). The* definite *formulas D^* and* goal *formulas G^* are defined inductively*

$$\begin{array}{rcll} D^* & := & P^* \mid G^* \to D^* & \text{if } D^* \text{ relevant or } G^* \text{ irrelevant} \\ & & \mid \forall_x D^*, & \\ G^* & := & P^* \mid D^* \to G^* & \text{if } D^* \text{ relevant or } D^* \text{ quantifier-free} \\ & & \mid \forall_x G^* & \text{if } G^* \text{ irrelevant,} \end{array}$$

Remark 3.5. *In the definition of the goal formula it is required that the premise of G^* is quantifier-free. The purpose is to allow case distinction for these formulas and as seen in Lemma 3.4, it suffices to consider decidable formulas, such that the above definition can be extended to:*

$$D^* \to G^* \text{ if } D^* \text{ relevant or decidable.}$$

Proposition 3.1. *Let $\mathcal{D}, \mathcal{G}, \mathcal{R}_D$ and $\mathcal{I}_G$ be the classes of formulas as given by Definition 3.5 and the relevant, definite and goal formulas as in Definition 3.6. Then:*

1. *$\mathcal{D}$ are the* definite *formulas*
2. *$\mathcal{G}$ are the* goal *formulas*
3. *$\mathcal{R}_D$ are the* relevant definite *formulas*
4. *$\mathcal{I}_G$ are the* irrelevant goal *formulas.*

Proof. Let P range over prime $\mathcal{L}$-formulas, $D \in \mathcal{D}$, $G \in \mathcal{G}$, $R \in \mathcal{R}_D$, $I \in \mathcal{I}_G$ and D_0 be a decidable formula from $\mathcal{D}$. Let P^* range over prime formulas, including also $\bot$, and D^*, G^* be as in Definition 3.6.

We prove the claims by simultaneous induction on the formulas.

1. "$\subset$" We show: $P, \forall_x D, R, I \to D$ are the definite formulas.

   Clearly, P is definite.

   $\forall_x D$ is definite, because D is definite by the induction hypothesis.

   Let $R \in \mathcal{R}_D$. If R is $\bot$, it is also definite. If it is $\forall_x R$, then R is by Remark 3.4 a (relevant) definite formula, so $\forall_x R$ is also a definite formula. If $G \to R$, then it is a definite formula, since by hypothesis G is a goal formula and R a (relevant) definite formula.

   $I \to D$ is definite, because I is an irrelevant goal formula (by Remark 3.4) and D is a definite formula.

   "$\supset$" We show: $P^*, G^* \to D^*$ (with D^* relevant or G^* irrelevant), $\forall_x D^* \in \mathcal{D}$.

   Cases P^* and $\forall_x D^*$ are trivial.

   Case $G^* \to D^*$. If D^* is a relevant (definite) formula, then by the induction hypothesis $D^* \in \mathcal{R}_D$. Since by (IH) $G^* \in \mathcal{G}$, it follows by Remark 3.4 that $G^* \to D^* \in \mathcal{R}_D \subset \mathcal{D}$.

   If D^* is not relevant, then G^* must be irrelevant, so $G^* \in \mathcal{I}_G$. Since $D^* \in \mathcal{D}$ by (IH), it follows that $G^* \to D^* \in \mathcal{D}$.

2. $\bot, I, R \to G, D_0 \to G$ are *the* goal formulas. $\bot$ is clearly both in $\mathcal{G}$ and a goal formula.

   Case $I \in \mathcal{I}_G$. Since by (IH) $\mathcal{I}_G$ is the class of irrelevant goal formulas, I is a goal formula.

For $\rightarrow$, we have:

"$\subset$" The case $R \rightarrow G \in \mathcal{G}$ coincides by (IH) with $D^* \rightarrow G^*$, D relevant, so this is a goal formula.

Case $D_0 \rightarrow G$. If D_0 is decidable, then by Remark 3.5 $D_0 \rightarrow G$ is a goal formula.

"$\supset$" When $D^* \rightarrow G^*$ is a goal formula, then D^* is either relevant or quantifier-free definite formula. In the first situation, we either have $R \rightarrow G$, which is clearly in $\mathcal{G}$, or $R \rightarrow I$. For the latter, observe that by the (IH) and Remark 3.4, $I \in \mathcal{I}_G \subset \mathcal{G}$, so $R \rightarrow I \in \mathcal{G}$. If D^* is a quantifier-free definite formula, we are in the case $D_0 \rightarrow G \in \mathcal{G}$.

3. $\bot, \forall_x R, G \rightarrow R \in \mathcal{R}_D$ are the relevant definite formulas.

   By Remark 3.4, $\mathcal{R}_D \subset \mathcal{D}$ and since all above formulas "end" in $\bot$, they are relevant.

   For the reverse, if $G \rightarrow R$ is a relevant definite formula, then R must be relevant and by (IH) this implies that $R \in \mathcal{R}_D$. Since G is a goal formula, then by (IH) $G \in \mathcal{G}$. Thus, we are in the case $G \rightarrow R \in \mathcal{R}_D$.

4. $P, \forall_x I, D \rightarrow I \in \mathcal{I}_G$ are the irrelevant goal formulas. By Remark 3.4 and (IH) $\mathcal{I}_G \subset \mathcal{G}$, so by 2. such formulas are goal.

   Since P does not range over $\bot$, the formulas in $\mathcal{I}_G$ cannot "end" in bottom, so they are irrelevant by Definition 3.6.

□

3.3.2 Properties of the Definite/Goal Formulas

As already mentioned, the purpose of the restrictions introduced by Definition 3.5 is to allow substitution of $\bot$ by F. By this, the computationally relevant occurrences of negation are delimited from those to which no term should be associated in the extraction process. The following lemma identifies the relevant occurrences of $\bot$.

Notation 3.2. *Let A^F denote $A[\bot := F]$.*

Lemma 3.6. *Let $D \in \mathcal{D}$, $R \in \mathcal{R}_D$, $G \in \mathcal{G}$ and $I \in \mathcal{I}_G$, where the classes are given by Definition 3.5. Using $\mathsf{efq}_\bot$ and efq_P, the following formulas can be*

proved in minimal logic:

$$((R^F \to F) \to \bot) \to R \tag{3.1}$$
$$D^F \to D, \tag{3.2}$$
$$I \to I^F, \tag{3.3}$$
$$G \to (G^F \to \bot) \to \bot. \tag{3.4}$$

Proof. The proof is treated in detail in (Schwichtenberg and Wainer, 2011). However, since we will need it in order to illustrate how the case studies presented in Chapters 5-7 work, we replicate it here.

(3.1)–(3.4) are proved simultaneously by induction on formulas.

(3.1). *Case* $\bot$. Clearly $\tilde{\neg}\,\tilde{\neg}(\mathrm{F} \to \mathrm{F})$ is derivable.

Case $\forall_x R$.

$$\dfrac{(3.1)\colon \tilde{\neg}\neg R^{\mathrm{F}} \to R \qquad \dfrac{\dfrac{\tilde{\neg}\neg\forall_x R^{\mathrm{F}} \qquad \dfrac{\dfrac{\neg R^{\mathrm{F}} \qquad \forall_x R^{\mathrm{F}}}{\mathrm{F}}}{\neg\forall_x R^{\mathrm{F}}}}{\bot}}{\tilde{\neg}\neg R^{\mathrm{F}}}}{\dfrac{R}{\forall_x R}}$$
$$\tilde{\neg}\neg\forall_x R^{\mathrm{F}} \to \forall_x R$$

Case $G \to R$.

$$\dfrac{(3.1)\colon\ \tilde{\neg}\neg R^{\mathrm{F}} \to R \qquad \dfrac{\dfrac{(3.4)\colon\ G \to\ \tilde{\neg}\,\tilde{\neg} G^{\mathrm{F}}\ \ G}{\tilde{\neg}\,\tilde{\neg} G^{\mathrm{F}}} \qquad \dfrac{\dfrac{\tilde{\neg}\neg(G^{\mathrm{F}} \to R^{\mathrm{F}}) \qquad \dfrac{\dfrac{\neg R^{\mathrm{F}} \qquad \dfrac{G^{\mathrm{F}} \to R^{\mathrm{F}} \quad G^{\mathrm{F}}}{R^{\mathrm{F}}}}{\mathrm{F}}}{\neg(G^{\mathrm{F}} \to R^{\mathrm{F}})}}{\bot}}{\tilde{\neg} G^{\mathrm{F}}}}{\bot}}{\tilde{\neg}\neg R^{\mathrm{F}}}}{\dfrac{R}{\tilde{\neg}\neg(G^{\mathrm{F}} \to R^{\mathrm{F}}) \to G \to R}}$$

(3.2). *Case* $\bot$. Then $\bot^{\mathrm{F}} = \mathrm{F}$ and we have $\mathsf{efq}_\bot$.

Case P. $P^F = P$, so the case is trivial.

Case $\forall_x D$.

$$\dfrac{\dfrac{(3.2):\ D^{\mathrm{F}} \to D \quad \dfrac{\forall_x D^{\mathrm{F}}}{D^{\mathrm{F}}}}{\dfrac{D}{\forall_x D}}}{\forall_x D^{\mathrm{F}} \to \forall_x D}$$

Case R.

$$\dfrac{(3.1):\ \tilde{\neg}\neg R^{\mathrm{F}} \to R \qquad \dfrac{\mathrm{F} \to \bot \quad \dfrac{\neg R^{\mathrm{F}} \quad R^{\mathrm{F}}}{\mathrm{F}}}{\dfrac{\bot}{\tilde{\neg}\neg R^{\mathrm{F}}}}}{\dfrac{R}{R^{\mathrm{F}} \to R}}$$

Case $I \to D$.

$$\dfrac{\dfrac{(3.2):\ D^{\mathrm{F}} \to D \qquad \dfrac{I^{\mathrm{F}} \to D^{\mathrm{F}} \quad \dfrac{(3.3):\ I \to I^{\mathrm{F}} \quad I}{I^{\mathrm{F}}}}{D^{\mathrm{F}}}}{D}}{(I^{\mathrm{F}} \to D^{\mathrm{F}}) \to I \to D}$$

(3.3). *Case* P is trivial.
Case $\forall_x I$.

$$\dfrac{\dfrac{(3.3):\ I \to I^{\mathrm{F}} \quad \dfrac{\forall_x I}{I}}{\dfrac{I^{\mathrm{F}}}{\forall_x I^{\mathrm{F}}}}}{\forall_x I \to \forall_x I^{\mathrm{F}}}$$

Case $D \to I$.

$$\dfrac{\dfrac{(3.3):\ I \to I^{\mathrm{F}} \quad \dfrac{D \to I \quad \dfrac{(3.2):\ D^{\mathrm{F}} \to D \quad D^{\mathrm{F}}}{D}}{I}}{I^{\mathrm{F}}}}{(D \to I) \to D^{\mathrm{F}} \to I^{\mathrm{F}}}$$

(3.4). *Case* $\bot$. Follows immediately.
Case I.

$$\dfrac{\dfrac{(3.3):\ I \to I^{\mathrm{F}} \quad I}{I^{\mathrm{F}}} \qquad I^{F} \to \bot}{\bot}$$

$$I \to (I^{\mathrm{F}} \to \bot) \to \bot$$

Case $R \to G$.

$$\dfrac{\dfrac{\text{(3.4)}\colon\ G \to\ \tilde{\neg}\,\tilde{\neg} G^{\mathrm{F}} \quad \dfrac{R \to G \quad R}{G}}{\tilde{\neg}\,\tilde{\neg} G^{\mathrm{F}}} \quad \dfrac{\tilde{\neg}(R^{\mathrm{F}} \to G^{\mathrm{F}}) \quad \dfrac{G^{\mathrm{F}}}{R^{\mathrm{F}} \to G^{\mathrm{F}}}}{\dfrac{\bot}{\tilde{\neg} G^{\mathrm{F}}}}}{\dfrac{\bot}{\tilde{\neg} R\ (1)}}$$

$$\dfrac{\tilde{\neg}(R^{\mathrm{F}} \to G^{\mathrm{F}}) \quad \dfrac{\dfrac{\dfrac{\neg R^{\mathrm{F}} \quad R^{\mathrm{F}}}{\mathrm{F}}\,\mathsf{efq}_{G^F}}{G^{\mathrm{F}}}}{R^{\mathrm{F}} \to G^{\mathrm{F}}}}{\dfrac{\bot}{\tilde{\neg}\neg R^{\mathrm{F}}\ (2)}}$$

Further,

$$\dfrac{\text{(1)}\colon\ \tilde{\neg} R \quad \dfrac{\text{(3.1)}\colon\ \tilde{\neg}\neg R^{\mathrm{F}} \to R \quad \text{(2)}\colon\ \tilde{\neg}\neg R^{\mathrm{F}}}{R}}{\dfrac{\bot}{(R \to G) \to\ \tilde{\neg}\,\tilde{\neg}(R^{\mathrm{F}} \to G^{\mathrm{F}})}}$$

Case $D_0 \to G$. $\tilde{\neg} D_0$ and $\tilde{\neg}\neg D_0^{\mathrm{F}}$ can be derived as (1) and (2) above. In addition, we have:

$$\dfrac{\text{(1)}\colon\ \tilde{\neg} D_0 \quad \dfrac{\text{(3.2)}\colon\ D_0^{\mathrm{F}} \to D_0 \quad D_0^{\mathrm{F}}}{D_0}}{\dfrac{\bot}{\tilde{\neg} D_0^{\mathrm{F}}(3)}}$$

If we now apply *case distinction* on $D_0^{\mathrm{F}}\colon (D_0^{\mathrm{F}} \to \bot) \to (\neg D_0^{\mathrm{F}} \to \bot) \to \bot$ to (2) and (3), we can prove our claim. □

We point out the importance of the last step in the above proof. Because (Cases) is necessary, we need by Lemma 3.4 that D_0 is decidable. Whenever the proof transformation amounts to using this step, the extracted terms contain the case distinction operator.

Lemma 3.6 provides a mechanism by which some of the $\bot$-occurrences can be safely replaced by F, using the derivation (3.2) in the case of definite formulas and the proof of (3.4) for the goal formulas. For reasons that should be clear from the above proof, these substitutions can only be performed

when the formulas are definite and goal, respectively. Moreover, by carefully analyzing the above proof steps, we observe that for (3.2) and (3.4) we need (3.1) and (3.3). This justifies the further restrictions (relevant and irrelevant, respectively) on the $\mathcal{D}$ and $\mathcal{G}$ classes.

3.3.3 Refined A-Translation

We consider proofs of Π^0_2-formulas, to which we have applied Lemma 3.6. Having kept $\bot$ only in the necessary positions, we can now substitute it by $A := \exists_y G(x, y)$, as in the original A-Translation method. Figure 3.5 illustrates this process, which we summarize in the following lemma.

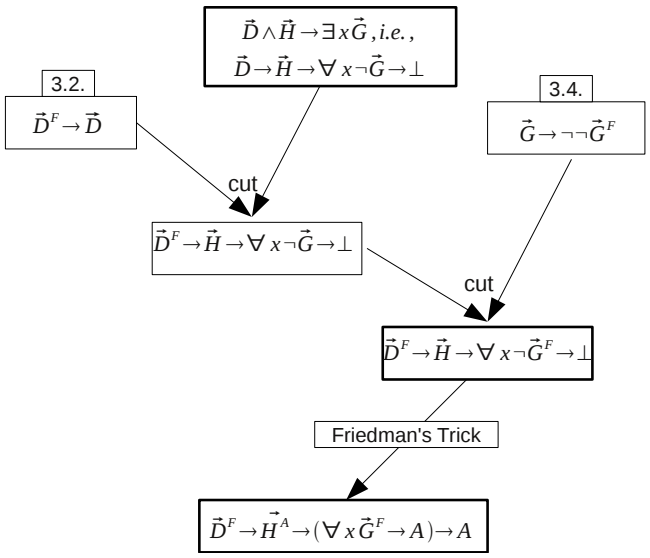

Figure 3.5: Refined A-Translation: substituting $\bot$ by F.

Lemma 3.7. *Consider that we are given a non-constructive proof of*

$$\vec{D} \to \vec{H} \to \forall_x \tilde{\exists}_y \, G(x, y), \tag{3.5}$$

where $G \in \mathcal{G}$ and the assumptions are either definite ($\vec{D} \in \mathcal{D}$) or arbitrary ($\vec{H}$) formulas. Then

$$\vdash_i \forall_x. \, \vec{D}^F \to \vec{H}[\bot := \exists_y \, \vec{G}^F(x, y)] \to \exists_y \, \vec{G}^F(x, y). \tag{3.6}$$

Proof Sketch. (3.5) can be seen as

$$\forall_x. \, \vec{D} \to \vec{H} \to \forall_y (\vec{G}(x, y) \to \bot) \to \bot \tag{3.7}$$

Assume x. According to the Lemma 3.6, (3.7) can be transformed to

$$\vdash_i (F \to \bot) \to \vec{D}^F \to \vec{H} \to \forall_y(\vec{G}^F(x,y) \to \bot) \to \bot.$$

We substitute in this latter $\bot$ by $\exists_y \vec{G}^F(x,y)$. Since $F \to \exists_y \vec{G}^F(x,y)$ holds by efq and $\forall_y. \vec{G}^F(x,y) \to \exists_y \vec{G}^F(x,y)$ by $\exists^{+}$[4], we obtain

$$\vdash_i \vec{D}^F \to \vec{H}[\bot := \exists_y \vec{G}^F(x,y)] \to \exists_y \vec{G}^F(x,y). \tag{3.8}$$

□

3.4 Program Extraction

We give in what follows the mechanism necessary in order to associate programs to proofs. We first introduce the notion of modified realizability and then give the rules by which we recursively compute the extracted terms. The soundness theorem will guarantee that the extracted terms are indeed realizers for the proof terms. We show further how this procedure can be applied to the A-Translated proofs.

3.4.1 Modified Realizability. Extracted Terms

Since we are interested in the computational content of proofs, we need to carefully distinguish between computationally relevant and irrelevant formulas. The latter class consists of formulas A whose proofs M have no computational content and has been identified in (Troelstra and Schwichtenberg, 2000) with the *Harrop formulas*.

Definition 3.7 (Harrop Formulas)**.** *Computationally irrelevant (c.i.) or Harrop formulas belong to the class* $\mathcal{H}$[5]

$$P, A \to H, \forall_x H \in \mathcal{H},$$

where P *ranges over prime formulas and* $H \in \mathcal{H}$*.*

Formulas which are not Harrop are considered to be computationally relevant (c.r.).

Remark 3.6. *The irrelevant goal formulas* $\mathcal{I}_G$ *are Harrop formulas.*

[4] $\exists^{+}$ corresponds to the introduction axiom for the strong existential quantifier.

[5] Recall from Remark 3.3 that we do not need to treat conjunction explicitely.

Notation 3.3. *We assign to every formula a computational type $\tau(A)$ and denote by $[\![M]\!]$ the program extracted from the proof of A.*

In the special case when A is computationally irrelevant (c.i.), we assign to it the type ε and let $[\![M]\!] := \epsilon$. Clearly, if $\tau(A) \neq \varepsilon$, then A is a computationally *relevant* formula.

Definition 3.8 (Computational type)**.** *Let P denote a predicate (variable or constant). In case of a predicate variable, since we do not a priori know what comprehension term will be substituted for it, we assign to P some generic type variable α_P. With the conventions made in* (ε red.) *on page 11, we consider the following typing rules*

$$\begin{aligned}
\tau(P(\vec{r})) &:= \begin{cases} \varepsilon & \text{if } P \text{ does not have content} \\ \alpha_P & \text{otherwise,} \end{cases} \\
\tau(A \to B) &:= \tau(A) \Rightarrow \tau(B), \\
\tau(\forall_{x^\rho} A) &:= \rho \Rightarrow \tau(A), \\
\tau(\exists_{x^\rho} A) &:= \rho \times \tau(A).
\end{aligned}$$

As pointed out in Chapter 1, we view the formulas $\forall_x \tilde{\exists}_y G(x,y)$ as specifications of the form: "Given an input x and a requirement G on x, is there an algorithm which produces from x an output y, such that G is met?" The extracted program represents in this context a computational solution to the problem of producing the data with the specified property. In the following, we overview the rules by which such a solution is constructed.

Notation 3.4. *Let $r\,\mathbf{mr}\,A$ denote* (Kreisel's) modified realizability*.*

$r\,\mathbf{mr}\,A$ reads as "the term r realizes the formula A" and represents the answer to the aforementioned question. The following definitions give a method to recursively construct the modified realizer of a formula.

Definition 3.9 (Modified realizability)**.** *Let P be a predicate: if P is a predicate variable of signature $\vec{\sigma}$ and type α_P, let $P^{\mathbf{r}}$ be a new predicate variable of signature $(\tau(\mathbf{r}), \vec{\sigma})$ associated to P. $r\,\mathbf{mr}\,A$ is given inductively for A by:*

$$\begin{aligned}
r\,\mathbf{mr}\,P(\vec{s}) &:= \begin{cases} P^{\mathbf{r}}(r,\vec{s}) & \text{if } P \text{ is a predicate variable} \\ P(\vec{s}) & \text{if } P \text{ is a predicate constant} \end{cases} \\
r\,\mathbf{mr}\,\forall_x A &:= \forall_x\, rx\,\mathbf{mr}\,A \\
r\,\mathbf{mr}\,A \to B &:= \forall_x.\, x\,\mathbf{mr}\,A \;\to\; rx\,\mathbf{mr}\,B \\
r\,\mathbf{mr}\,\exists_x A(x) &:= r0\,\mathbf{mr}\,A(r1)
\end{aligned}$$

If $\tau(A) = \varepsilon$, *then* $r\,\mathbf{mr}\,\exists_x A(x) := A(r)$.

Definition 3.10 (Extracted term)**.** *The extracted term of a derivation is given by the following rules:*

$$\begin{aligned}
[\![u^A]\!] &:= x_u^{\tau(A)} \quad (x_u \text{ uniquely associated with } u),\\
[\![(\lambda u^A M^B)^{A\to B}]\!] &:= \lambda x_u^{\tau(A)} [\![M]\!]^{\tau(B)},\\
[\![M^{A\to B} N^A]\!] &:= [\![M]\!]^{\tau(A\to B)} [\![N]\!]^{\tau(A)},\\
[\![(\lambda x^\rho M^A)^{\forall_x A}]\!] &:= \lambda x^\rho [\![M]\!]^{\tau(A)},\\
[\![M^{\forall_x A} r]\!] &:= [\![M]\!]^{\tau(\forall_x A)} r.
\end{aligned}$$

In particular, when $\sigma := \tau(A) \neq \varepsilon$, the following recursion operators are associated to the induction schemes

$$[\![\mathsf{Ind}_{b,A}]\!] := \mathcal{R}^\sigma_{\mathtt{B}}, \qquad [\![\mathsf{Ind}_{n,A}]\!] := \mathcal{R}^\sigma_{\mathtt{N}}, \qquad [\![\mathsf{Ind}_{L,A}]\!] := \mathcal{R}^\sigma_{\mathtt{L}(\rho)}$$

with the typing and conversion rules from Section 2.1.

The Cases extracts to $\mathcal{C}$,

$$[\![\mathsf{Cases}_{n,A}]\!] := \mathcal{C}^\sigma_{\mathtt{N}}, \qquad [\![\mathsf{Cases}_{l,A}]\!] := \mathcal{C}^\sigma_{\mathtt{L},(\rho)}$$

each of which, according to the conversion rules from Section 2.1, corresponds to an "if ... then ... else" construct.

Since the boolean induction corresponds to proof by cases, one can view the boolean recursion operator as a cases operator

$$\mathcal{R}_{\mathtt{B}}(b^{\mathtt{B}}, f, g) := \mathcal{C}^\sigma_{\mathtt{B}}(b, f, g) = \text{if } b \text{ then } f \text{ else } g$$

The soundness is guaranteed by the following theorem:

Theorem 3.4 (Soundness, Berger et al. (2002))**.** *Let* M *be a derivation of* B *and* $FA(M)$ *collect its free assumptions. Then there exists a derivation of* $[\![M]\!]\,\mathbf{mr}\,B$ *from the assumptions* $\{x_u^{\tau(C)}\,\mathbf{mr}\,C | u^C \in FA(M)\}$.

Proof. The proof, by induction on the logical rules of M, is given in detail in (Schwichtenberg and Wainer, 2011). In the case of axioms, the realizers are given explicitely and are shown to be correct. □

3.4.2 Program Extraction from A-Translated Proofs

In this section, we apply the extraction mechanism to the proofs transformed by the refined A-Translation. We follow the exposition in (Schwichtenberg and Wainer, 2011). In order to determine the realizer for the A-translated formula (3.8), we first need to construct the realizers for the proofs inserted when replacing $\bot$ by F - these can be obtained from Lemma 3.6.

Definition 3.11. *A formula A is considered to be* invariant *when*

$$\varepsilon\,\mathbf{mr}\,A = A.$$

Lemma 3.8. *Under the assumption that all prime formulas P in D^F and G^F are computationally irrelevant (c.i.) and invariant, we can construct from the proofs of* (3.2) *and* (3.4) *terms t_D and s_G such that*

(a) $D^F \to t_D\,\mathbf{mr}\,D$ *and*

(b) $\forall_{v,w}.\ v\,\mathbf{mr}\,G \to (G^F \to w\,\mathbf{mr}\,\bot) \to s_G\,w\,v\,\mathbf{mr}\,\bot$

are derivable from $\forall_y(\mathrm{F} \to y\,\mathbf{mr}\,\bot)$ and efq_P.

Proof. Consider the formulas (3.2) and (3.4) from Lemma 3.6. Let N_D be the proof of $D^F \to D$ and N_G the derivation for $G \to (G^F \to \bot) \to \bot$.

By the Soundness Theorem we have $t_D = [\![N_D]\!]$ such that by Definition 3.9 and since D^F is c.i.:

$$[\![N_D]\!]\,\mathbf{mr}\,D^F \to D \Leftrightarrow D^F \to t_D\,\mathbf{mr}\,D$$

The Soundness Theorem gives us also $s_G = [\![N_G]\!]$, which is inferred using Definition 3.9, as follows:

$$\begin{aligned}
&[\![N_G]\!]\,\mathbf{mr}\,((G^F \to \bot) \to G \to \bot) \Leftrightarrow\\
&\forall_w.\ w\,\mathbf{mr}\,(G^F \to \bot) \to [\![N_G]\!]\,w\,\mathbf{mr}\,(G \to \bot) \Leftrightarrow\\
&\forall_{v,w}.\ w\,\mathbf{mr}\,(G^F \to \bot) \to v\,\mathbf{mr}\,G \to [\![N_G]\!]\,w\,v\,\mathbf{mr}\,\bot
\end{aligned}$$

Since G^F is c.i., we obtain:

$$\forall_{v,w}.\ (G^F \to w\,\mathbf{mr}\,\bot) \to v\,\mathbf{mr}\,G \to s_G\,w\,v\,\mathbf{mr}\,\bot$$

and this is equivalent to what we want to show. □

In addition, we need *external realizers* for the non-definite assumptions $\vec{H}$, i.e., terms $\vec{r}$ s.t.

$$\vdash \vec{H}^{\exists_y G^F(y)} \to \vec{r}\,\mathbf{mr}\,\vec{H}^{\exists_y G^F(y)}. \tag{3.9}$$

Let in the following $\tilde{M}$ be the initial derivation for $\vec{D} \to \vec{H} \to \tilde{\exists}_y G(y)$.

Remark 3.7. *By Definition 3.9* $t\,\mathbf{mr}\,\bot = \bot^{\mathbf{r}}(t)$. *Since* G^F *is c.i. and invariant we therefore have:*

$$(t\,\mathbf{mr}\,\bot)[\bot^{\mathbf{r}}/G^F(\cdot)] = G^F(t) = t\,\mathbf{mr}\,\exists y G^F(y). \tag{3.10}$$

By structural induction on formulas we can thus construct for all H *in which* $\bot^{\mathbf{r}}$ *does not occur:*

$$(t\,\mathbf{mr}\,H)[\bot^{\mathbf{r}}/G^F(\cdot)] = t\,\mathbf{mr}\,H^{\exists y G^F(y)}. \tag{3.11}$$

Theorem 3.5. *Assume that all prime formulas* P *in* $\vec{D}^F$ *and* G^F *are computationally irrelevant and invariant. Let* t_D *and* s_G *be the realizers provided by Lemma 3.8 and let* $\vec{r}\,\mathbf{mr}\,\vec{H}^{\exists_y G^F(y)}$ *be given by* (3.9).

Then, by the Soundness Theorem for realizability we can derive

$$\vec{D}^F \to \vec{H}^{\exists_y G^F(y)} \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,s_G\,\mathbf{mr}\,\exists_y G^F.$$

Proof. By the Soundness Theorem,

$$[\![\tilde{M}]\!]\,\mathbf{mr}\,(\vec{D} \to \vec{H} \to (\forall_y.\ G(y) \to \bot) \to \bot).$$

Using Definition 3.9 of modified realizability and the realizing terms that we have from the hypothesis, we infer:

$$\begin{aligned}
&\vec{t_D}\,\mathbf{mr}\,\vec{D} \to [\![\tilde{M}]\!]\vec{t_D}\,\mathbf{mr}\,(\vec{H} \to \tilde{\exists}_y G(y)) \Leftrightarrow\\
&\vec{t_D}\,\mathbf{mr}\,\vec{D} \to \vec{r}\,\mathbf{mr}\,\vec{H} \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,\mathbf{mr}\,\tilde{\exists}_y G(y) \Leftrightarrow\\
&\vec{t_D}\,\mathbf{mr}\,\vec{D} \to \vec{r}\,\mathbf{mr}\,\vec{H} \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,\mathbf{mr}\,((\forall_y.\ G(y) \to \bot) \to \bot) \Leftrightarrow\\
&\vec{t_D}\,\mathbf{mr}\,\vec{D} \to \vec{r}\,\mathbf{mr}\,\vec{H} \to \forall_s.\ s\,\mathbf{mr}\,(\forall_y.\ G(y) \to \bot) \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,s\,\mathbf{mr}\,\bot \Leftrightarrow\\
&\vec{t_D}\,\mathbf{mr}\,\vec{D} \to \vec{r}\,\mathbf{mr}\,\vec{H} \to \forall_s.\ \forall_y s\,y\,\mathbf{mr}\,(G(y) \to \bot) \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,s\,\mathbf{mr}\,\bot \Leftrightarrow\\
&\vec{t_D}\,\mathbf{mr}\,\vec{D} \to \vec{r}\,\mathbf{mr}\,\vec{H} \to \forall_s.\ \forall_{y,t}(t\,\mathbf{mr}\,G(y) \to s\,y\,t\,\mathbf{mr}\,\bot) \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,s\,\mathbf{mr}\,\bot
\end{aligned}$$

So, by Lemma 3.8a,

$$\vec{D}^F \to \vec{r}\,\mathbf{mr}\,\vec{H} \to \forall_s.\ \forall_{y,t}(t\,\mathbf{mr}\,G(y) \to s\,y\,t\,\mathbf{mr}\,\bot) \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,s\,\mathbf{mr}\,\bot \quad (\circ)$$

If we take $s := s_G$ in $(\circ)$ and use (3.9) we obtain by Remark 3.7:

$$\begin{array}{l}\vec{D}^F \to \vec{H}^{\exists_y G^F(y)} \to \forall_{y,t}(t\,\mathbf{mr}\,G(y)^{\exists_y G^F(y)} \to s_G\,y\,t\,\mathbf{mr}\,\exists_y G^F(y)) \to \\ \quad \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,s_G\,\mathbf{mr}\,\exists_y G^F(y)\end{array} \qquad (*)$$

Let us substitute $\bot^{\mathbf{r}}$ by $G^F(\cdot)$ in Lemma 3.8b. We get:

$$\forall_{v,w}.\,(G^F(y) \to w\,\mathbf{mr}\,\exists_y G^F(y)) \to v\,\mathbf{mr}\,G(y)^{\exists_y G^F(y)} \to s_G\,w\,v\,\mathbf{mr}\,\exists_y G^F(y)$$

or, equivalenty (see (3.10))

$$\forall_{v,w}.\,(G^F(y) \to G^F(w)) \to v\,\mathbf{mr}\,G(y)^{\exists_y G^F(y)} \to G^F(s_G\,w\,v)$$

For $w := y$ and $v := t$ we obtain: $t\,\mathbf{mr}\,G(y)^{\exists_y G^F(y)} \to G^F(s_G\,y\,t)$. and from (3.10) we have $G^F(s_G\,y\,t) = s_G\,y\,t\,\mathbf{mr}\,G(y)^{\exists_y G^F(y)}$. Hence, $(*)$ becomes:

$$\vec{D}^F \to \vec{H}^{\exists_y G^F(y)} \to [\![\tilde{M}]\!]\vec{t_D}\,\vec{r}\,s_G\,\mathbf{mr}\,\exists_y G^F(y).$$

□

3.5 Related Work

We carry out in this section a thorough comparison between the definite and goal formulas as introduced in Section 3.3.1 and similar classes introduced in Ishihara (2000), with a two-folded purpose. On the one hand, we aim at extending the former and on the other, we want to better understand their structure. The first question has, however, a negative answer: it is not possible to extend the definite/goal classes by formulas in $\mathcal{Q}, \mathcal{J}$ and $\mathcal{S}$ from Ishihara (2000), while guaranteeing the properties in Lemma 3.6. The reason lies in the fact that Ishihara is only concerned with defining classes of formulas for which one can infer intuitionistic validity from the classical one, but not with restricting the logic to be minimal, in the sense of eliminating $\mathsf{efq}^{\bot}$. Hence, for the formulas additional allowed in the classes $\mathcal{Q}, \mathcal{J}$ and $\mathcal{S}$ the transformations in Lemma 3.6 are not possible.

Ishihara (2000) does not envisage realizability, so no distinction between week and strong operators is present. We will however restrict our attention in what follows to the basic operators fixed in Chapter 2 and limit the definitions of $\mathcal{Q}, \mathcal{J}, \mathcal{S}$ below to the NA^{ω}-fragment.

Definition 3.12. *(Ishihara (2000)) Let P $(\neq \bot)$ range over prime formulas and Q, J, S range over $\mathcal{Q}, \mathcal{J}, \mathcal{S}$, respectively.*

1. *Let $\mathcal{Q}, \mathcal{S}, \mathcal{J}$ be defined by the following inductive clauses*

$$\begin{aligned} &\bot, P, \forall_x Q, J \to Q \ \in \ \mathcal{Q}, \\ &\bot, P, S \to J \ \in \ \mathcal{J} \\ &\bot, \forall_x S, J \to S \ \in \ \mathcal{S}. \end{aligned}$$

2. *For $K, K' \in \mathcal{K}$, the class $\mathcal{K}$ is generated inductively by*

$$J, \forall_x K, Q \to K \ \in \ \mathcal{K}.$$

We first observe that there is a similarity between the above-defined classes and the definite/goal formulas, which can be paired as follows: $\mathcal{Q}$ and $\mathcal{D}$, $\mathcal{J}$ and $\mathcal{G}$, $\mathcal{S}$ and $\mathcal{R}_D$. We give a detailed comparison below.

The following remark is an easy exercise, using induction on formulas.

Remark 3.8. *$\mathcal{S} \subsetneq \mathcal{Q}$ and $\mathcal{J} \subsetneq \mathcal{K}$.*

Notation 3.5. *We denote by $\sqcup$ the disjoint union of sets.*

Convention. Throughout this section, P is used to denote a prime formula distinct from $\bot$. $\mathcal{Q}, \mathcal{S}, \mathcal{J}$ are given by Definition 3.12 and $\mathcal{D}, \mathcal{R}_D, \mathcal{G}$ and $\mathcal{I}_G$ by Definition 3.6. We take $S \in \mathcal{S}$, $D \in \mathcal{D}$, $G \in \mathcal{G}, R \in \mathcal{R}_D, J \in \mathcal{J}$ and $I \in \mathcal{I}_G$.

Lemma 3.9. *We define the set collecting all irrelevant formulas in $\mathcal{J}$:*

$$\mathcal{I}_J := \{\, \vec{S} \to P \mid S \in \mathcal{S}, P \neq \bot \,\}., \tag{$\mathcal{I}_J$}$$

1. *We have*

$$\mathcal{I}_G \cap \mathcal{J} = \mathcal{I}_J. \tag{I_J}$$

Furthermore,

   (a) $\mathcal{I}_G \setminus \mathcal{J} = \{\forall_x I, D \to I, \text{ for } D \notin \mathcal{S} \text{ or } I \notin \mathcal{I}_J\}$
   (b) $\mathcal{J} \setminus \mathcal{I}_G = \{\, \vec{S} \to \bot \mid S \in \mathcal{S} \,\}$.

2. *$\mathcal{J} \subsetneq \mathcal{G}$. Moreover, with D_0 decidable formulas ranging over the $\mathcal{D}$,*

$$\mathcal{G} = \mathcal{J} \sqcup (\mathcal{I}_G \setminus \mathcal{J}) \sqcup \{\, D_0 \to G \mid D_0 \notin \mathcal{S} \text{ or } G \notin \mathcal{J} \cup \mathcal{I}_G \,\}. \tag{3.12}$$

3. *$\mathcal{S} \subsetneq \mathcal{R}_D$. More precisely,*

$$\mathcal{R}_D = \mathcal{S} \sqcup \{\, G \to R \mid G \notin \mathcal{J} \text{ or } R \notin \mathcal{S} \,\}. \tag{3.13}$$

Proof. We prove the claims by simultaneous induction.

1. By Remark 3.4 it is immediate to see that $\mathcal{I}_J \subset \mathcal{I}_G \cap \mathcal{J}$. The converse follows from (1a), (1b) and the definitions of $\mathcal{J}$ and $\mathcal{I}_G$.

   (a) Let $I \in \mathcal{I}_G$.

   If I is $\forall_x I$, then $I \notin \mathcal{J}$, since $\mathcal{J}$ does not allow outer universal quantification.

   If we have $D \to I$ then this is by definition not in $\mathcal{J}$ if $D \notin \mathcal{S}$, which is possible by 3. and Remark 3.10.

   If we have $D \to I$, but $I \notin \mathcal{J}$, then $D \to I$ is clearly not in $\mathcal{J}$. This is for instance the case with the formulas $D \to \forall_x I \in \mathcal{I}_G$.

   All other formulas in $\mathcal{I}_G$ (P, $D \to I$ with $D \in \mathcal{S}, I \in \mathcal{I}_J$) are collected in $\mathcal{I}_J$.

   (b) All formulas "ending" in $\bot$ from $\mathcal{J}$ are clearly not in $\mathcal{I}_G$. The other formulas from $\mathcal{J}$, i.e., formulas of the kind $S \to J$, with $S \in \mathcal{S}$ and $J \in \mathcal{I}_J$ are by Remark 3.4 and by 3. in $\mathcal{I}_J$.

(12) We first show that $\mathcal{J} \subset \mathcal{G}$.

   Let $J \in \mathcal{J}$.

   Cases $\bot$ and P. Then clearly, $J \in \mathcal{G}$.

   Case $S \to J$. Follows by the (IH) from $S \in \mathcal{S} \subset \mathcal{R}_D$ (3.) and $J \in \mathcal{J} \subset \mathcal{G}$ (2.)

   $\mathcal{J} \neq \mathcal{G}$ is an immediate consequence of (1a) and of Remark 3.4.

   To show (3.12), let $G \in \mathcal{G}$.

   If G is $\bot$ or P, it is already in $\mathcal{J}$.

   If $\forall_x I$ or if $D \to I$, with $D \in \mathcal{D} \setminus \mathcal{S}$ or $I \in \mathcal{I}_G \setminus \mathcal{J}$ (i.e., $I \notin \mathcal{I}_J$), the formulas are in $\mathcal{G} \setminus \mathcal{J}$ and by (1a), also in $\mathcal{I}_G \setminus \mathcal{J}$.

   In the cases $R \to G$ with $G \in \mathcal{G} \setminus \mathcal{J}$ and $D_0 \to G$ with $D_0 \notin \mathcal{S}$ or $G \in \mathcal{G} \setminus \mathcal{J}$, we are again outside $\mathcal{J}$. The former is in $\mathcal{I}_G \setminus \mathcal{J}$ and so is $D_0 \to G$, for $G \in \mathcal{G}$.

   "$\supset$" is trivial by the definition of $\mathcal{J}$ and 1a,.

3. We first show $\mathcal{S} \subset \mathcal{R}_D$. Let $S \in \mathcal{S}$.

   Case $\bot$ is clear and case $\forall_x S$ follows by the induction hypothesis.

   Case $J \to S$ follows also by the induction hypothesis, using 2.

For $\mathcal{R}_D \neq \mathcal{S}$ and (3.13), by 2 we have $\mathcal{J} \neq \mathcal{G}$. Therefore, if $G \in \mathcal{G} \setminus \mathcal{J}$, then $G \to R \notin \mathcal{S}$. By (IH), there is also some $R \notin \mathcal{S}$ for which $G \to R \notin \mathcal{S}$.

"$\supset$" is trivial.

□

Remark 3.9. *Since $\mathcal{I}_G \not\subset \mathcal{J}$, this means that $\mathcal{J}$ does not capture all irrelevant goal formulas.*

Remark 3.10. $\mathcal{R}_D \neq \mathcal{Q}$

Proof. We only regard the formulas $G \to R$. As shown in (3.12), $\mathcal{G} \supsetneq \mathcal{J}$, so we can have $G \in \mathcal{G}$, but $G \notin \mathcal{J}$. In such cases, $G \to R \notin \mathcal{Q}$. □

Let us analyze the connection between $\mathcal{D}$ and $\mathcal{Q}$. As we will see in the following, neither $\mathcal{D} \subset \mathcal{Q}$, nor $\mathcal{Q} \subset \mathcal{D}$.

Lemma 3.10. *The following relations exist between $\mathcal{D}$ and $\mathcal{Q}$:*

1. $\mathcal{Q} \setminus \mathcal{D} = \{ J \to Q \mid (J \notin \mathcal{I}_J \wedge Q \notin \mathcal{R}_D)$ *or* $Q \notin \mathcal{D} \} \cup \{ \forall_x Q \mid Q \notin \mathcal{D} \}$
2. $\mathcal{D} \setminus \mathcal{Q} = \{ G \to R \mid G \notin \mathcal{J}$ *or* $R \notin \mathcal{Q} \} \cup \{ I \to D \mid I \notin \mathcal{J}$ *or* $D \notin Q \}$ $\cup \{ \forall_x D \mid D \notin \mathcal{Q} \}$
3. *Let $I_J \in \mathcal{I}_J$ be given by $(\mathcal{I}_J)$. We inductively define the class $\bar{\mathcal{D}}$ as*

$$P, \forall_x \bar{D}, S, I_J \to \bar{D}, G \to R \in \bar{\mathcal{D}},$$

where $\bar{D} \in \bar{\mathcal{D}}$ and $G \in \mathcal{J}, R \in \bar{\mathcal{D}} \cap \mathcal{R}_D$.

Then $\bar{\mathcal{D}} := \mathcal{D} \cap \mathcal{Q}$.

Proof. We use Lemma 3.9 to show (1) and regard only the subformulas which reside in the complement of each class with respect to $\bar{\mathcal{D}}$.

1. $\mathcal{Q} \not\subset \mathcal{D}$. Case $\bot$. This is in $\mathcal{R}_D$, therefore also in $\mathcal{D}$.

   Case P. This is in $\mathcal{D}$ as well.

   Case $\forall_x Q$. By the (IH) there is some $\mathcal{Q} \notin \mathcal{D}$ for which $\forall_x Q \notin \mathcal{D}$.

   Case $J \to Q$. Since $\mathcal{J}$ contains also relevant formulas (for instance $\bot$), in these cases J is not in $\mathcal{I}_G$, so $J \to Q \notin \mathcal{D}$ when $J \notin \mathcal{I}_J$ and $Q \in \mathcal{R}_D$. Otherwise, if $Q \in \mathcal{R}_D$, then $G \to R \in \mathcal{R}_D \subset \mathcal{D}$. Clearly, if $D \in \mathcal{D} \setminus \mathcal{Q}$, then $J \to D \notin \mathcal{Q}$.

2. $\mathcal{D} \not\subset \mathcal{Q}$. Case P. This is both in $\mathcal{D}$ and $\mathcal{Q}$.

   Case $\forall_x D$. As above, for $D \in \mathcal{D} \notin \mathcal{Q}$ it follows that $\forall_x D \notin \mathcal{Q}$.

   Case R.

   - $\bot \in \mathcal{Q}$.
   - Case $\forall_x R$. Since $\mathcal{R}_D \neq \mathcal{Q}$ (Remark 3.10), it follows that for $R \notin \mathcal{Q}$ also $\forall_x R \notin \mathcal{Q}$.
   - Case $G \to R$. As shown for Remark 3.10 (3.12), for $G \in \mathcal{G} \setminus \mathcal{J}$, we have that $G \to R \notin \mathcal{Q}$.

   Case $I \to D$. By (1a), there exist some $I \notin \mathcal{J}$, so in such cases $I \to D \notin \mathcal{Q}$. Likewise, if $D \in \mathcal{Q} \setminus \mathcal{D}$.

3. Recall that $\mathcal{I}_J$ are the irrelevant $\mathcal{J}$-formulas and $\mathcal{I}_G \cap \mathcal{J} = \mathcal{I}_J$ (Lemma 3.9).

   Let $\bar{D} \in \bar{\mathcal{D}}$. We show that $\bar{D}$ is in both $\mathcal{D}$ and $\mathcal{Q}$..

   Cases $\bot$ and P are trivial. The case $\forall_x \bar{D}$ follows by the (IH).

   Case S follows by Lemma 3.9, (3.13) and Remark 3.4 for $\mathcal{D}$ and by Remark 3.10 for $\mathcal{Q}$.

   Case $I_J \to \bar{D}$. By (IH), $\bar{D} \in \mathcal{D} \cap \mathcal{Q}$ and by (I_J) $I_J \in \mathcal{I}_G$ and $I_J \in \mathcal{J}$. Then, by the corresponding definitions, $I_J \to \bar{D} \in \mathcal{D} \cap \mathcal{Q}$.

   For the converse, notice that $\bar{\mathcal{D}}$ contains all the clauses defining $\mathcal{D}$ and $\mathcal{Q}$, with the exception of those excluded by (2) and (1).

□

From the above proof, it follows that:

Corollary 3.5.1. *1.* $\bar{\mathcal{D}} = \mathcal{S} \cup \{P, \forall_x \bar{D}, I_J \to \bar{D}\}$.

2. $\mathcal{R}_D \setminus \bar{\mathcal{D}} = \{\, G \to R \mid G \notin \mathcal{I}_J \text{ or } R \notin \bar{\mathcal{D}} \,\} \cup \{\, \forall_x R \mid R \notin \bar{\mathcal{D}} \,\}$.

$\mathcal{K}$ seems similar in purpose to $\mathcal{G}$. However, Lemmas 3.9 and 3.10 entitle us to formulate right away the following:

Lemma 3.11. *Neither* $\mathcal{G} \subset \mathcal{K}$, *nor* $\mathcal{G} \supset \mathcal{K}$.

Proof Sketch. Clearly, if $Q \in \mathcal{Q} \setminus \bar{\mathcal{D}}$, then $Q \to K \notin \mathcal{G}$.
Likewise, when $R \to G \in \mathcal{G}$, if $R \in \mathcal{R}_D \setminus \bar{\mathcal{D}}$, then $R \to G \notin \mathcal{K}$. □

Summary. From the above analysis we draw the following *conclusions*:

- neither $\mathcal{Q}$, nor $\mathcal{D}$ are an extension of the other, but they have in common a superclass of $\mathcal{S}$. Some, but not all relevant definite formulas $\mathcal{D}$ lie in this common class.
- $\mathcal{G}$ is richer than $\mathcal{J}$. However, since $\mathcal{J}$ does not separate the irrelevant formulas from the relevant ones, no inclusion-relation can be established with $\mathcal{I}_G$.

We summarize these conclusions below and depict them in Figure 3.6.

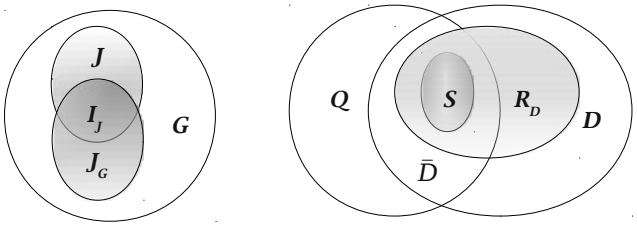

Figure 3.6: Relation among the classes definite-goal and $\mathcal{Q}, \mathcal{J}, \mathcal{S}$

Proposition 3.2 (Classification)**.** *The following relations can be identified among the definite/goal formulas and the classes defined in Ishihara (2000):*

1. *Between $\mathcal{S}, \mathcal{R}_D, \mathcal{D}$ and $\mathcal{Q}$ we have the following relations*
   - $\mathcal{S} \subset \mathcal{Q}$ *(Remark 3.10)*
   - $\mathcal{S} \subsetneq \mathcal{R}_D \subset \mathcal{D}$ *(Remark 3.4 and Lemma 3.9)*
   - $\mathcal{R}_D \not\subset \mathcal{D} \cap \mathcal{Q}$ *(Corollary 2)*
   - $\bar{\mathcal{D}} = \mathcal{D} \cap \mathcal{Q}$ *(Lemma 3.10).*
2. *Between $\mathcal{J}, \mathcal{I}_G$ and $\mathcal{G}$ we have the following relations*
   - $\mathcal{I}_G \subset \mathcal{G}$ *(Remark 3.4).*
   - $\mathcal{J} \subsetneq \mathcal{G}$ *(Lemma 3.9)*
   - $\mathcal{I}_G \cap \mathcal{J} = \mathcal{I}_J$ *(Lemma 3.9)*

Based on these relations, we investigate whether it is possible to extend the classes $\mathcal{D}$ and $\mathcal{G}$. Since we require that for the definite/goal formulas the transformations from Lemma 3.6 are NA^ω-valid, the extension by formulas from $\mathcal{Q}$ and $\mathcal{J}$ is not possible.

- $J \to Q$ from $\mathcal{Q} \setminus \bar{\mathcal{D}}$, with $J \notin \mathcal{I}_G$. Since $J \notin \mathcal{I}_G$ it is relevant, so in order to show $J \to J^F$ as in Lemma 3.6 we would need $\mathsf{efq}^\bot$, which is not permitted in NA^ω.

- $J \to Q$ from $\mathcal{Q} \setminus \bar{\mathcal{D}}$, with $Q \notin \mathcal{D}$. If $Q \notin \mathcal{D}$, then it is not possible to derive $Q^{\mathrm{F}} \to Q$, as required by Lemma 3.6.

- By the above, if $Q \notin \bar{\mathcal{D}}$, then $\forall_x Q$ cannot be included in the class $\mathcal{D}$.

- In order to extend $\mathcal{I}_G$ by $\mathcal{J} \setminus \mathcal{I}_G$ we need to consider $S \to \bot$. By Lemma 3.6 this means that $\vdash_m (S \to \bot) \to S^{\mathrm{F}} \to \mathrm{F}$. This is not possible, since the derivation requires $\mathsf{efq}^\bot{}_{\mathrm{F}}$.

In fact, the converse is also the case: none of the formulas in the (relevant) definite and (irrelevant) goal formulas would be appropriate for extending $\mathcal{Q}, \mathcal{J}$ and $\mathcal{S}$, while at the same time keeping the properties guaranteed by (Ishihara, 2000). For instance, $\mathcal{J}$ cannot be extended by considering universally quantified formulas, since $\forall_x \tilde{\neg}\,\tilde{\neg} J \not\to \tilde{\neg}\,\tilde{\neg} \forall_x J$.

In conclusion, by analyzing the formulas which are in $\mathcal{Q}$ and $\mathcal{J}$, but not in $\mathcal{D}$ and $\mathcal{G}$, it becomes clear that the requirements imposed on the latter are all necessary. This is due to the fact that the provability in NA^ω of the transformations in Lemma 3.6 needs to be guaranteed. These restrictions are relaxed in case one allows $\mathsf{efq}^\bot$, as it is the case with the system used in (Ishihara, 2000).

Counterexample Following the analysis above, one is therefore tempted believe that the refined A-Translation can only be applied to the classes of formulas that we have introduced in Section 3.3.

Yet, (Schwichtenberg and Wainer, 2011) gives the following example:

$$S := \forall_x.\, ((Qx \to \mathrm{F}) \to \mathrm{F}) \to Qx$$
$$D := (\forall_x.\, Qx \to \bot) \to \bot.$$

It easy to derive $(S \to D)^{\mathrm{F}} \to S \to D$. However, since $D \notin \mathcal{D}$, it follows that $S \to D \notin \mathcal{D}$. In fact, since $D \notin \mathcal{Q}$ (and $S \notin \mathcal{J}$), it is also the case that $D \to S \notin \mathcal{Q}$.

We have not yet found in practice a use of such formulas, nor instances in which this situation can occur. Nevertheless, the fact that there exist formulas which are not definite/goal, but for which the properties specified by Lemma 3.6 still hold, leaves as an open question the full characterization of these classes.

A similar analysis can be carried out with respect to the wiping (W), spreading (S) and isolating (I) formulas defined in (Troelstra and Van Dalen, 1988). We do not carry out this comparison here, since also in this case no information on how to extend the definite/goal classes is obtained.

Connection with the critical formulas

Definition 3.13. *We define recursively a new class of formulas, $\mathcal{E}$ by*

$$\bot \to P, (\vec{S} \to \bot) \to E \in \mathcal{E}, \text{ with } E \in \mathcal{E}.$$

Terminology: $\mathcal{J}_{\mathcal{R}} := \mathcal{J} \setminus \mathcal{I}_J = \{\vec{S} \to \bot\}$ are *relevant $\mathcal{J}$-formulas.*

Lemma 3.12. *1. $\mathcal{E} \subset \mathcal{Q} \setminus \mathcal{D}$.*

2. Let $E \in \mathcal{E}$, with $P \in \mathcal{D}$ decidable. Then

$$E' := E[P/(P \to \bot) \to \bot] \in \mathcal{D}.$$

Thus, $\vdash E'^F \to E'$.

Proof. 1. "$\subset$" Let $E \in \mathcal{E}$, i.e., $E = (\vec{S_0} \to \bot) \to ... \to (\vec{S_n} \to \bot) \to P$.

Since $\vec{S_i} \in \mathcal{S}$ and $\bot \in \mathcal{J}$ and by this $\vec{S_i} \to \bot \in \mathcal{J}$ and since $P \in \mathcal{Q}$, it is clear that E is of the form $J \to Q \in \mathcal{Q}$.

On the other hand, E is neither of the form $I \to D$ (since clearly $\vec{S_n} \to \bot \notin \mathcal{I}_G$), nor is it relevant, since $P \notin \mathcal{R}_D$.

Therefore, $E \notin \mathcal{D}$.

"$\supset$" Let now a formula be in $\mathcal{Q} \setminus \mathcal{D}$. Then this is either $\forall_x Q$, with $Q \in \mathcal{Q} \setminus \mathcal{D}$ or $J \to Q$, with $Q \in \mathcal{Q} \setminus \mathcal{D}$ or

2. Let $E = (\vec{S_0} \to \bot) \to ... \to (\vec{S_n} \to \bot) \to P \in \mathcal{E}$, with $P \in \mathcal{D}$ decidable. Then $(P \to \bot) \to \bot \in \mathcal{Q}$, because $P \to \bot \in \mathcal{J}$, so $E' \in \mathcal{E}$.

Moreover, $(P \to \bot) \to \bot \in \mathcal{R}_D$, because $P \to \bot \in \mathcal{G}$, when $P \in \mathcal{D}$ decidable. Since $\vec{S_n} \to \bot \in \mathcal{G}$, because $\mathcal{S} \subset \mathcal{R}_D$, we have that $E' \in \mathcal{D}$.

Let us show that indeed $\vdash E'^F \to E'$. Assume E'^F, $\vec{S_i} \to \bot$ and $P \to \bot$. We show $\bot$. For this, it suffices to show $\vec{S_i}$.

Since $\vec{S_i} \in \mathcal{S} \subset \mathcal{R}_D$, it suffices by Lemma 3.6 to infer $(\vec{S_i^F} \to F) \to \bot$. Assume $\vec{S_i^F} \to F$. We need to infer $\bot$; for this, it suffices to derive P.

Using the above assumptions on $E'^F := (\vec{S_0^F} \to F) \to ... \to (\vec{S_n^F} \to F) \to (P \to F) \to F$, we obtain $(P \to F) \to F$. Since P is decidable, this implies P.

□

From (Ishihara, 2000) we know that for any $Q \in \mathcal{Q}$, we have $\vdash Q^F \to Q^t$, where Q^t is the double-negated formula Q. Thus, all E-formulas have also this property and the proof is very similar to the one for Lemma 3.12.

We fix a set L of quantifier-free formulas and identify in this paragraph which $\mathcal{Q}$ -formulas may contain L-critical atoms, as defined in Chapter 8.

Remark 3.11. *The formulas in $\mathcal{E}$ are of the form*

$$E := (\vec{S_0} \to \bot) \to ... \to (\vec{S_n} \to \bot) \to P.$$

Thus, for a fixed set L, if $E \in L$, then P is an L-critical predicate symbol.

Remark 3.12. *Let L be a fixed set of quantifier-free formulas. For any $E \in \mathcal{Q}$ of L, if $E \in \mathcal{E}$, then E is L-critical. Hence, E captures the L-critical formulas from $\mathcal{Q}$.*

3.6 Summary of the Chapter

In this chapter we have reviewed the A-Translation method which combines Friedman's "trick" with the Gödel-Gentzen Double-negation Translation. Further, we have presented the (Berger et al., 2002) refinement of the A-translation for minimal logic, while remarking that by working in NA^ω the Double-negation translation can be avoided. A characteristic of the proposed translation consists in regarding $\bot$ as a placeholder, by defining it as a predicate variable and allowing substitution of computationally relevant formulas for it. In the refined method, some of the $\bot$-negations are marked as computationally irrelevant, through their substitution by the constant F (Lemma 3.6). Only the remaining logical negations are replaced by Friedman's trick with the strong version of the goal formula, $\exists_x \vec{G}$. As a consequence of this refinement, the extracted terms are "brushed-up" of irrelevant computations present in the original proofs due to the inserted double negations.

A special emphasis lies on identifying the classes of formulas to which the refined A-Translation can be applied. Lemma 3.6 limits the structure of the

involved formulas and the resulting classes are called "definite" and "goal". We have compared in Section 3.5 the approaches from (Berger et al., 2002) and (Ishihara, 2000), in order to get more insight regarding the structure of the formulas and the role of the restrictions. We have concluded that an extension of the definite/goal classes by the additional formulas from (Ishihara, 2000) is not possible in NA^ω when we require that Lemma 3.6 holds. In order to transform further formulas into goal/definite ones, we will propose in Chapter 8 a refined form of the double negation translation.

By the refined A-Translation, the proofs of $\bot$ are transformed in proofs of $\exists_x G$, such that the constructive meaning of the existential quantifier is recovered. The translated proofs can be interpreted by (modified) realizability, such that computational terms are associated to these proofs. We have given in Section 3.4.2 the extraction rules by which we associate realizers to A-translated proofs. In Chapters 5-7 we will exemplify this extraction procedure on interesting case studies. We will give in each case the proof terms and for the simpler situations we will describe the way in which A-translation is applied.

Chapter 4

A-Translation and Programming

We overview in this chapter a couple of A-Translation specific programming techniques. As a result of applying the refined method on classical proofs, the extracted programs have properties which are desirable from the programming point of view:

- they are in general in tail recursive form; tail recursion allows for an easy transformation into iteration
- they give possibility to use abort, which can be seen as computing with exceptions and to use continuations, which simulate jumps.

We will explain intuitively why the refined A-Translation results in general in tail recursive programs and in which way the control/abort operators are present in these programs. We argue that by using A-Translation no further transformation of the proofs is required, in order to guarantee these features of the extracted programs. In the following chapters, we will exemplify the claims on concrete case studies.

In the first part, Section 4.1, we underline the connection with tail recursion. We oppose this (Section 4.1.2) to a related work of (Chiarabini, 2010), in which this programming feature is obtained by a careful manipulation of the constructive proofs.

In the second part, Section 4.2, we present the work of (Griffin, 1990), which uses continuation operators in order to realize the weak existence quantifier ($\tilde{\exists}$) and disjunction ($\tilde{\vee}$) and in order to interpret some classical principles, such as Peirce's Formula. We observe that the programs extracted from the refined A-translated proofs also adhere to the continuation

passing style. We comment in Section 4.2.2 on these similarities and explain intuitively the mechanism giving rise to programs using continuations.

4.1 Tail Recursion

Tail recursion is regarded in computer science as an efficient form of recursion since it allows for an optimization in the sense that it can be easily transformed into iteration. Based on the intuitive definition presented in what follows, we give in this section a more formal one in terms of λ-calculus.

4.1.1 Tail Recursion in Computer Science

A recursion is considered to be in tail (or tail-end) form if the step of the recursion consists of a *tail* call, i.e., the recursive call is the last operation performed by the function.

Therefore, the following restrictions are imposed on program terms:

- An "$\mathsf{if}\,(c)\,\mathsf{then}\,t\,\mathsf{else}\,e$" statement cannot contain the recursive call inside the condition c
- In the recursive terms, no other operation should be performed on top of the recursive call.
- The kernels of the λ-expressions are in tail-form.

Intuitively, for these requirements to be met, it suffices that the recursive call is the outmost function in the step term of a recursion operator.

Let us regard an example, written in meta-language:

Example 2. *We define a function that computes the sum of a list of natural numbers. We consider the following possibilities:*

- $sum(l) := \mathsf{if}\,(l = \mathsf{nil}^{\mathbb{N}})\,\mathsf{then}\,0\,\mathsf{else}\,l_0 + sum(\mathsf{cdr}(l))$.

  In this case, sum is not a tail recursive function, since the addition is performed after *the recursive call.*

- $sum(l) := sum_{aux}(l, 0)$, *where*
  $sum_{aux}(l, r) := \mathsf{if}\,(l = \mathsf{nil}^{\mathbb{N}})\,\mathsf{then}\,r\,\mathsf{else}\,sum_{aux}(\mathsf{cdr}(l), l_0 + r)$.

  The addition was pushed inside the recursive call of sum_{aux}. Since no further operations are performed after the recursive call., this is a tail-recursive variant of sum.

We summarize the above conditions in the following inductive definition.

Definition 4.1 (Tail recursion)**.** *A term T is in tail form if*

$$T := [\,] \mid x \mid (\lambda x T) \mid (T_1\, T_2) \mid \langle T_1, T_2\rangle \mid T_0 \mid T_1 \mid \mathcal{R}^\sigma_\tau t^\tau T_1\ (\lambda t, p, \vec{x}.\ p\ T_2) \mid \mathcal{C}^\sigma_\tau c^\tau T_1\ T_2,$$

with $\mathcal{C}\ T_1\ T_2$, denoting the $\mathsf{if}\,(c)\,\mathsf{then}\,T_1\,\mathsf{else}\,T_2$ *construct.*

Remark 4.1. *The advantage of tail recursive functions is that they can be easily transformed to iterations. This allows for compiler optimizations, since the intermediate information does not need to be saved on - and therefore later on recovered from - the stack, as is the case with non-tail recursion.*

The iterations obtained in this way are a form of recursion. For instance, for the natural numbers we have $\mathrm{It}^\sigma_{\mathtt{N}} :\ \mathtt{N} \Rightarrow \sigma \Rightarrow (\mathtt{N} \Rightarrow \sigma \Rightarrow \sigma) \Rightarrow \sigma$ with

$$\mathrm{It}_{\mathtt{N}}(0, f, g) = f, \qquad \mathrm{It}_{\mathtt{N}}(Sn, f, g) = g(\mathrm{It}_{\mathtt{N}}(n, f, g))$$

The step case is similar to $\mathcal{R}_{\mathtt{N}}(Sn, f, g)$, with the exception that n is not used by g to compute the final result.

4.1.2 Connection with the Refined A-Translation

Observation 4.1. *The recursion extracted from an A-translated inductive proof in NA^ω is in general in tail form.*

Justification. The property that the recursion is in tail form characterizes in fact the classical proofs and not the refined A-Translation. It is only because the A-Translation preserves the structure of the classical proofs to which it is applied and because these are in fact proofs of $\bot$ that this property propagates to the transformation method.

Let C be a formula shown by induction. In order for the realizer associated to the A-translated proof of C to be non-empty, the formula must be relevant, i.e., $C := \forall_n\ \tilde{\neg} B(n)$.

In the step case of the inductive proof we show $\forall_n.\ \tilde{\neg} B(n) \to\ \tilde{\neg} B(\mathsf{S}n)$.

For this, we assume n, $u_1 :\ \tilde{\neg} B(n)$ and $u_2 : B(\mathsf{S}n)$ and need to infer $\bot$. Since u_1 ends in $\bot$, in general this is applied first (so in head position) to a term deriving $B(n)$, in order to prove $\bot$. In fact, this is always the case when B is irrelevant. In such cases, the proof term for the step case is

$$M_{Step} := \lambda n, u_1, u_2.\ N^\bot = \lambda n, u_1, u_2.\ u_1\ N'[u_2]^{B(n)}.$$

After A-Translation, by the rules from Section 3.4 and since the assumption variables u_1, u_2 contain $\bot$ (so are relevant), the computational term associated to the step case is:

$$[\![M_{Step}]\!] = \lambda t, x_{u_1}, x_{u_2}.\ x_{u_1}[\![N']\!],$$

where the typing is $x_{u_1}^{\tau(n)\to(\tau(n)\to\tau(\vec{G}))\to\tau(\vec{G})}, x_{u_2}^{\tau(n)\to\tau(\vec{G})}$. Notice that x_{u_1} is the functional corresponding to the recursive call. As a consequence of the fact that u_1 is applied first, the recursive call invokes x_{u_1} in tail position and all subsequent computations corresponding to $C[u_2]$ take place inside the recursion call.

□

We illustrate this in what follows by the example of the factorial problem.

Example 3. *Let $f^{\mathtt{N}\Rightarrow\mathtt{N}}$ be the factorial function and fnk the specification for $k = f(n) = n!$. Assuming*

$$\mathsf{InitFact} : f\,0\,1 \text{ and } \mathsf{GenFact} : \forall_{n,k}(fnk \to f\,\mathsf{S}n\,(\mathsf{S}n \cdot k))^1$$

we show:

$$\mathsf{FCI} : \forall_n \tilde{\exists}_k f\,n\,k$$

by:

$$\begin{aligned}
M_{FCl} &:= \lambda n_0.\ \mathsf{Ind}\ n_0\ M_{Base}\ M_{Step}\\
M_{Base} &:= \lambda u_1^{\forall_k(f\,0\,k\to\bot)}.\ u_1\ 1\ \mathit{InitFact}\\
M_{Step} &:= \lambda n_1, u_2^{\tilde{\exists}_k f n_1 k}, u_3^{\forall_k(f(\mathsf{S}n_1)k\to\bot)}.\\
&\qquad u_2\ (\lambda k, u_4^{fn_1k}.\ (u_3\ (n_1 \cdot k + k)\ (\mathit{GenFact}\ n_1\ k\ u_4))^{\bot})^{\forall_k\ \tilde{\neg} f n_1 k}.
\end{aligned}$$

We associate $F^{\mathtt{N}\Rightarrow\mathtt{N}}$ to the assumption variable u_3 and $H^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$ to u_2. Since u_4 is irrelevant, the associated term is ε. The program extracted from the A-translated proof M_{FCl} is:

$$\begin{aligned}
P_1 &:= \lambda n_0.\ \mathcal{R}_{\mathtt{N}}^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}\ n_0\ P_{Base}\ P_{Step}\ (\lambda n.\ n), \textit{with}\\
P_{Base} &:= \lambda F.\ F\,1\\
P_{Step} &:= \lambda n_1, H, F.\ H\ (\lambda n_2.\ F\ \mathsf{S}n_1 \cdot n_2).
\end{aligned}$$

[1] S binds stronger than "$\cdot$".

Clearly, P_{Step} fits Definition 4.1. The application of the recursion is the outermost operation since H is the applied on top position in the step term of P_1. Thus, P_1 is a tail-recursive program.

We take $P_1'(n) := \mathcal{R}\ n\ P_{Base}\ P_{Step}$ and have

$$\begin{aligned} P_1'(0) &= P_{Base}\ 0 = \lambda F.\ F\,1 \\ P_1'(\mathsf{S}n) &= \mathcal{R}\ \mathsf{S}n\ P_{Base}\ P_{Step} = \\ &= P_{Step}\,n\,(\mathcal{R}\ n\ P_{Base}\ P_{Step}) = \\ &= P_{Step}\,n\,P_1'(n) = \\ &= \lambda F.\ P_1'(n)(\lambda n_2.\ F\ \mathsf{S}n \cdot n_2), \end{aligned}$$

Thus, the program uses an auxiliary function G computing recursively the result as follows:

$$G(0, F) = F\,1 \text{ and } G(\mathsf{S}n, F) = G(n, \lambda n_2.\ F\ \mathsf{S}n \cdot n_2).$$

The result is obtained by applying G to the identity function. G (and therefore P_1) is tail recursive, since the computation is performed inside the recursive call.

Proof Transformation vs. the refined A-Translation

In what follows, we compare the above results, obtained by applying the refined A-Translation on classical proofs with the approach from (Chiarabini, 2010). In (Chiarabini, 2010) a proof transformation method is applied to constructive proofs in order to obtain tail recursion in the extracted programs. The key idea is to replace the (Ind)-scheme by

$$P0 \to \forall_n(Pn \to P(\mathsf{S}n)) \to \forall_n Pn, \qquad \text{(IndCont)}$$

proven by means of

$$P0 \to \forall_n(Pn \to P(\mathsf{S}n)) \to \forall_n \forall_m^{nc}((Pn \to P(n+m)) \to P(n+m)).$$

By this, the computation of the recursion step is pushed inside the recursive call.

We present this transformation in what follows on the factorial example from Section 3. We will show that what is achieved in (Chiarabini, 2010) by a clever proof transformation is a direct consequence (or side-effect) of applying A-Translation to the classical variant of this proof.

Example 4. *The goal formula of the factorial problem from Example 3 is formulated by the strong form of* $\exists$*:*

$$\mathsf{FCon} : \forall_n \exists_k f\, n\, k$$

and we make the same assumptions InitFact *and* GenFact.

Using the following instances of the axioms for $\exists^-$ *and* $\exists^+$*:*

- $\exists^+ : \forall_n^{nc}.\, \forall_m (f\, n\, m \to \exists_k f\, n\, k)$
- $\exists^- : \forall_n^{nc}.\, \exists_m f\, n\, m \to \forall_m (f\, n\, m \to \exists_k f\, \mathsf{S}n\, k) \to \exists_k f\, \mathsf{S}n\, k.$

we construct the λ*-proof:*

$$\begin{aligned} M_{FCon} &:= \lambda n_0.\, \mathsf{Ind}\ n_0\ M_{Base}\ M_{Step} \\ M_{Base} &= \exists^+\, 0\ 1\ \mathit{InitFact} \\ M_{Step} &= \lambda n, \mathit{IH}^{\exists_k f\, n\, k}.\, (\exists^-\, n\, \mathit{IH}) \\ &\qquad \lambda k, H^{f\, n\, k}.\, \exists^+\, \mathsf{S}n\, (\mathsf{S}n \cdot k)\, (\mathit{GenFact}\ n\ k\ H). \end{aligned}$$

Already at this level, we observe that the induction hypothesis is not applied in head position. We obtain from M_{FCon} *the program:*

$$\begin{aligned} P_2 &:= \lambda n_0.\, \mathcal{R}^{\mathtt{N} \Rightarrow \mathtt{N}}\ n_0\ P_{Base}\ P_{Step} \\ P_{Base} &= 1 \\ P_{Step} &= \lambda n_1, n_2.\, n_1 \cdot n_2 + n_2. \end{aligned}$$

This unfolds as:

$$\begin{aligned} P_2(0) &= 1 \\ P_2(\mathsf{S}n) &= \mathcal{R}\ (\mathsf{S}n)\ 1\ (\lambda n_1, n_2.\, n_1 \cdot n_2 + n_2) = \\ &= (\lambda n_1, n_2.\, n_1 \cdot n_2 + n_2)\ n\ (\mathcal{R}\ n\ 1\ (\lambda n_1, n_2.\, n_1 \cdot n_2 + n_2)) = \\ &= (\lambda n_1, n_2.\, n_1 \cdot n_2 + n_2)\ n\ P_2(n) = \\ &= n \cdot P_2(n) + P_2(n) \end{aligned}$$

Thus, the constructive proof gives the recursive program

$$P_2(0) = 1, P_2(\mathsf{S}n) = \mathsf{S}n \cdot P_2(n),$$

which is clearly not tail recursive, because the multiplication is performed outside the recursion.

We apply the proof transformation from (Chiarabini, 2010) and use (IndCont)*, thus constructing the proof:*

$$M_3 := \mathsf{IndCont}\, M_3', \text{ where}$$
$$M_3' := (\exists^+\ 1\ \mathsf{InitFact})$$
$$\lambda n, IH^{\exists_k f n k}.\ \exists^-\ n\ IH\ (\lambda k, H^{f n k}.\ \exists^+\ n\ \mathsf{S}n \cdot k\ \mathsf{GenFact}\ n\ k\ H).$$

Expanding IndCont *we have:*

$$\begin{aligned} M_3 &:= \lambda u_1, u_2, n.\ ((M_{3,Ind}\, u_1\ u_2\ n\ 0)\lambda u_3.\ u_3)\ M_3' \\ M_{3,Ind} &:= \lambda u_4, u_5, n'.\ Ind\ n'\ M_{3,Base}\ M_{3,Step} \\ M_{3,Base} &:= \lambda m, v_1.\ v_1\ u_4 \\ M_{3,Base} &:= \lambda n, v_2, m, v_3.\ (\lambda v_4.\ v_2\ \mathsf{S}m\ v_4)\lambda v_5.\ v_3(u_5\ n\ v_5), \end{aligned}$$

where $\exists^+/\exists^-$ *are the introduction/elimination axioms for the strong existential quantifier. The assumption variables have the following types:*

$$\begin{aligned} &u_1^{\exists_k f 0 k}, u_2^{\forall_n.\ \exists_k f n k \to \exists_k f(\mathsf{S}n)k}, u_3^{\exists_k f n k}, u_4^{\exists_k f 0 k}, u_5^{\forall_n.\ (\exists_k f n k \to \exists_k f(\mathsf{S}n)k)} \\ &v_1^{\exists_k f 0 k \to \exists_k f(0+m)k} v_2^{\forall_m^{nc}.\ (\exists_k f n k \to \exists_k f(n+m)k) \to \exists_k f(n+m)k}, \\ &v_3^{\exists_k f \mathsf{S} n k \to \exists_k f \mathsf{S}(n+m)k}, v_4^{\exists_k f n k \to \exists_k f(n+\mathsf{S}m)k}, v_5^{\exists_k f n k} \end{aligned}$$

As a result of using this technique, we obtain the tail-recursive program:

$$\begin{aligned} P_3 := \lambda n_0.\ \mathcal{R}^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}\ n_0 \\ (\lambda F.\ F\,1) \\ (\lambda n_1, H, F.\ H\ (\lambda n_2.\ F\ (n_1 \cdot n_2 + n_2))) \\ (\lambda n.\ n), \end{aligned}$$

with $F^{\mathtt{N}\Rightarrow\mathtt{N}}$ *and* $H^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$.

This is identical to P_1 *(on page 58), the program extracted from the A-translated classical proof of the factorial.*

Since the refined A-translated version of the classical proof already extracts into a tail recursive program, it is doubtful that in this case the transformation procedure from (Chiarabini, 2010) would be of any computational gain. Indeed, using (IndCont) to prove the factorial yields the program:

$$\begin{aligned} P_{Base} &:= \lambda g.\ g\ (\lambda F.\ F\,1) \\ P_{Step} &:= \lambda n_1, Hc, g.\ Hc\ (\lambda H.\ g\,(\lambda F.\ H\ (\lambda n_2.\ F\ (n_1 \cdot n_2 + n_2)))) \\ P_4 &:= \lambda n_0.\ \mathcal{R}\ n_0\ P_{Base}\ P_{Step}\ (\lambda H.\ H)\ (\lambda n.\ n), \end{aligned}$$

where $Hc^{(((\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N})\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N})\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$ and $H^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$.

The type of the step function g is lifted to $((\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N})\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}$ and that of the recursion operator $\mathcal{R}$ to

$$\mathtt{N}\Rightarrow(((\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N})\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N})\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N},$$

in order to accumulate the results inside the recursive call. However, this brings no computational gain and the layer of imbrication is superfluous.

To sum up, while in the constructive case it is possible the directly extract programs from proofs, the recursions present in such programs are not in tail form. (Chiarabini, 2010) proposed some transformations at the proof-level in order to obtain tail-recursive programs, which are less expensive computationally then their counterparts using standard recursion. In the case of A-translated proofs, however, the recursion is in general in tail form, since the structure of the classical proofs is preserved. We can view this as a fortunate side-effect of working in NA^ω in which existence is shown indirectly.

Further examples. In Chapters 5 - 7 we will present more complex case studies. For each of these examples, we will provide the proof terms and the extracted program, in order to illustrate the above observation.

Counter-example. In Observation 4.1 we have specified that there is *in general* a connection between the refined A-translated proofs and tail recursion. In what follows, we give an example to illustrate a situation when this is not the case.

We regard the program extracted from the proof of the Corollary of Infinite Pigeonhole Principle. The step term

$$G_1 = \lambda n_1, g, f', h, p.\ p\ n_1\ (\lambda n_2, u_1.\ g\,(\lambda n_3.\ f'\,(n_2 \sqcup n_3))M'$$

for the recursion operator $\mathcal{R}_1$ in M_{IPHcor} (given on page 89) is not in tail form, since the step function g is applied inside the recursive call. This comes from the proof strategy used to show IPH, where the term u for the induction hypothesis is used inside $M_{G(f,q)}$.

4.2 Computing with Continuations

Another useful programming technique is that of computing with continuations, by which constructs such as control operators or abort are added to

the language. In what follows, we overview the results from (Griffin, 1990) regarding the interpretation of classical principles via the control operators and establish a connection with the programs that we extract from the refined A-translated classical proofs. As we will see, an explicit extension of NA^ω by control operators is not necessary, since the extracted programs simulate the continuation passing style, due to the interpretation given to $\bot$ in the process of refine A-translating.

4.2.1 Classical Proofs and the Notion of Control

Griffin bases his analysis on the work of Felleisen[2], who has introduced the notion of continuation/control context and the abort ($\mathcal{A}$) and control ($\mathcal{C}$) operators. In (Griffin, 1990) these operators are associated to the classical axioms and the weak operators, in a way which we briefly overview below.

Definition 4.2 (Continuation context)**.** *An evaluation context E is of the form*

$$E := [] \mid E\,N \mid V\,E,$$

where $[]$ is a hole, N a λ-expression and V a value (variable or λ-expression).

The understanding is that the term $E[M]$ is the result of placing M in the hole of the evaluation context E. If M is not a value and $E[M]$ is part of an evaluation sequence, then E must wait for M to evaluate, in order for the evaluation sequence to resume computing further subterms of E. In such a case, E represents the *continuation or control sequence* of M.

Definition 4.3 (Control and abort operators)**.** *The λ-expressions are extended by the following operators:*

- abort*, $\mathcal{A}$, given by the rule*

$$E[\mathcal{A}(M)] \mapsto_{\mathcal{A}} M.$$

  Intuitively, this means that the control context in which M is placed is abandoned and the computation resumes with M.

- control*, $\mathcal{C}$, with the semantics described in terms of*

$$E[\mathcal{C}(M)] \mapsto_{\mathcal{C}} M\lambda z.\,\mathcal{A}(E[z]).$$

[2]Selected works of Felleisen cited by Griffin are (Felleisen et al., 1987), (Felleisen and Friedman, 1986) and (Felleisen et al., 1986), but his work in this direction has further developed.

The intended meaning of $\mathcal{C}$ becomes more clear by regarding the following situation

$$E'[(\lambda z.\ \mathcal{A}(E[z]))\, V] = E'[\mathcal{A}(E[V])] \mapsto_{\mathcal{A}} E[V].$$

Intuitively, this means that the context E' is abandoned and the control returns to the context E.

Remark 4.2. *$\mathcal{A}$ can be defined in terms of $\mathcal{C}$, with d not free in M:*

$$\mathcal{A}(M) = \mathcal{C}(\lambda d.\ M).$$

Typing. For $\mathcal{C}$, this is defined by (Griffin, 1990) to be:

"If M has the type $(\alpha \to \beta) \to \beta$, then $\mathcal{C}(M)$ has type α",

where, in order to be logically consistent, β must be a proposition which has no proof, i.e. $\bot$. As a consequence, $\mathcal{C}(M)$ is associated with $\mathsf{Stab}^{\bot}$ and further, $\mathcal{A}(M)$ corresponds to $\mathsf{efq}^{\bot}$. Also, the typing for M implies that the term is of the form $\lambda k^{\tilde{\neg}\alpha}.\ k\, M$.

Further, (Griffin, 1990) introduces the computational terms for disjunction and conjunction, distinguishing between call-by-value and call-by-name computations. Of interest to us is in particular the treatment of weak existential. For the $\tilde{\exists}$-introduction rule, the term considered is

$$P_2 = \lambda x, w^{\alpha(x)}.\ \lambda f^{\forall_y\, \tilde{\neg}\alpha(y)} f\, x\, w.$$

and for the $\tilde{\exists}$-elimination rule the computational term:

$$P_1 := \lambda p^{\tilde{\exists}_x \alpha(x)}, f^{\forall_x.\ \alpha(x)\to\beta}.\ \mathcal{C}(\lambda j^{\tilde{\neg}\beta}.\ p\,(\lambda x, w.\ j\,(f\, x\, w)))$$

Is is important to observe that the control operator is used to make the transition from $\tilde{\neg}\,\tilde{\neg}\beta$ to β, thus mapping $\mathsf{Stab}^{\bot}$.

4.2.2 Connection with the Refined A-Translation

As briefly discussed in Chapter 2 - in Section 2.3.2 - and for reasons that should become more clear in Chapter 8, we work in a (minimal) system without $\mathsf{Stab}^{\bot}$ and $\mathsf{efq}^{\bot}$. The double negation of the logical stability axiom and of the logical ex-falso are however provable. The logical falsity plays a special role for the refined A-Translation: taken to be a predicate variable, $\bot$ allows its substitution by computational terms. However, $\bot$ in itself cannot be interpreted computationally, so it plays the role of a *hole* in the computational context. Consequently, whenever $\bot$ is present at the proof level, this results in a computation by continuations in the extracted program.

Observation 4.2. *Assume that we have shown by the proof M a Π^0_2-formula:*

$$B := \forall_{\vec{x}}.\ \vec{D} \to \vec{H} \to \tilde{\exists}_y\ \vec{G} \tag{4.1}$$

The realizer for the proof of B depends on the realizers for the logical falsity $\bot$ occurring in the proof after the refined A-translation. These realizers simulate a controlled computation, in the sense that they allow to abort and to compute with jumps.

Proof Sketch. We apply the refined A-Translation method to the proof of B. Replacing $\bot$ in this process helps recover the constructive meaning of $\exists$. Further, we apply the modified realizability rules and by Theorem 3.5, the refined A-translated $B^{\exists_y G}$ is realized by

$$[\![M]\!]\vec{t_D}\, r\, s_G\, \mathbf{mr}\, \exists_y G^F,$$

where t_D and s_G are given by Lemma 3.8, such that

$$D^F \to t_D\, \mathbf{mr}\, D \text{ and } \forall_{v,w}.\ v\, \mathbf{mr}\, G \to (G^F \to w\, \mathbf{mr}\, \bot) \to s_G\, v\, w\, \mathbf{mr}\, \bot$$

are derivable from the proofs of (3.2) and (3.4) (given on page 35) using $\forall_y(\mathrm{F} \to y\, \mathbf{mr}\, \bot)$ and efq_P.

The realizer for the goal, $\bot$, thus depends on the realizers s_G, v and w. In particular the latter is a realizer for $\bot$, thus only known when this occurrence is instantiated by a computationally relevant formula (or some dummy variable). Therefore, the realizer for the goal formula is part of a context in which we need to wait for another value to be computed first. To take this into consideration, we must allow for *abort*, to account for all situations. By this, as soon as a suitable candidate for w is found, the context in which this was computed exits with the current result. By a controlled book-keeping of the ongoing computation, this result for w is passed to the context waiting for it. In the case when the current context cannot compute a satisfactory result based on the value for w, the computation must backtrack, such that a new candidate for w is computed. As a consequence, the goal $\bot$ is placed in a context in which certain computations can be abandoned, such that the control is passed on to the outer context. One can also regard this as a computation with *jumps*. In the context of A-translation, this is achieved implicitly, by generating a context similar to the one involving the *control* operator from (Griffin, 1990).

□

We will see this phenomenon exemplified in the case of Stolzenberg's Principle and the Infinite Pigeonhole Principle in Chapter 5. In these examples, in case the evaluation fails for some conditions from the specification $\vec{G}$, the accumulated results intended to produce the output for $\exists_x G$ are abandoned. The computation resumes (or jumps back to) from the point where the intermediate result used in the computation producing x was computed. As a consequence of abandoning the intermediate results when constructing x, the program does not use, in general, the first entries from the input data producing the witness for x. A brute-force method would find the first candidates, but might necessitate a longer computing time - the corollary of Dickson's Lemma presented in Chapter 7, Section 7.4 provides such an example in which the computation with jumps is faster.

The implicit *abort* in the terms associated to classical proofs is correlated with the fact that A-Translation enables us to extract programs from proofs using classical principles, which cannot be otherwise interpreted by computable functionals. Consider the example of Dickson's Lemma, presented in detail in Section 7.4. In the proof we use the minimum principle in order to find two values with the desired properties. At the computational level these values cannot be guaranteed to be the minimal values in an infinite input sequence. This is a classical property for which it is not possible to find an associated computable functional. It is rather the case that once a potential suitable candidate is found, the verification whether this is indeed a minimum is abandoned. A jump to the context in which this is compared to the next value is performed; in case this pair of indices fulfills G, the result is returned. If the pair fails the test, this computation is abandoned and a new search for a potential candidate for first index is started, beginning at a position in the sequence which was abandoned in the previous search. We present in Chapter 7 some concrete runs of this program.

4.3 Summary of the Chapter

In this chapter we argue that transforming classical proofs by A-Translation in order to synthesis their computational content has beneficial side-effects from the computational point of view. In the case of constructive proofs, as (Chiarabini, 2010) suggests, it is possible to obtain tail-recursive programs from inductive proofs, by applying transformations at the proof level. However, as we have observed in this chapter, when using the refined A-Translation on NA^{ω}-proofs we obtain in general recursions in tail form, without further transformation of the proofs.

We have also compared the effects of the refined A-Translation with the work of (Griffin, 1990) in which control operators are associated to the classical principle $\mathsf{efq}^{\perp}$ and Peirce's Law. (Griffin, 1990) interprets the existence and disjunction operators in a manner identical to our distinction of weak vs. strong operators and has also a special treatment of negation. In our interpretation of $\perp$, this plays the role of a placeholder and is substituted during the translation by the (strong) $\exists$-goal. As a result, the extracted program computes the results by means of the control and abort operators, simulating the computation by continuations suggested in (Griffin, 1990).

Chapter 5

The Infinite Pigeonhole Principle

We apply in what follows the mechanism of the refined A-Translation coupled with modified realizability to the theorem stipulating the existence of a constant subsequence in an infinite sequence taking values from a finite set. We will start with a simpler case and consider an infinite boolean sequence and then present the generalization to what is known as the Infinite Pigeonhole Principle (IPH).

The example is suitable for understanding how the refined A-translation affects proofs and enables program extraction, and for the simpler case we will trace its steps in more detail. For the generalized version, we will illustrate the strong connection with the Continuation Passing Style (CPS). We will also emphasize other characteristics of the programs extracted from A-Translated classical proofs, such as the asymmetrical behavior.

5.1 The Infinite Boolean Tape

We present in what follows a simplified version of IPH, in which we consider a 2-colored sequence or a "boolean tape". The example is traced by (Coquand, 1995) back to Stolzenberg.

In his PhD thesis, (Urban, 2000) presents this case study in order to illustrate the argument that by cut-elimination procedures one has more flexibility in associating programs to proofs. For this, he gives four distinct normal forms corresponding to the same proof of Stolzenberg's example. In the following, we will show how to achieve this by instead handling the symmetries that we observe at the proof level. We claim that when using

A-Translation we can reach the same variety of extracted programs, by carefully manipulating the formalization of the proof.

5.1.1 Statement and Formalization

Different formulations have been chosen in the literature, in order to state Stolzenberg's principle, which can be also viewed as an instance of the excluded middle:

- Given R infinite and $R_0 \subseteq R$, either R_0 or $R_1 := R \setminus R_0$ is infinite.
- For any $R \subset \mathtt{N}$ there exists $R_0 \subseteq \mathtt{N}$ infinite such that either $R_0 \subset R$ or $R_0 \subset \complement R$ (Veldman and Bezem, 1993).
- Let $\mathcal{S} := \{\, f^{\mathtt{N}\Rightarrow\mathtt{N}} \mid \forall_n f(n) < f(\mathsf{S}n) \,\}$. Then

$$\forall_{A\subset\mathtt{N}}.\ \tilde{\exists}_{f\in\mathcal{S}}\forall_n f(n) \notin A \vee \tilde{\exists}_{f\in\mathcal{S}}\forall_n f(n) \in A. \tag{5.1}$$

  Similarly, if we consider the characteristic function of A, we have

$$\forall_{f_A^{\mathtt{N}\to\mathtt{B}}}.\ \tilde{\exists}_{f\in\mathcal{S}}\forall_n f_A(f(n)) = 0 \vee \tilde{\exists}_{f\in\mathcal{S}}\forall_n f_A(f(n)) = 1 \tag{5.2}$$

Our choice of formalization is as follows:

Proposition 5.1 (Stolzenberg's Principle)**.** *On an infinite tape with cells containing either 0 or 1, there exist two (distinct) cells having the same content.*

Proof Idea. We observe that an infinite boolean tape has either infinitely many 0's or infinitely many 1's, from which our claim trivially follows. □

We take $f^{\mathtt{N}\Rightarrow\mathtt{N}}$ to encode[1] the infinite tape. Thus, our hypothesis is

$$A:\ \forall_n.\ f(n) = 0 \mathbin{\tilde{\vee}} f(n) = 1 \tag{5.3}$$

and we want to prove that

$$G : \tilde{\exists}_{n,m}.\ n < m \mathbin{\tilde{\wedge}} f(n) = f(m). \tag{5.4}$$

As presented in Section 2.2 from the chapter introducing the language, $\tilde{\vee}$ is viewed only as an abbreviation. Thus, (5.3) can be expressed by (variants of):

$$A : \forall_n.\ (f(n) = 0 \to \bot) \to (f(n) = 1 \to \bot) \to \bot \tag{5.5}$$

[1] We could take a two-valued function $f^{\mathtt{N}\Rightarrow\{0,1\}}$ instead, but we choose the more general form, in view of further generalizations of the proposition.

Instead of directly proving Proposition 5.1, we formulate the proof idea as the property:

$$\forall_n \, \tilde{\exists}_k. \;\; n \leq k \;\; \tilde{\wedge} \;\; fk = i, \; i = 0, 1 \qquad (\infty_i)$$

and use this to show G by splitting the proof in two parts; first, we show the following lemmas:

$$A \to \infty_0 \;\tilde{\vee}\; \infty_1 \qquad (\infty_0 _or_ \infty_1)$$
$$\infty_i \to G, i \in \{0, 1\} \qquad (\infty_i \to G)$$

and then put them together by a cut, to show Stolzenberg's Principle.

As with (5.3), the $\tilde{\vee}$ in $(\infty_0 _or_ \infty_1)$ needs to be rewritten, such that we have (variants of)

$$A \to \;\tilde{\neg}(\infty_0) \to \;\tilde{\neg}(\infty_1) \to \bot \qquad (\infty_0 \;\tilde{\vee}\; \infty_1)$$

Remark 5.1. *By "variants of" $(\infty_0 \;\tilde{\vee}\; \infty_1)$, we refer to the fact that, on the one hand, if the double negation is not essential in order to carry on the proof in minimal logic, then we can (classically) drop it. We will discuss in Subsection 5.1.2 below and further in Chapter 8 when the double negations are necessary. On the other hand, the symmetry enables us in addition to consider either of the (classically) equivalent forms:*

$$A \to \;\tilde{\neg}(\infty_0) \to \infty_1 \qquad (\neg\infty_0 \to \infty_1)$$
$$A \to \;\tilde{\neg}(\infty_1) \to \infty_0 \qquad (\neg\infty_1 \to \infty_0)$$

First Auxiliary Property. We first show the stronger statement:

Proposition 5.2. *The infinite tape with cells containing either 0 or 1 has either infinitely many 1's or infinitely many 0's.*

Proof. We give the proofs in natural deduction style for both variants:
$(\neg\infty_0 \to \infty_1)$ and $(\neg\infty_1 \to \infty_0)$.

Let $(m \sqcup n)$ abbreviate $\max(m, n)$ and $P(s) : s \leq (m \sqcup n)$. It is easy to show the auxiliary properties $\forall_{m,n} P(m)$ and $\forall_{m,n} P(n)$, so we consider them as part of the library of available lemmas.

Assume A and for $i = 0, 1$ let $u_{i,s} : \forall_k. \, s \leq k \to f(k) = i \to \bot$.

We have:

$$\dfrac{\dfrac{\text{(5.5): A } (m \sqcup n) \qquad \dfrac{u_{0,m}(m \sqcup n) \qquad P(m)}{f(m \sqcup n) = 0 \to \bot}\to^-}{(f(m \sqcup n) = 1 \to \bot) \to \bot}\to^- \qquad \dfrac{u_{1,n}\ (m \sqcup n) \qquad P(n)}{f(m \sqcup n) = 1 \to \bot}\to^-}{\bot}\to^-$$

Further, depending on whether we show $(\neg\infty_0 \to \infty_1)$ or $(\neg\infty_1 \to \infty_0)$:

$$\dfrac{\dfrac{w_0 :\ \tilde{\neg}\infty_0 \qquad \dfrac{\dfrac{\dfrac{\bot}{\tilde{\neg}u_{0,m}} \to^+ u_{0,m}}{\infty_0}\forall^+ m}{\bot} \to^-}{\tilde{\neg}u_{1,n}} \to^+ u_{1,n}}{\infty_1}\forall^+ n}{\tilde{\neg}\infty_0 \to \infty_1} \to^+ w_0}{A \to\ \tilde{\neg}\infty_0 \to \infty_1} \to^+ A$$

$$\dfrac{\dfrac{w_1 :\ \tilde{\neg}\infty_1 \qquad \dfrac{\dfrac{\dfrac{\bot}{\tilde{\neg}u_{1,n}} \to^+ u_{1,n}}{\infty_1}\forall^+ n}{\bot} \to^-}{\tilde{\neg}u_{0,m}} \to^+ u_{0,m}}{\infty_0}\forall^+ m}{\tilde{\neg}\infty_1 \to \infty_0} \to^+ w_1}{A \to\ \tilde{\neg}\infty_1 \to \infty_0} \to^+ A$$

□

The difference between these two last subproofs lies in the swap of 0 and 1. The symmetry will be reflected in the two distinct associated programs. However, the choice whether to first consider 0 or 1 is essential, as this consequently influences the order in which $u_{1,n}$ and $u_{0,n}$ are introduced and whether ∞_0 or ∞_1 is used. As we will see, this triggers an asymmetry in each of the extracted programs.

Second Auxiliary Property. The next step is to show $(\infty_i \to G)$, i.e., that G follows from either of $\infty_i, i \in \{0,1\}$ and this is immediate.

Proposition 5.3. *From infinitely many 0's or 1's, one can select two distinct occurrences of the same value*[2].

Proof. Let $falseH : \forall_{n,m}.\ n < m \to f(n) = f(m) \to \bot$. We assume that we already have in our library the auxiliary properties

$$\begin{aligned} Lem_{=Trans} &: \forall_{f,k,n,m}.\ f(n) = k \to f(m) = k \to f(m) = f(n) \\ Lem_{\mathsf{S}LeToLt} &: \forall_{n,m}.\ \mathsf{S}\ n \le m \to n < m. \end{aligned}$$

We show $\bot$.

$$\dfrac{Lem_{=Trans} \qquad w_1 : f(n) = i \qquad w_2 : f(k) = i}{f(k) = f(n) \quad (\mathcal{G}_1)} \to^-$$

[2]One can easily generalize this to select finitely many occurrences of either 0 or 1.

$$\dfrac{falseH\ k\ n \qquad \dfrac{Lem_{\mathsf{S}LeToLt}\ k\,n \qquad u : \mathsf{S}k \le n}{k < n}\to^- \qquad \dfrac{f(k) = f(n) \to \bot \qquad \mathcal{G}_1}{\dfrac{\bot}{f(n) = i \to \bot}\to^+ w_1}\to^- \qquad \dfrac{}{\forall_n.\, \mathsf{S}k \le n \to f(n) = i \to \bot \quad (\mathcal{G}_2)}\to^+ u, \forall^+ n}{\dfrac{(\infty_i)\ i \qquad \forall_k.\, i \le k \to f(k) = i \to \bot \qquad \dfrac{\dfrac{(\infty_i)\ \mathsf{S}k \qquad \mathcal{G}_2}{\bot}\to^- }{f(k) = i \to \bot}\to^+ w_2}{\bot}\to^+ v^{i \le k}, \forall^+ k}\to^-$$

□

Excluded Middle. In order to show Proposition 5.1, we need to further derive:

$$(\infty_i \to G) \to (\ \tilde{\neg}\infty_i \to G) \to G \qquad (EM_i)$$

Although in general not provable in minimal logic, this form of excluded middle is in our case derivable due to the fact that G is relevant. The proof is, for $w :\ \forall_{n,m}.\, n < m \to f(n) = f(m) \to \bot$:

$$\dfrac{u_2' :\ \tilde{\neg}\infty_i \to G \qquad \dfrac{\dfrac{u_1' : \infty_i \to G \qquad v_2 : \infty_i}{G}\to^- \qquad w}{\dfrac{\bot}{\tilde{\neg}\infty_i}\to^+ v_2}\to^- }{\dfrac{G \qquad w}{\dfrac{\bot}{G}\to^+ w}\to^-}\to^-$$

Final Step: the Cut. We conclude, for the case of $(\neg\infty_0 \to \infty_1)$, with the derivation[3]:

[3] EM_i is as on page 73 and $\infty_0 \to G$ and $\infty_1 \to G$ are the corresponding instances $\infty_i \to G$ for $i = 0$ and $i = 1$.

$$\dfrac{EM_i \qquad \dfrac{\infty_0 \to G \qquad \dfrac{\dfrac{\dfrac{(\neg\infty_0 \to \infty_1) \qquad \text{u:A}}{\tilde{\neg}\infty_0 \to \infty_1}\to^- \qquad v:\ \tilde{\neg}\infty_0}{\infty_1}\to^-}{G}\to^+ v}{\tilde{\neg}\infty_0 \to G}\to^-}{\dfrac{\text{G}}{A \to G}\to^+ u}$$

or, as a λ-term,

$$\begin{aligned}
M_{(\neg\infty_0 \to \infty_1)} :=& \lambda f, A, u_\perp^{\infty_0 \to \perp}, n, u_{1,n}. \\
& u_\perp\ \lambda m, u_{0,m}.\ A\ (m \sqcup n)\ (u_{0,m}\ (m \sqcup n)\ P(m))\ (u_{1,n}\ (m \sqcup n)\ P(n)) \\
M_{(\infty_i \to G)} :=& \lambda f, v_2^{(\infty_i)}, w.\ v_2\ 0\ \lambda k, v^{0 \le k}, w_2^{fk=i}.\ v_2\ \mathsf{S}k \\
& \lambda n, u^{\mathsf{S}k \le n}, w_1^{fn=i}.\ (w\ k\ n\ (Lem_{\mathsf{S}LeToLt} k\, n\, u)) \\
& (Lem_{=Trans} f\, 0\, k\, n\, w_2\, w_1), i \in \{0,1\} \\
M_1 :=& \lambda f, A. \\
& (\lambda EM_0.\ EM_0\ M_{\infty_0 \to G}\ \lambda u_\perp.\ M_{\infty_1 \to G}\, M_{(\neg\infty_0 \to \infty_1)}\, A\, u_\perp) \\
& \lambda u_1', u_2', w.\ u_2'\ (\lambda u_0^{\infty_0}.\ u_1'\, u_0\, w)\, w,
\end{aligned}$$

where the variables have the typing as in the proofs and derivation trees above.
Here, EM_0 is (EM_i) with $i := 0$ and the last line corresponds to its proof.

The proof using $(\neg\infty_1 \to \infty_0)$ is symmetrical.

Remark 5.2. *Another way to exploit the symmetry is by reformulating of A to be*

$$A^s : \forall_n.\ (f(n) = 1 \to \bot) \to (f(n) = 0 \to \bot) \to \bot \tag{5.6}$$

Consequently, if the assumption is A^s, then in the proof of $(\neg\infty_0 \to \infty_1)$ we need to first use $\tilde{\neg}\infty_0$ and then $\tilde{\neg}\infty_1$. Likewise, we can manipulate the proof involving $(\neg\infty_1 \to \infty_0)$ to use A^s. This will result in yet another pair of symmetrical programs - symmetric with respect to each other, but also correspondingly symmetric with respect to the programs associated with our first pair of proofs.

5.1.2 The Logical Falsity $\bot$

The reader might have noticed that, whereas for $(\infty_0\ \tilde{\vee}\ \infty_1)$ we have allowed to rewrite $\tilde{\vee}$ into "variants", we have been avoiding to do so for A and did not consider for example $A^{np} : \forall_n.\ \tilde{\neg} f(n) = 0 \to f(n) = 1$. Although

the proof of $(\neg\infty_0 \to \infty_1)$ can be carried out with this version, A^{np} is not a definite formula. In fact, "=" turns out to be an L-critical predicate symbol in A^{np}, in the sense of Definition 8.1 from Section 8.2. Thus, the double negation in the conclusion is needed in order to turn A^{np} into a definite formula. This is exactly our chosen A.

Another option to express the hypothesis A is by the more natural version

$$A(2) : \forall_n f(n) < 2, \tag{5.7}$$

which is a definite formula. From $A(2)$ we can infer the properties of A that we need in the proof by case distinction on the value of $f(n)$. Using $A(2)$ would allow us to generalize the lemma to r values, i.e., to $A(r) : \forall_n f(n) < r$ (see Section 5.2).

However, as we will further see in Section 5.2, the proof cannot be carried out in minimal logic if we have $A(r)$, so the double negation still needs to be inserted explicitely. Moreover, if we would use $A(2)$, we would not be able to exploit the symmetry anymore, since the weak disjunction $\tilde{\vee}$ is not explicit.

5.1.3 The Extracted Program

The A-Translation of the non-constructive proof

In order to apply the refined A-Translation we have to ensure first that A is a definite formula and that G's kernel consists of goal subformulas. The latter is easy to see, since both subformulas of G ($n < m$ and $f(n) = f(m)$ are atomic.

The analysis for A is immediate. We let A_i denote $f(n) = i$.

$$\begin{array}{rrrrl} (A_0 \to \bot) & \to & (A_1 \to \bot) & \to & \bot \in \mathcal{D} \quad \text{iff} \\ (A_0 \to \bot) & \in \mathcal{G}, & (A_1 \to \bot) \to \bot & \in \mathcal{D} & \quad \text{iff} \\ A_0 & \in \mathcal{D}, & (A_1 \to \bot) & \in \mathcal{G} & \quad \text{iff} \\ & & A_1 & \in \mathcal{D} & \end{array}$$

Since $f(n) = 0$, $f(n) = 1$ are atomic formulas, therefore definite, and $\bot$ is both goal and definite, it follows that A is a (relevant) definite formula. From the previous section we have:

Remark 5.3. *$A^{np} : \forall_n.\ \neg A_0 \to A_1$ is not a definite formula because it is not in $\mathcal{R}_D$ (not relevant), nor is $\neg A_0$ in $\mathcal{I}_G$ (an irrelevant goal formula).*

A and G are thus in the proper classes, so we can apply the refined A-Translation.

Step 1: Eliminating logical falsity. Based on Lemma 3.6, we investigate how are the following derivations for the definite formula A and goal formula G produced:

$$\vdash A^F \to A \text{ and } \vdash G \to (G^F \to \bot) \to \bot$$

- We need to derive $A^{\mathrm{F}} \to A$, i.e.,

  $$(\forall_n.\ (A_0 \to \mathrm{F}) \to (A_1 \to \mathrm{F}) \to \mathrm{F}) \to \forall_n.\ (A_0 \to \bot) \to (A_1 \to \bot) \to \bot.$$

  We denote by Cases_A Lemma 3.4 (Case Distinction). The derivation $M_{A^F to A}$ is presented below, with $v_i : \neg A_i$ and $w_i : \tilde{\neg} A_i$.

$$\dfrac{\dfrac{\mathsf{efq} : \mathrm{F} \to \bot \qquad \dfrac{\dfrac{u : A^F}{A^F\ n}\forall^- n \qquad v_0 \qquad v_1}{\mathrm{F}}\to^-}{\bot}\to^- \qquad \dfrac{\mathrm{Cases}_{A_0} \qquad \dfrac{\dfrac{\bot}{\neg A_0 \to \bot}\to^+ v_0 \qquad w_0}{\bot}\to^- \qquad \dfrac{\mathrm{Cases}_{A_1} \qquad \dfrac{\dfrac{\bot}{\neg A_1 \to \bot}\to^+ v_1 \qquad w_1}{\bot}\to^-}{\bot}\to^-}{\bot}\to^-}{A}\to^+ w_0, w_1, \forall^+ n$$

  As a λ-term, the above derivation is:

  $$\begin{aligned} M_{A^{\mathrm{F}} \to A} =& \lambda u.\ \lambda n, w_0, w_1. \\ & Cases_{A_1}\ (Cases_{A_0}(\lambda v_0.\ \mathsf{efq}\, u\, n\, v_0\, v_1) w_0)\ w_1 \end{aligned}$$

- For $G \to (G^F \to \bot) \to \bot$, since both conjuncts of G are atomic and without $\bot$, there is nothing to prove.

Step 2: Friedman's "trick" For the remaining occurrences of $\bot$, we use Friedman's substitution of $\bot$ by the strong version of G and thus have no more free predicate variables occurring in the proof. We can associate a program by the modified realizability using Lemma 3.7 and Lemma 3.8, as illustrated in the following section.

Associated Realizers

The realizer s_D from Lemma 3.8 is obtained by interpreting the proof in the Step 1 above. The Case Distinction Lemma used to transform A to A^F

extracts to

$$\lambda x_1 \lambda x_2.\ \text{if } t_A \text{ then } x_1 \text{ else } x_2,$$

where t_A represents a term s.t. $A \leftrightarrow t_A$ is provable. This amounts to using the if-operator with the test value $f(n) = 0$ and $f(n) = 1$, corresponding to A_0 and A_1, since

$$[\![M_{A^{\mathrm{F}} \to A}]\!] = \lambda n, x_{w_0}, x_{w_1}.\ [\![Cases_{A_1}]\!]\ ([\![Cases_{A_0}]\!] x_{\mathrm{dummy}}\ x_{w_0})\ x_{w_1},$$

where x_{dummy} is associated to efq.

Since no transformation was necessary in order to replace G by G^F, the associated realizer (s_G in Lemma 3.8) is the identity.

By coupling these with the realizer associated with the initial proof using Lemma 3.7 we obtain after normalization the programs $P1$ and $P2$ from Figure 5.1, corresponding to the use of $(\neg\infty_0 \to \infty_1)$ (to the right) and $(\neg\infty_1 \to \infty_0)$ (to the left). If we use A^s we obtain the pair of symmetrical versions of programs depicted in Figure 5.2, which in fact mirror the first two variants (while also swapping 0 and 1).

The symmetry is not easy to read from the structure of the extracted programs, but a careful analysis reveals that if we consider a sequence s and its complement $\overline{s}$, then we have: $P1(\overline{s}) = \overline{P2(s)}$. However, each of the two programs is asymmetric, i.e., $Pi(\overline{s}) \neq \overline{Pi(s)}, i \in \{1, 2\}$. This asymmetry characterizes the classical proof and is a consequence of the interpretation given to the weak existential quantifier. Since the refined A-Translation preserves the structure of the proof, the assymetry is present in the extracted programs. We will analyze in the following section various runs of the extracted programs, in order to illustrate these claims.

Experiments

We compare the first 2 versions of the programs on some systematic runs for sequences of length 4. These examples are depicted in Figure 5.3 and a generalization is presented in Section 5.2. On the one hand, they confirm the symmetry that we expected to have between the two programs. This can be seen for instance in the output of $P1$ for (0 1 0 0) and $P2$ for its complement (1 0 1 1) or by comparing $P1$(1 0 1 0) and $P2$(0 1 0 1).

On the other hand, the runs of the same program on these sequences show that $Pi(\overline{s}) \neq \overline{Pi(s)}, i \in \{0, 1\}$. This is the consequence of the fact that in the proof we have fixed the value $i \in \{0, 1\}$ for which the search begins. One can depict the programs as binary trees and observe that if the first index in the output pair p is such that $s(p_1) = i$ and $s(\mathsf{S}(p_1)) = 1 - i$, then

```
P1:=
[f]
 [if (f 0=1)
   [if (f 1=1)
     (0@1)
     [if (f 1=0)
       [if (f 2=1)
         (0@2)
         [if (f 2=0)
           (1@2)
           (0@0)]]
       (0@0)]]
   [if (f 0=0)
     [if (f 1=1)
       [if (f 2=1)
         (1@2)
         [if (f 2=0)
           [if (f 3=1)
             (1@3)
             [if (f 3=0)
               (2@3)
               (0@0)]]
           (0@0)]]
       (0@[if (f1=0) 1 0])]
     (0@0)]]
```

```
P2:=
[f]
 [if (f 0=1)
   [if (f 1=1)
     (0@1)
     [if (f 1=0)
       [if (f 2=1)
         [if (f 3=1)
           (2@3)
           [if (f 3=0)
             (1@3)
             (0@0)]]
         [if (f 2=0)
           (1@2)
           (0@0)]]
       (0@0)]]
   [if (f 0=0)
     [if (f 1=1)
       [if (f 2=1)
         (1@2)
         (0@[if (f 2=0) 2 0])]
       (0@[if (f 1=0) 1 0])]
     (0@0)]]
```

Figure 5.1: First pair of programs associated to Stolzenberg's Principle

```
P3:=
[f]
 [if (f 0=0)
   [if (f 1=0)
    (0@1)
    [if (f 1=1)
      [if (f 2=0)
        [if (f 3=0)
          (2@3)
          [if (f 3=1)
            (1@3)
            (0@0)]]
        [if (f 2=1)
          (1@2)
          (0@0)]]
      (0@0)]]
   [if (f 0=1)
     [if (f 1=0)
       [if (f 2=0)
         (1@2)
         (0@[if (f 2=1) 2 0])]
       (0@[if (f 1=1) 1 0])]
     (0@0)]]
```

```
P4:=
[f]
 [if (f 0=0)
   [if (f 1=0)
     (0@1)
     [if (f 1=1)
       [if (f 2=0)
         (0@2)
         [if (f 2=1)
           (1@2)
           (0@0)]]
       (0@0)]]
   [if (f 0=1)
     [if (f 1=0)
       [if (f 2=0)
         (1@2)
         [if (f 2=1)
           [if (f 3=0)
             (1@3)
             [if (f 3=1)
               (2@3)
               (0@0)]]
           (0@0)]]
       (0@[if (f 1=1) 1 0])]
     (0@0)]]
```

Figure 5.2: Second pair of programs associated to Stolzenberg's Principle

this index is abandoned. In other words, the program returns the pair p only if both of the following conditions are met

- p_1 is the smallest index in the sequence
- $1 - i$ does not appear between s_{p_1} and s_{p_2}

As a side effect, we remark that it does not suffice to investigate a sequence of length 3, as it would be the case with the naive search. For the programs to output correct values in all cases, we need to have at least 4 input values.

Test sequence	1^{st} variant	2^{nd} variant
(0 0 0 0)	(0 . 1)	(0 . 1)
(1 0 0 0)	(1 . 2)	(1 . 2)
(0 1 0 0)	(2 . 3)	(0 . 2)
(1 1 0 0)	(0 . 1)	(0 . 1)
(0 0 1 0)	(0 . 1)	(0 . 1)
(1 0 1 0)	(0 . 2)	(1 . 3)
(0 1 1 0)	(1 . 2)	(1 . 2)
(1 1 1 0)	(0 . 1)	(0 . 1)
(0 0 0 1)	(0 . 1)	(0 . 1)
(1 0 0 1)	(1 . 2)	(1 . 2)
(0 1 0 1)	(1 . 3)	(0 . 2)
(1 1 0 1)	(0 . 1)	(0 . 1)
(0 0 1 1)	(0 . 1)	(0 . 1)
(1 0 1 1)	(0 . 2)	(2 . 3)
(0 1 1 1)	(1 . 2)	(1 . 2)
(1 1 1 1)	(0 . 1)	(0 . 1)

Figure 5.3: Results of the different program variants extracted from the Corollary of Stolzenberg's principle

5.1.4 Related Work

The two pairs of proofs that we have presented are symmetrical to each other, reflecting the variants in which $\tilde{\vee}$ has been rewritten. Consequently, we were able to extract the same programs as (Urban, 2000), which uses the cut elimination strategy as a transformation method, in order to associate programs to classical proofs. However, the variants have been obtained in

(Urban, 2000) by manipulating the cut elimination rather than altering the proof. The key idea is to direct the normalization procedure to the left or to the right, by strategically assigning colors to the formulas. This gives therefore more freedom in how programs are associated to a fixed proof and in Urban and Ratiu (2006) we went a step further and conjectured that in this way one can extract more programs than by the double negation translation. As the example of the infinite boolean tape shows, it is however possible at least in some cases to obtain the same variety of associated programs by considering all possible interpretations of the weak connectors. More precisely, whereas it is true that by the (refined) A-Translation one can only associate *one* program to a proof, the interpretation given to the weak disjunction allows for flexibility in the formulation and consequently in the proof of the theorem.

It would be interesting to investigate the correlation between the colored cut-elimination procedure proposed in (Urban, 2000) and the interpretation given to weak connectors, combined with the double negation. It seems reasonable that a suitable strategy could be developed, in order to consider all possible variants of rewriting the weak connectors of the formula to prove. Further, one could investigate whether this corresponds to the coloring from (Urban, 2000).

Another line of research in which Stolzenberg's Principle has served the role of a relevant case study is the composition of the refined A-Translation and bar recursion. (Seisenberger, 2003) illustrates the interaction between the refined A-Translation and external realizers, in this case computed by the bar recursion, by considering a proof using the axiom of dependent choice. (Seisenberger, 2003) has shown a generalization of Stolzenberg's principle to

$$\forall_{f_A} \tilde{\exists}_a^{\mathsf{B}} \tilde{\exists}_{f \in \mathcal{S}} \forall_k f_A(f(\mathsf{S}k)) = a, \quad (5.2)$$

i.e., the claim that a *finite* number of equally colored elements exist in a two-colored sequence.

Thus, (Seisenberger, 2003) considers a different proof than the one that we have presented, but uses the refined A-Translation as well, in order to interpret it. $\tilde{\exists}$ is seen also as an abbreviation and the same interpretation is given to the logical negation ($\bot$). Although (Seisenberger, 2003) considers the generalization of Stolzenberg's Principle and uses dependent choice in the proof, the same behavior can be observed in the extracted program. This consists in selecting the output subsequence s' from the input sequence s such that the color of interest ($\neg a$ in (5.2)) does not occur in s between the first and the last index of s'. In the following section, we go a step further

in the direction of generalizing the principle, and allow that sequence is "colored" by an arbitrary number of values. We will point out that the same phenomenon occurs in the extracted program.

5.2 Generalization: IPH

In the following, we consider a generalization of Stolzenberg's Principle, in which the sequence is "colored by r colors", i.e., the tape takes values from $\{0, ..., r-1\}$. This more general case in known in the literature as the "Infinite Pigeonhole Principle" (IPH). We present in the section the results from (Ratiu and Trifonov, 2010).

IPH states that any finitely colored infinite sequence has an infinite monochromatic subsequence. In general, there is no computable functional producing such an infinite subsequence. However, IPH can be used to give a simple classical proof of the Π^0_2-statement that a finite monochromatic subsequence of any given length exists. For this reason, we will transform this corollary by the refined A-Translation and then use the modified realizability to produce a computable functional.

5.2.1 IPH - Statement and Proof

As with Stolzenberg's Principle, IPH can be stated in various forms. For instance:

Proposition 5.4. *For $r \in \mathtt{N}$ and $R_i \subset \mathtt{N}, i \in \{1, ..., r\}$ disjoint sets,*

$$\textit{if } R := \bigcup_{i=1}^{r} R_i \textit{ is infinite then } \exists q \leq r \textit{ s.t. } R_q \textit{ infinite.}$$

We work with the following statement:

Lemma 5.1 (Infinite Pigeonhole Principle)**.** *Any infinite sequence that is colored with finitely many colors has an infinite monochromatic subsequence.*

Proof. Let $f^{\mathtt{N}\Rightarrow\mathtt{N}}$ encode the infinite sequence colored by r colors, i.e. we make the assumption

$$\forall_n f(n) < r.$$

We need to show that infinitely many positions on f are colored by some color q:

$$G : \tilde{\exists}_q \forall_n \tilde{\exists}_k.\ n \leq k \mathbin{\tilde{\wedge}} f(k) = q.$$

We prove

$$\forall_f.\, \forall_n f(n) < r \to \tilde{\exists}_q \forall_n \tilde{\exists}_k.\, n \leq k \mathbin{\tilde{\wedge}} f(k) = q \tag{IPH}$$

by induction on the number of colors r.

Base case If $r = 0$, then the assumption $f(n) < r$ is false, so we use efq_G.

Step Assume that the statement to prove is true for r, i.e.,

$$\forall_f.\, \forall_n f(n) < r \to \tilde{\exists}_q \forall_n \tilde{\exists}_k.\, n \leq k \mathbin{\tilde{\wedge}} f(k) = q \tag{IH}$$

Let us fix an arbitrary f and assume

$$\forall_n\, f(n) < \mathsf{S}r \tag{StepH}$$

$$\forall_q.\, (\forall_n\, \tilde{\exists}_k.\, n \leq k \mathbin{\tilde{\wedge}} f(k) = q) \to \bot, \tag{NegG}$$

where (NegG) results from unfolding the weak existence $\tilde{\exists}$. It remains to derive a contradiction ($\bot$) from the above.

We make a case distinction as to whether there is an infinite subsequence of color r or not. If r appears infinitely often in the given sequence, i.e., if

$$\forall_n\, \tilde{\exists}_k.\, n \leq k \mathbin{\tilde{\wedge}} f(k) = r, \tag{Inf}$$

then this trivially leads to a contradiction, if we take q to be r in (NegG).

If this is not the case, then

$$\tilde{\exists}_n\, \forall_k.\ \ n \leq k \to f(k) = r \to \bot, \tag{NegInf}$$

which means that there exists an index n', from which on the color r does not appear anymore in the sequence. By (StepH) this means that from n' on we have only r distinct colors. We therefore look at a variant of the sequence f, in which we overwrite the positions up to n' by $f(n')$. Formally, this is the sequence

$$f' := \lambda n f\, (n \sqcup n'),$$

where $(x \sqcup y)$ denotes the maximum of x and y. The (IH) on f', colored by r colors, provides a color q for which we can find an infinite monochromatic subsequence:

$$\forall_n\, \tilde{\exists}_k.\, n \leq k \mathbin{\tilde{\wedge}} f\,(k \sqcup n') = q,$$

which contradicts (NegG). □

Formalization. We give in the following the λ-terms expressing the informal proof as presented above. For reasons that will be explained in Section 5.2.2 and further in Chapter 8, we consider the double negation of the assumption. Let

$$\begin{aligned} A(f,r) &:= \forall_n.\ (fn < r \to \bot) \to \bot \\ K(f,q,n) &:= \forall_k.\ n \le k \to f(k) = q \to \bot \\ G(f,q) &:= \forall_n.\ K(f,q,n) \to \bot \quad \text{I.e., } \forall_n \tilde{\exists}_k.\ n \le k \mathbin{\tilde{\wedge}} f(k) = q, \end{aligned}$$

where we have expanded all $\tilde{\exists}$ according to the definition in Section 2.2.

The Infinite Pigeonhole Principle can be now formulated as:

$$\forall_f.\ A(f,r) \to (\forall_q.\ G(f,q) \to \bot) \to \bot. \tag{IPH_r}$$

We show $\forall_r \mathrm{IPH}(r)$ by

$$M_{\mathrm{IPH}} := \lambda r.\ \mathsf{Ind}^{\mathrm{IPH}(r)}\, r M_{Base} M_{Step},$$

where

$$\begin{aligned} M_{Base} &:= \lambda f \lambda v^{A(f,0)} \lambda w^{\forall_q \neg G(f,q)}.\ v\, 0\, \mathsf{efq} \\ M_{Step} &:= \lambda r \lambda u^{IH} \lambda f \lambda v^{A(f,r)} \lambda w^{\forall_q \neg G(f,q)}.\ w\, r M_{G(f,r)} \\ M_{G(f,r)} &:= \lambda n \lambda w_1^{K(f,r,n)}.\ u\ (\lambda q.\ f(n \sqcup q))\, M_{A(f,r)}\, M_{\forall_q \neg G(f,q)} \\ M_{A(f,r)} &:= \lambda n_1 \lambda v_1^{f(n \sqcup n_1) < r \to \bot}.\ v(n \sqcup n_1) M_{\bot Cases} \\ M_{\bot Cases} &:= \lambda v_2^{f(n \sqcup n_1) < \mathsf{S} r}.\ Lem_{< \mathsf{S} Cases}\, r\, f(n \sqcup n_1) v_2 \\ &\qquad\qquad v_1\, (w_1\, (n \sqcup n_1)\, (P(n)\, n\, n_1)) \\ M_{\forall_q \neg G(f,q)} &:= \lambda q \lambda w_2^{\forall_{n_1} \tilde{\exists}_k.\, n_1 \le k \mathbin{\tilde{\wedge}} f(n \sqcup k) = q}.\ w\, q\, M_{G(f,q)} \\ M_{G(f,q)} &:= \lambda n_1 \lambda w_3^{K(f,q,n_1)}.\ w_2\ n_1\ M_{K(f,q,(n \sqcup k))} \\ M_{K(f,q,(n \sqcup k))} &:= \lambda k \lambda u_1^{n_1 \le k} \lambda u_2^{f(n \sqcup k) = q}.\ w_3\, (n \sqcup k)\, L\, u_2 \\ L &:= Lem_{\le Trans}\, n_1\, k\, (n \sqcup k)\, u_1\, (P(k)\, n\, k), \end{aligned}$$

with the types for the intermediate terms written respectively as subscripts.

As in the case of 2 colors, we assumed here that the following additional lemmas are already available:

$$\begin{aligned} Lem_{< \mathsf{S} Cases} &: \forall_{n_1,n_2}.\ n_2 < \mathsf{S} n_1 \to (n_2 < n_1 \to \bot) \to (n_2 = n_1 \to \bot) \to \bot \\ Lem_{\le Trans} &: \forall_{n_1,n_2,n_3}.\ n_1 \le n_2 \to n_2 \le n_3 \to n_1 \le n_3 \\ P(s) &: \forall_{n_1,n_2}.\ s \le (n_1 \sqcup n_2), \text{ with } s \text{ either } n_1 \text{ or } n_2. \end{aligned}$$

5.2.2 Double Negations

In the Step Case, we assume that (NegInf) holds, i.e., that from an n onwards the color r does not appear anymore in the sequence. We use the argument to show that all subsequent values must be strictly smaller than r, since by (StepH) we only have colors up to (including) r. This amounts however to deriving something positive (i.e., "all values in f' are $< r$") from a negation ("it is not possible that the color r appears in the sequence f'"). Since (NegInf) contains a *logical* falsity this derivation cannot be carried out in the minimal logic setting from which $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$ are excluded. Therefore, we need to change (StepH), so that it contains logical negations as well, and thus we double negate A and weaken it to

$$\forall_n \ \tilde{\neg}\,\tilde{\neg} f(n) < r.$$

As it turns out, this double negation suffices in order to carry out the proof in minimal logic and apply the refined A-Translation. This is the least price to pay in order to recover the computational content, since no further negations need to be inserted, as it would be the case with the double negation translation.

The insertion of negations might seem rather ad-hoc. Yet, in Chapter 8 we present a systematic way by which we identify which double negations are needed in order to avoid $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$. For this, we will define a class of "critical atoms" which are the only ones that need to be double negated. This is the case with $f(n) < r$. The example presented here, together with the ones in the following two chapters have served as a basis for developing the detection mechanism introduced in Chapter 8.

5.2.3 The Π^0_2-Corollary

As we have already mentioned, we cannot expect to extract by A-Translation a computable functional from Lemma 5.1. This is due to the fact that IPH cannot be stated as a Π^0_2-formula. Therefore, we analyze the following corollary instead:

Corollary 5.0.2. *In any infinite sequence f colored with a finite number of colors and for any $n \in \mathtt{N}$, one can find a finite monochromatic subsequence of f of length n.*

Proof. This follows trivially from Lemma 5.1, by taking the first n elements of the infinite monochromatic subsequence. However, since we are interested in the way this sequence is produced, we will analyze the formalization in

detail. We break the proof in two steps. First, we show how the sequence of a given length n is obtained from an infinite monochromatic sequence. We cut then this result with the IPH to get the finite monochromatic subsequence from the given infinite sequence.

In the formalization, we store in a list l the indices at which the given sequence f is of some color q. We will require that the list is strictly decreasing[4], to ensure that we get distinct indices. Let $G(f, n, l, q)$ gather these specifications for the list l[5]:

$$\tilde{\exists}_l.\ |l| = n \mathbin{\tilde{\wedge}} \forall_m(\mathsf{S}m < n \to l_{\mathsf{S}m} < l_m) \mathbin{\tilde{\wedge}} \forall_m(m < n \to f(l_m) = q) \qquad (G(f, n, l, q))$$

We show that for any color q,

$$\begin{array}{lll} \text{if} & \forall_n\ \tilde{\exists}_k.\ n \le k \mathbin{\tilde{\wedge}} f(k) = q, & \text{(Step1H)} \\ \text{then} & \forall_n G(f, n, l, q) & \text{(Step1G)} \end{array}$$

and construct the list by induction on its length n.

For $n = 0$ our only choice is nil.

For $n = 1$, we take a list containing a single element, obtained from (Step1H) applied at 0.

Finally, for the step case, assume that we have already a list l with $n > 0$ elements. (Step1H) on $\mathsf{S}(l_0)$ gives a fresh element $k > l_0$, which we add at the beginning of l.

With this, we can infer that for any sequence f and number of colors r:

$$\forall_n((f(n) < r \to \bot) \to \bot) \to \forall_n G(f, n, l, q)$$

using IPH. □

Formalization. We make the following notations:

$$\begin{array}{rcl} G_2(l, n) & := & \forall_m.\ \mathsf{S}m < n \to l_{\mathsf{S}m} < l_m \\ G_3(l, n, q) & := & \forall_m.\ m < n \to f(l_m) = q \\ \tilde{G}(q, n) & := & \forall_l.\ |l| = n \to G_2(l, n) \to G_3(l, n, q) \to \bot \end{array}$$

Step 1. We derive $\forall_{f,q}.\ G(f, q) \to \forall_n.\ \tilde{G}(q, n) \to \bot$ (with $G(f, q)$ as defined on page 84) by

$$M_{auxL} := \lambda f \lambda q \lambda w^{G(f,q)} \lambda n.\ \mathsf{Ind}^{\neg\tilde{G}(q,n)}\, n\, M_{base} M_{step},$$

[4]The list will be strictly decreasing rather than increasing, to avoid using "append" when constructing it.

[5]We use the notations from 2.3.

where

$$M_{base} := \lambda u_0^{\tilde{G}(q,0)}.\ u_0\ \mathsf{nil}\ \mathsf{AxT}\ (\lambda m.\ \mathsf{efq})\ (\lambda m.\ \mathsf{efq})$$
$$M_{step} := \lambda n.\ \mathsf{Ind}^{\neg\tilde{G}(q,n)\to\neg\tilde{G}(q,\mathsf{S}n)}\, n\, M_{n=0} M_{\mathsf{S}n}$$
$$M_{n=0} := \lambda u_1^{\tilde{G}(q,0)\to\bot} \lambda u_2^{\tilde{G}(q,1)}.\ u_1(\lambda l\, \lambda u_{1,1}^{|\mathsf{nil}|=0} \lambda u_{1,2}^{G_2(\mathsf{nil},0)} \lambda u_{1,3}^{G_3(\mathsf{nil},0,q)}.\ w\, 0\, M_{K(f,q,0)})$$
$$M_{K(f,q,0)} := \lambda k \lambda v_1^{0\le k} \lambda w_1^{f(k)=q}.\ u_2\ (k{:})\ \mathsf{AxT}\ (\lambda m.\ \mathsf{efq})\ (\lambda m.\ \mathsf{Ind}\, m\, (\lambda u^{0<1} w_1)(\lambda m.\ \mathsf{efq}))$$
$$M_{\mathsf{S}n} := \lambda n \lambda u_1^{\tilde{G}(q,n)\to\bot} \lambda u_2^{\tilde{G}(q,\mathsf{S}n)}.\ u_1(\lambda l_2.\ \mathsf{Ind}^{\tilde{G}(q,n)}\, l_2\, \mathsf{efq}\, M_{n_1::l})$$
$$M_{n_1::l} := \lambda n_1 \lambda l \lambda u_{1,1}^{|l|=n} \lambda u_{1,2}^{G_2(n_1::l,n)} \lambda u_{1,3}^{G_3(n_1::l,n,q)}.\ w\ (\mathsf{S}n_1)\, M_{K(f,q,\mathsf{S}n_1)}$$
$$M_{K(f,q,\mathsf{S}n_1)} := \lambda k_1 \lambda v_1^{\mathsf{S}n_1\le k_1} \lambda w_2^{f(k)_1=q}.\ u_2\ (k_1::n_1::l)\ u_{1,1}\ M_{G_2} M_{G_3}$$
$$M_{G_2} := \lambda m.\ \mathsf{Ind}^{G_2(k_1::n_1::l,\mathsf{S}n)}\, m\, (\lambda u.\ Lem_4\, n_1\, k_1\, v_1)\ (\lambda m \lambda v_2^{\mathsf{S}m<\mathsf{S}n}.\ u_{1,2}\, m\, v_2)$$
$$M_{G_3} := \lambda m.\ \mathsf{Ind}^{G_3(k_1::n_1::l,\mathsf{S}n,q)}\, m\, (\lambda u\ w_2)\ (\lambda m \lambda v_3^{\mathsf{S}m<\mathsf{S}(\mathsf{S}n)}.\ u_{1,3}\, m\, v_3)$$

In the above we have used $Lem_4 : \forall_{n_1,n_2}.\ \mathsf{S}n_1 \le n_2 \to n_1 < n_2$.

Step 2. The corollary now follows from IPH and Step 1:

$$M_{\mathrm{IPHcor}} := \lambda f \lambda r \lambda v^{A(f,r)} \lambda n \lambda u^{\forall_q \tilde{G}(q,n)}.\ M_{\mathrm{IPH}}\, r\, f\, v\, (\lambda q \lambda w^{G(f,q)}.\ M_{auxL}\, f\, q\, w\, n(uq))$$

Remark 5.4. *It is straightforward to infer that if all elements are of color q, then each two are equal*

$$\forall_q.\ \forall_m(m < n \to f(l_m) = q) \to \forall_m(\mathsf{S}m < n \to f(l_m) = f(l_{\mathsf{S}m})). \qquad (5.8)$$

Since the color of the monochromatic subsequence is not relevant to us, we could thus consider for simplification the modified goal formula:

$$\tilde{\exists}_l.\ |l| = n\ \tilde{\wedge}\ \forall_m(\mathsf{S}m < n \to l_{\mathsf{S}m} < l_m)\ \tilde{\wedge}\ \forall_m(\mathsf{S}m < n \to f(l_m) = f(l_{\mathsf{S}m})) \qquad (G'(f,n,l))$$

5.2.4 Results with A-Translation

We first need to ensure that the specification conforms to our requirements. We have already observed that the Infinite Pigeonhole Principle (IPH) is clearly not a Π^0_2-formula. Since the A-Translation mechanism is not modular, i.e., cannot be applied to intermediate lemmas unless they are Π^0_2-formulas, we apply it to the proof of the corollary

$$\forall_{r,f}.\ \forall_n((f(n) < r \to \bot) \to \bot) \to \forall_n G(f,n,l,q), \qquad (\mathrm{IPHcor})$$

where the goal formula's kernel can be also taken to be the conjunction $(G'(f,n,l))$.

(IPHcor) is a Π^0_2-formula and its subformulas definite and goal:

- $D := \forall_n.\ (f(n) < r \to \bot) \to \bot$ is a relevant formula of the form $\forall_n D$, with $D = (D_0 \to G) \to R \in \mathcal{R}_D$.
- $\vec{G} := G(f, n, l, q)$ consists of irrelevant conjuncts only, each of which is a goal formula.

Therefore, we can apply the refined A-Translation to it.

In the first step of eliminating the irrelevant occurrences of $\bot$ by Lemma 3.6, we have $G^F := G$, so no transformation on the goal is needed.

D is, however, relevant and we need to derive (3.2): $D^F \to D$. We follow the proof of Lemma 3.6 and break it down into steps

- first, we observe that we are in the case $\forall_n D'$ with $D' = \tilde{\neg}\,\tilde{\neg} f(n) < r$. Thus, it suffices to show $D'^F \to D'$;
- elaborating, D' is a relevant formula of the form $G \to R$, with $G = f(n) < r \to \bot$ and $R = \bot$;
- thus, it suffices to have the derivation $\tilde{\neg}\neg(G \to R)^F \to G \to R$;
- we use the (IH) for G and R.
  - Since $R = \bot$, deriving $\tilde{\neg}\neg R^F \to R = ((\mathrm{F} \to \mathrm{F}) \to \bot) \to \bot$ is trivial.
  - For G, by (3.4) we have $G \to (G^F \to \bot) \to \bot$, i.e.,

    $$(f(n) < r \to \bot) \to (f(n) < r \to \mathrm{F} \to \bot) \to \bot.$$

    This later property is given by Lemma 3.4 (Case Distinction).

The essential observation in the above sketch of the way $D^F \to D$ is inferred, is to notice that such proofs usually amount to using case distinction. As it was the case with the boolean sequence, this will be reflected in the extracted program, where we will need to investigate whether $f(n) < r$.

The next step in transforming our proof by the refined A-Translation consists in replacing throughout the proof the remaining occurrences of $\bot$ by the strong goal $G(f, n, l, q)$. The constructive counterpart

$$\forall_{f,r}.\ \forall_n((f(n) < r \to \mathrm{F}) \to \mathrm{F}) \to \forall_n G(f, n, l, q) \qquad \text{(IPHcor-}\exists\text{)}$$

of our initial (classical) proof is now provable by plugging together the above transformation on D and the derivation for M_{IPHcor}. In the proof M_{IPHcor}, we will expand M_{IPH}, such that its computational content is also exploited.

The proof is now ready to be interpreted by the modified realizability, in the way suggested by the Soundness Theorem 3.4 and Theorem 3.5. Thus, with the rules from Definition 3.10, we are able to automatically extract the following program[6]:

$$
\begin{aligned}
P_{\mathrm{IPHcor}} :=\ & \lambda f, r, n.\ \mathcal{R}_1\ r\ F_1\ G_1\ f \\
& (\lambda n_1, ln_1.\ [\mathsf{if}\ (f n_1 < r) ln_1 (\mathsf{nil}^{\mathtt{N}}, 0)]) \\
& (\lambda n_1, w.\ [if n (\mathsf{nil}^{\mathtt{N}}, n_1) (\lambda n_5.\ \mathcal{R}_3\, n_5\, F_3\, G_3\, \lambda l.\ (l, n_1))]) \text{ with} \\
F_1 :=\ & \lambda f', h, p.\ h\ 0\ (\mathsf{nil}^{\mathtt{N}}, 0) \\
G_1 :=\ & \lambda n_1, g, f', h, p.\ p\ n_1\ (\lambda n_2, u_1.\ g\ (\lambda n_3.\ f'\ (n_2 \sqcup n_3)) \\
& (\lambda n_3, ln_2.\ h\ (n_2 \sqcup n_3) (\mathcal{R}_2\, n_1\, F_2\, G_2\, f'\ (n_2 \sqcup n_3)\, ln_2\ (u_1\ (n_2 \sqcup n_3)))) \\
& (\lambda n_3, w.\ p\, n_3\ (\lambda n_5, u_2.\ w\, n_5\ (\lambda n_4.\ u_2\ (n_2 \sqcup n_4))))) \\
F_2 :=\ & \lambda n_5, ln_3, ln_4.\ [\mathsf{if}\ (n_5)\ ln_4\ \lambda n_6.\ ln_3] \\
G_2 :=\ & \lambda n_5, v, n_7, ln_5, ln_6.\ [\mathsf{if}\ (n_7)\ ln_5\ \lambda n_8.\ v\ n_8\, ln_5\, ln_6] \\
F_3 :=\ & \lambda L.\ w\, 0\, (\lambda n_8.\ L\, n_8\ :) \\
G_3 :=\ & \lambda n_9, H, L.\ H\ \lambda l'.\ [\mathsf{if}\ l' (\mathsf{nil}^{\mathtt{N}}, 0) \lambda n_3, l.\ w\ \mathsf{S} n_3\ (\lambda n_5.\ L\ (n_5 :: n_3 :: l))],
\end{aligned}
$$

where

$$
\begin{aligned}
& L^{\mathtt{LN}\to\mathtt{LN}\times\mathtt{N}}, H^{(\mathtt{LN}\to\mathtt{LN}\times\mathtt{N})\to\mathtt{LN}\times\mathtt{N}}, h^{\mathtt{N}\to\mathtt{LN}\times\mathtt{N}\to\mathtt{LN}\times\mathtt{N}}, p^{\mathtt{N}\to(\mathtt{N}\to(\mathtt{N}\to\mathtt{LN}\times\mathtt{N})\to\mathtt{LN}\times\mathtt{N})\to\mathtt{LN}\times\mathtt{N}} \\
& g^{(\mathtt{N}\to\mathtt{N})\to(\mathtt{N}\to\mathtt{LN}\times\mathtt{N}\to\mathtt{LN}\times\mathtt{N})\to(\mathtt{N}\to(\mathtt{N}\to(\mathtt{N}\to\mathtt{LN}\times\mathtt{N})\to\mathtt{LN}\times\mathtt{N})\to\mathtt{LN}\times\mathtt{N})\to\mathtt{LN}\times\mathtt{N}} \\
& u^{\mathtt{N}\to\mathtt{LN}\times\mathtt{N}}, v^{\mathtt{N}\to\mathtt{LN}\times\mathtt{N}\to\mathtt{LN}\times\mathtt{N}\to\mathtt{LN}\times\mathtt{N}}, w^{\mathtt{N}\to(\mathtt{N}\to\mathtt{LN}\times\mathtt{N})\to\mathtt{LN}\times\mathtt{N}}, ln^{\mathtt{LN}\times\mathtt{N}} \\
& \mathcal{R}_{1,\mathtt{N}} : (\mathtt{N}\to\mathtt{N})\to(\mathtt{N}\to\sigma\to\sigma)\to(\mathtt{N}\to(\mathtt{N}\to(\mathtt{N}\to\sigma)\to\sigma)\to\sigma)\to\sigma, \\
& \mathcal{R}_{2,\mathtt{N}} : \mathtt{N}\to\sigma\to\sigma\to\mathtt{LN}\times\mathtt{N},\ \mathcal{R}_{3,\mathtt{N}} : (\mathtt{LN}\to\sigma)\to\sigma,\ \text{with } \sigma := \mathtt{LN}\times\mathtt{N}
\end{aligned}
$$

Let us first observe that all terms are in tail recursion form, with the exception of G_1. This results from the inductive proof of IPH in which the proof M_{Step} the induction hypothesis is not applied first, but only at the level of $M_{G(f,r)}$. Since A-Translation preserves the structure of the proof, this is reflected in the resulting step function for the recursion operator $\mathcal{R}_1$.

In Figure 5.4 we present two extracted programs corresponding to the proofs of IPHcor and IPH in a more readable form, with suggestive variable names and separate definitions for functional arguments. In order to understand the program code better, it is helpful to consider Table 5.1, which summarizes the computational types associated with the relevant subformulas of the two statements.

[6]We follow closely the output of (Minlog), in which we have extracted the program from the refined A-translated proof of IPHcor.

Formula	Specification	Input	Output
IPH	IPH	$r, f, \mathsf{FC}, \mathsf{IMS}$	$\bot$
$\forall_n \ \tilde{\neg}\,\tilde{\neg} f(n) < r$	FC	$n, \bot$	$\bot$
$\forall_q \tilde{\exists}_n \forall_k.\ n \leq k \rightarrow \ \tilde{\neg}(f(k) = q)$	IMS	q, IS	$\bot$
$\forall_n \tilde{\exists}_k.\ n \leq k \ \tilde{\wedge}\ f(k) = q$	IS	n, SE	$\bot$
$\forall_k.\ n \leq k \rightarrow \ \tilde{\neg} f(k) = q$	SE	k	$\bot$
(Step1G)	FS	$n, (\bot \Rightarrow \bot)$	$\bot$
IPHcor-$\exists$	IPHcor	r, f, n	$\bot$

Table 5.1: A-Translation computational types

With $\bot$ being a predicate variable, we regard here its computational type as being *abstract*, i.e., a type variable that can be substituted with any type.[7] Since $\bot$ is to be substituted with the formula $\exists x\, G$, we can in fact think of the computational type of $\bot$ as the type of the *final result* x, which in our case is determined by Corollary 5.0.2 (IPHcor). Computationally relevant formulas always end in $\bot$, so they correspond to computations producing an output of the abstract type, i.e., a potential candidate for a final result, as shown in the "Output" column in Table 5.1. The abstract view of $\bot$ restricts the possible form of the extracted programs: they can either return a dummy value, or refer to some of their (functional) parameters, whose type end in $\bot$. Thus, the program adheres to the *continuation passing style (CPS)*. This way of computing the values to be placed in the "hole" corresponding to $\bot$ captures the functionality of the operators abort $\mathcal{A}$ and control $\mathcal{C}$ from Chapter 4.

In what follows, we analyze the separate computational components in more detail.

- The program FC (Finitely Colored) is given an index n and a candidate for the final result $\bot$, which is correct if the color of f at n is below r, according to the condition from our assumption.
- The program SE (Sequence Extension) has to produce the result given an index $k \geq n$, at which the sequence f is colored by q.
- The program IS (Infinite Sequence) takes an index n and is expected to provide a next occurrence k of the color q in f. Note that IS

[7] By abuse of notation we will denote here by $\bot$ the predicate variable, its computational type, and object variables of this type.

cannot return k directly, because this could amount to computing through an infinite sequence. Instead, it should return a final result by passing a candidate for k to its parameter SE (traditionally called a *continuation*), which verifies if k fits the requirement.

- Similarly, IMS (Infinite Monochromatic Sequence) takes a color q and computes a final result by invoking IS for some indices n. Note that IMS will not obtain the final result directly: in order to invoke IS it needs to be provided with a program SE, which specifies how to continue the computation after receiving an answer k with $f(k) = q$.
- The main program IPH can be used only after the continuations FC and IMS have been provided.
- The conclusion (Step1G) of (IPHcor) states that the final result should be a finite list l of indices. However, since we are using the weak existence, the program FS (Finite Sequence) would not directly return the final answer, but instead would feed the currently computed list of length n to its continuation parameter, named P in Figure 5.4b).
- The final program IPHcor is the algorithm that we expect to extract: given a sequence f, a number r of colors and the length n, it returns a finite list l of indices at which the sequence f has the same color. Note that the specification corresponds to the strong existence formula (IPHcor-$\exists$).

We will make use of the following two abbreviations:

$$(h\lceil n)(m) := h\,(n \sqcup m) \qquad (h\rceil n)(m) := (n \sqcup h(m)). \tag{5.9}$$

The program scheme IPH given in Figure 5.4a follows the proof structure of IPH very closely by recursing on r. For the base case, the use of efq in the proof is reflected by calling FC with a dummy candidate ($\Box$). The recursive case calls IMS in an attempt to build a monochromatic subsequence of color $r-1$. For this, the program IS is expected to calculate an index after n of this color and to pass it to its parameter SE. Following the proof, IS assumes that $r-1$ does not appear after n and attempts to obtain the final result by a recursive call to IPH, with the parameters f, FC and IMS changed to $f\lceil(n)$, FC_n and IMS_n respectively, thus disregarding indices smaller than n. FC_n deserves special attention: as noted above, it obtains an index n' and a result $\bot$, which will be correct if $(*)$ $(f\lceil(n))(n') < r-1$. When we know for a fact that this color differs from $r-1$, $(*)$ is equivalent to

$$
\begin{aligned}
\mathsf{IPH}(r, f, \mathsf{FC}, \mathsf{IMS}) &:= \textbf{if } r = 0 \textbf{ then } \mathsf{FC}(0, \Box) \textbf{ else } \mathsf{IMS}(r-1, \mathsf{IS}) \\
\mathsf{IS}(n, \mathsf{SE}) &:= \mathsf{IPH}(r-1, f\lceil(n), \mathsf{FC}_n, \mathsf{IMS}_n) \\
\mathsf{FC}_n(n', \bot) &:= \mathsf{FC}\big((n \sqcup n'), \textbf{if } f(n \sqcup n') \neq r-1 \textbf{ then } \bot \textbf{ else } \mathsf{SE}\ (n \sqcup n'))\big) \\
\mathsf{IMS}_n(q, \mathsf{IS}') &:= \mathsf{IMS}(q, \mathsf{IS}'_n), \\
\mathsf{IS}'_n(k, \mathsf{SE}) &:= \mathsf{IS}'(k, \mathsf{SE}\lceil n)
\end{aligned}
$$

a) A-Translation program for IPH

$$
\begin{aligned}
\mathsf{IPHcor}(r, f, n) &:= \mathsf{IPH}(r, f, \mathsf{FC}, \mathsf{IMS}) \\
\mathsf{FC}(n, \bot) &:= \textbf{if } f(n) < r \textbf{ then } \bot \textbf{ else } \Box, \\
\mathsf{IMS}(q, \mathsf{IS}) &:= \mathsf{FS}(n, \lambda l\, l) \\
\mathsf{FS}(n, \mathsf{P}) &:= \textbf{if } n = 0 \textbf{ then } \mathsf{nil} \textbf{ else} \\
&\qquad \textbf{if } n = 1 \textbf{ then } \mathsf{IS}\big(0, \lambda k.\ \mathsf{P}(k{:})\big) \textbf{ else } \mathsf{FS}(n-1, \mathsf{P}') \\
\mathsf{P}'(k :: l) &:= \mathsf{IS}\big(k+1, \lambda k'.\ \mathsf{P}(k' :: k :: l)\big)
\end{aligned}
$$

b) A-Translation program for IPHcor

Figure 5.4: A-Translation programs for IPH and IPHcor

$f\,(n \sqcup n') < r$, which allows us to pass $\bot$ to FC in this case. If the color happens to be $r-1$, this contradicts the assumption under which IPH was called recursively; however, we have found exactly an index that SE expects.

We now examine the main program given in Figure 5.4b. IPHcor can be viewed as a "search breakpoint" - it overwrites the requirement that an infinite subsequence exists, as soon as the specified number of elements, all of the same color, have been found. Since the proof is obtained by a cut, IPHcor is defined by a call to IPH with specially constructed parameters. The program FC, being extracted from the Cases Lemma 3.4, acts as a correctness guard: returns $\bot$ on $f(n) < r$ and a dummy value $\Box$ otherwise. FS corresponds to Step 1 of Corollary 5.0.2 and thus recursively extracts the values of constant color. The role of the function SE is played by P, which receives the next index of the infinite sequence and accumulates it in a list.

In order to understand the operational semantics of the obtained program, we should note that both recursions on r and n unfold immediately and the actual computation is carried out during the folding process, from the base case up. This has the effect that for every color $q < r$ a program FS is started, each of them calculating a list of indices of the corresponding

color. The program FS performs a "successful step" only when SE receives some index k of color q; in this case IS is invoked, asking for the index after $k+1$. The process ends when some list reaches length n, and it is returned as the final result. However, there is one important pitfall: the program IS, called after each successful step, restarts IPH from the base case. This invokes fresh programs FS for all colors below q, while the partially accumulated lists for these colors are lost. As a result, the program in Figure 5.4, which generalizes the case of a 2-colored sequence, need not necessarily find the first n occurrences of constant color; it returns a list of the smallest possible indices of a color q, *between which no color larger than q appears.*

Experiments

In the following we give the results for selected runs of the extracted program on uniformly distributed random sequences. The case studies have been presented also in (Ratiu and Trifonov, 2010), where we go a step further and compare the programs corresponding to the A-translated classical proof with the one obtained by applying the Dialectica method.

n	r	**last index**	*time (ms)*
10	2	**18**	< 1
100	2	**184**	48
200	2	**390**	133
500	2	**988**	835
10	3	**33**	2
10	4	**45**	6
50	4	**205**	289

Figure 5.5: Average runs of the extracted program

Complexity. In (Ratiu and Trifonov, 2010) we present a detailed complex analysis, which our colleague Trifon Trifonov has carried out, in order to compare the result of using Dialectica on classical proofs to the program which we have presented above. The outcome is that the worst time complexity is $O(n^r)$, while the average time complexity of IPHcor is $O(r^2 \cdot n)$. This relies on the characteristics that the programs extracted from the A-translated proofs have: the returned subsequence is such that it has color q and no color larger than q appears between its indices in the given sequence. This generalizes the boolean case, where we have already observed

that this amounts to inspecting sequences of length at least 4, as opposed to the constructive case, where 3 elements suffice in order to verify the claim. However, the average time complexity is the same as the worst (and the average) time complexity of the direct naïve algorithm, which goes linearly on the given sequence, constructing r lists until one of them reaches length n. Moreover, the A-translated proof results in a program of better average time complexity than the Dialectica-interpreted proof, while the worst time complexity is the same in both cases.

5.2.5 Related work

In (Ratiu and Trifonov, 2010) we also present the program extracted from IPHcor using Gödel's Dialectica interpretation. Our colleague Trifon Trifonov has carried out an extensive comparison between the two extraction methods and treats this aspect also in his PhD Thesis (Trifonov, 2011). The outcome is that each method has a certain advantage over the other. On the one hand, unlike A-Translation, Dialectica allows for a modular interpretation of classical proofs, since it does not restrict the extracted functionals to be computable and thus allows to interpret also formulas which are not Π^0_2, as it is the case with IPH. However, as a result of the fact that the refined A-Translation is not modular, the components associated the non-Π^0_2-lemmas used in the proofs are much tightly connected in the extracted program. As a consequence, A-Translation's immediate backtracking in the extracted program brings down its average time complexity, which is better than the average time complexity of the Dialectica program. The worst time complexity is the same for both algorithms.

Other solution using Dialectica interpretation are given in (Oliva and Escardo, 2010), where products of selection functions are used in order to simulate the game of producing an example and a counter-example for the statement. In (Oliva, 2006) a finite version of Spector's bar recursion is used instead, in order to realize a negative translation of the IPH.

5.3 Summary of the Chapter

This chapter presents the effects of the refined A-Translation on the example of the Infinite Pigeonhole Principle (IPH). We have started from the simplified case of a boolean tape in which we search for two identical values. The extracted programs are easier to analyze and we were able to observe that the same asymmetry as in (Seisenberger, 2003) is present. This is due to the continuation passing style which characterizes the way classical proofs

are interpreted - in our case by the A-Translation. Further, although A-translation does not allow flexibility in the way proofs are translated, as is the case with using cut-elimination in (Urban, 2000), in the boolean case we were able to obtain the same extracted programs by manipulating the way in which weak disjunction is rewritten.

We have generalized this to the full IPH and pointed out that applying the A-translation on the IPH is not possible, since the statement is not a Π_2^0-formula. For this reason, we have restricted out attention to a corollary. Its proof cannot be carried out in minimal logic unless we manually insert negations, but as Section 5.2.1 demonstrates, it suffices to double negate the assumptions. Therefore, at the cost of restricting the proof system to NA^ω for the refined A-Translation, we benefit from the fact that only these controlled double negations are necessary, as opposed to using the Gödel-Gentzen transformation as it is the case for the original A-Translation method. As a result, due to the computational relevance of $\bot$, the extracted programs are brushed-up of some irrelevant computations. It would be interesting to compare the programs obtained by the refined A-Translation with the corresponding programs resulting from using the original A-Translation method and we leave this as a subject for investigation in future work.

In this chapter, we have given special attention to the formalization of the Infinite Pigeonhole Principle, since this is mirrored in the extracted programs. We have also illustrated the steps necessary in order to transform the proof by the refined A-Translation and have interpreted the resulting constructive proof. We have analyzed the extracted program from IPH and pointed out the specific behavior resulting from using the refined A-Translation: the search for the subsequence of color q is performed in such a way that no color larger than q appears between its indices in the given sequence. We have also drawn the attention to the fact that the refined A-Translation proofs mimic the continuation passing style from programming and hide a use of the control and abort operators, due to the special role played by $\bot$.

Chapter 6

The Erdös-Szekeres Theorem

An interesting property in the realm of combinatorics, the Erdös-Szekeres Theorem (ES) states the existence of monotonically increasing/decreasing subsequences of length at least $\mathsf{S}n$ in a given sequence of n^2+1 elements. The theorem follows from the finite version of the Pigeonhole Principle (FPH) and is shown by an indirect proof, using also a clever construction.

The focus in this chapter lies on the conditions that ES and its proof need to fulfill in order to apply the refined A-Translation. We will emphasize the formalization decisions with accent on the use of negations. As we will see, the elaborated proof and all the necessary changes are reflected in rather complicated extracted programs, but the effort invested in the formalization is well-worth, since by this the correctness of the associated algorithm is guaranteed.

The purpose of this chapter is two-folded. On the one hand, we have investigated the effects of the refined A-Translation on the extracted programs and on the other we were confronted with the issue of having to insert negations, in order to obtain NA^ω-provability. We point this out in Sections 6.2.1 and 6.2.2. This theorem serves therefore also as a case study for the general method proposed in Chapter 8 regarding the use of negations in order to obtain minimal logic proofs from intuitionistically valid ones.

The chapter has two main components, structured around FPH and ES. We will first show in Section 6.1 the *finite* version of the Pigeonhole Principle and present its formalization, together with the result of applying the A-Translation. In Section 6.2 we comment on the issues arising when formalizing the Erdös-Szekeres Theorem (ES), in particular on the necessity to adapt the Pigeonhole Principle by a controlled insertion of negations. The new variant of FPH is presented in Section 6.2.2, in which we also compare

the extracted program with the one obtained in Section 6.1. We conclude by presenting in Section 6.2.4 the result of applying the modified realizability on the refined A-translated classical proof of the Erdös-Szekeres Theorem. The extracted program is presented first by abstracting over FPH and then by expanding it and normalizing the resulting term.

6.1 The Finite Pigeonhole Principle

We present in this section a formalization of the classical proof of FPH and propose user-defined functions for the constructive part of this proof. Since the formulation that we consider is a Π^0_2-statement, we apply the modified realizability on the A-translated proof and obtain a computable functional. The program associated to FPH will illustrate how sensitive the extraction procedure is to the proof strategy.

Lemma 6.1 (Finite Pigeonhole Principle (FPH))**.** *Given $n \in \mathtt{N}$ objects, $r \in \mathtt{N}$ colors and a coloring function $c : \{0, ..., n-1\} \to \{0, ..., r-1\}$, if for some $k \in \mathtt{N}$ we have $k \cdot r < n$, then at least $\mathsf{S}k$ objects have identical color.*

Proof. Assume that each of the colors has at most k objects assigned to it. Then there are in total at most $k \cdot r$ objects and since by the hypothesis $k \cdot r < n$, this contradicts the assumption that the r colors are assigned to *all* n objects.

More concretely, let us group all objects by their colors and construct the list ll, with ll_i collecting all objects of color i, $0 < i \leq r$. Thus, $|ll| = r$ and we assume that each sublist contains at most k objects, i.e., $\forall_{i \leq r} ll_i \leq k$. Then there are at most $|ll_i| \cdot |ll| = k \cdot r$ objects in the list. Since $k \cdot r < n$, this contradicts the assumption that $\sum_{i=1}^{n} |ll_i| = n$.

□

In the following, we present the implementation of this proof, as done in (Minlog). Since the proof contains constructive components, we formalize these in terms of predefined functions. The proof remains however essentially classical, since the existence is interpreted in a weak sense, so it is shown in an indirect way.

We work as before in the system NA^ω introduced in Chapter 2. As a consequence, we will need to avoid $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$.

6.1.1 Formalization

On the premise that c assigns to the objects at most r colors and that there are more objects then $k \cdot r$, we show that there exists a list of at least k objects, all of the same color m:

$$\forall_{k,n,r,c}.\ \forall_i c(i) < r \to k \cdot r < n \to \tilde{\exists}_l.\ k < |l| \ \tilde{\wedge}\ \tilde{\exists}_m \forall_{i<|l|} c(l_i) = m. \qquad (6.1)$$

For the constructive part of the proof concerning the grouping of objects according to their colors we have defined the functions specified below. We have also verified that the properties necessary in order to carry out the proof are fulfilled. We give in what follows their formulations and labels and leave further implementation details for Appendix A.1.

The main operation is performed by GroupbyCols, which distributes the objects in lists according to their colors. We construct a list ll of type $\mathtt{L(LN)}$ with the property that ll_i contains all objects of color i. For this, we define:

- $\mathsf{ChangeAt}(ll, i, n^{\mathtt{N}})$, an auxiliary function that modifies the list ll, by appending n to the sublist ll_i. The returned type is $\mathtt{L(L\,N)}$.

  With the exception of the i-th sublist, the components in ll are not affected. In addition, the length is not modified, unless $i > |l|$, in which case a new component is created in ll and the positions in-between are filled by empty lists. Thus,

$$\begin{aligned}
&\forall_i \mathsf{ChangeAt}(ll, i, n)_i = (n :: ll_i) && \text{(ithchanged)}\\
&\forall_i \forall_{j \neq i} \mathsf{ChangeAt}(ll, i, n)_j = ll_j && \text{(jthnotchanged)}\\
&|\mathsf{ChangeAt}(ll, i, n)| = \max(|ll|, \mathsf{S}i) && \text{(LhChangeAt)}
\end{aligned}$$

- $\mathsf{GroupbyCols}(c, n)$ groups the objects in lists according to their colors. It uses ChangeAt in order to construct the list ll, such that the object n is inserted in the $c(n)$-th list, i.e., at the position given by its color. The returned type is $\mathtt{L(L\,N)}$.

  Notation 6.1. *For readability purposes, we use in what follows the abbreviation*

$$GC := \mathsf{GroupbyCols}(c, n).$$

  In order to use GroupbyCols for proving FPH, we need to know that the grouping has been as expected, i.e., that the number of generated

lists does not exceed the number of colors and that all elements in a sublist ll_i share the color given by its index i:

$$\forall_i c(i) < r \to |\mathsf{GC}| \leq r \qquad \text{(ColLists)}$$
$$\forall_{i \leq r, j < |\mathsf{GC}_i|} c(\mathsf{GC}_{i,j}) = i. \qquad \text{(GroupByCols)}$$

- In order to count all the objects in the list obtained from GC, we construct the union of its sublists by $flatten(ll)$. We thus need the guarantee that the following properties hold:
  - the number of elements in a list ll is at most the maximum length of its sublists multiplied by the number of sublists:

    $$\forall_{ll,k}.\ (\forall_{i<|ll|}.\ |ll_i| \leq k) \to |\mathsf{flatten}(ll)| \leq k \cdot |ll| \qquad \text{(Lhflattenll)}$$

  - ChangeAt has only added an element to the list:

    $$|\mathsf{flatten}(\mathsf{ChangeAt}(ll, i, n))| = \mathsf{S}|\mathsf{flatten}ll| \qquad \text{(LhflattenedChangeAt)}$$

  - all given objects and only those are grouped by GC:

    $$|\mathsf{flatten}(\mathsf{GC})| = \mathsf{S}n \qquad \text{(ObjGrouped)}$$

Since none of these properties has computational content, we omit their proofs.

With these functions and properties we can now formally prove FPH.

Detailed formal proof. We fix arbitrary k, n, r and c and assume

$$\forall_i\, c(i) < r \qquad (H_1)$$
$$k \cdot r < n \qquad (H_2)$$
$$\forall_l.\ k < |l| \to \tilde{\exists}_m \forall_{i<|l|} c(l_i) = m \to \bot, \qquad (H_3)$$

where (H_3) results from rewriting the weak existence goal. It remains to show $\bot$.

Following the reasoning from the informal proof, $\bot$ should be inferred from (H_2) and (Lhflattenll). More precisely, using (ObjGrouped), (H_2) can be turned into

$$k \cdot r < |\mathsf{flatten}(\mathsf{GC})|.$$

Using (ColLists) and (H_1), we have further:

$$k \cdot |\mathsf{GC}| \leq k \cdot r < |\mathsf{flatten}(\mathsf{GC})|. \qquad (6.2)$$

Let $ll := \mathsf{GC}$ in (Lhflattenll). We have

$$|\mathsf{flatten}(\mathsf{GC})| \leq k \cdot |\mathsf{GC}| \tag{6.3}$$

provided that

$$\forall_{k,i<|\mathsf{GC}|} |\mathsf{GC}_i| \leq k. \tag{6.4}$$

From (6.3) and (6.2), we can derive F and by efq we reach the conclusion $\bot$.

However, we have to verify that (6.4) is fulfilled. This can be inferred from (H_3) by taking $l := \mathsf{GC}_i$ and observing that (GroupByCols) gives us $m := i$ which satisfies the requirement $\tilde{\exists}_m \forall_{i<|l|} c(l_i) = m$ of (H_3).

Negations It remains to show

$$(k < |\mathsf{GC}_i| \to \bot) \to |\mathsf{GC}_i| \leq k,$$

This is an instance of the lemma $\forall_{i,j}.\ (i < j \to \bot) \to j \leq i$, which requires in the proof (by induction on i and case distinction on j) an application of $\mathsf{efq}^{\bot}_{\mathsf{S}j\leq 0}$. As discussed in Section 8.1, $\mathsf{efq}^{\bot}$ is not acceptable in our system.

Thus, it would be more convenient to derive $\forall_{k,i<|\mathsf{GC}|}.\ k < |\mathsf{GC}_i| \to \bot$ instead of (6.4). For this reason, we introduce logical negations by "twisting" (Lhflattenll):

$$\forall_{ll,k}.\ (\forall_{i<|ll|}.\ k < |ll_i| \to \bot) \to k \cdot |ll| < |\mathsf{flatten}(ll)| \to \bot, \quad \text{(nLhflattenll)}$$

which turns out to be NA^{ω}-provable (we give the proof below).

With this, $\bot$ follows naturally. □

The λ- proof terms. We consider the following (typed) assumption variables

$$H_j^{j<|\mathsf{GC}_i|}, H_i^{i<|\mathsf{GC}|}, \tilde{H}_2{}^{k<|\mathsf{GC}_i|}$$

The proof of FPH in λ-form is

$$\begin{aligned} M_{\mathrm{FPH}} := \lambda k, n, r, c, (H_1), (H_2), (H_3).\ & \\ & (\mathsf{nLhflattenll}\ \mathsf{GC}\ k\ (\lambda i, H_i, \tilde{H}_2.\ H_3\ \mathsf{GC}_i\ \tilde{H}_2\ M_1)) \\ & (Lem_{=-\mathsf{Compat}}\ k\ c\ n\ \mathsf{S}n\ |\mathsf{flatten}(\mathsf{GC})|\ \mathsf{ObjGrouped}\ M_2), \end{aligned}$$

where $M_1^{\tilde{\exists}_m \forall_j (j<|\mathsf{GC}_i| \to c((\mathsf{GC}_i)_j)=m)}$ and $M_2^{k\cdot|\mathsf{GC}|<n}$ are given by:

$$\begin{aligned} M_1 := \tilde{\exists}^{+}\ n\ \lambda j, H_j.\ \mathsf{GroupByCols}\ c\ r\ n\ i & \\ (Lem_{Trans}\ i\ |\mathsf{GC}|\ r\ H_i\ (\mathsf{ColLists}\ c\ r\ nH_1))^{i\leq r}\ j\ H_j, & \\ M_2 := Lem_{Trans}\ k\cdot|\mathsf{GC}|\ k\cdot r\ n & \\ (Lem_{\leq \mathsf{Times}}\ k\ k\ |\mathsf{GC}|\ r\ \mathsf{AxT}\ (\mathsf{ColLists}\ H_1)^{|\mathsf{GC}|\leq r})^{k\cdot|\mathsf{GC}|\leq k\cdot r}\ H_2 & \end{aligned}$$

We have left aside some intermediate lemmas related to the transitivity of $\leq$ and $<$ (generically denoted by Lem_{Trans}). In addition to the properties for our predefined functions, we have used

$$Lem_{=-\mathsf{Compat}} : \forall^{nc}_{n_1,n_2}.\, n_1 = n_2 \to P(n_1) \to P(n_2), \text{ with } P(n) := k \cdot |\mathsf{GC}| < n$$
$$\tilde{\exists}^{+} : \forall_m.\, P(m) \to \tilde{\exists}_m P(m), \text{ with } P(m) := \forall_j.\, j < |\mathsf{GC}_i| \to c(\mathsf{GC}_{i,j}) = m$$
$$Lem_{\leq \mathsf{Times}} : \forall_{m_1,n_1,m_2,n_2}.\, m_1 \leq n_1 \to m_2 \leq n_2 \to m_1 \cdot m_2 \leq n_1 \cdot n_2$$

Neither of these lemmas has computational content, with the exception of $\tilde{\exists}^{+}$ and (nLhflattenll); we give the detailed term of the latter. In order to show (nLhflattenll) we need a lemma guaranteeing the compatibility of $+$ with $<$. This needs to be adapted as well, by changing $\leq$ to its logical complement:

$$\forall_{m_1,n_1,m_2,n_2}.\, (m_1 < n_1 \to \bot) \to (m_2 < n_2 \to \bot) \to m_1 + m_2 < n_1 + n_2 \to \bot, \qquad (Lem_{nPlusLt})$$

which is easily provable by induction on m_1 and case distinction on n_1. As we will see in what follows (page 104), there is a more suited proof, by induction on m_2 and n_2, giving a better extracted term.

The proof of (nLhflattenll) in λ-notation is:

$$\begin{aligned}
M_{\mathsf{nLhflattenll}} &:= \lambda ll.\, Ind\, ll\, M_{Base}\, M_{Step}, \text{ where}\\
M_{Base} &:= \lambda k, H_B.\, \mathsf{efq}\\
M_{Step} &:= \lambda l, ll, IH_{ll}, k, H_S.\, M_L^{k\cdot|l::ll|<|\mathsf{flatten}(l::ll)|\to\bot}\\
M_L &:= Lem_{nPlusLt}\ k\ |l|\ k \cdot |ll|\ |\mathsf{flatten}(ll)|\\
&\qquad (H_S\ 0\ \mathsf{AxT})^{k<|l|\to\bot}\ (IH_{ll}\, k\ \lambda i.\, H_S\, \mathsf{S}i)^{k\cdot|ll|<|\mathsf{flatten}(ll)|\to\bot},
\end{aligned}$$

where we have considered the typed assumption variables

$$H_B^{\mathsf{nLhflattenll}[ll:=(\mathsf{nil\,L\,N})]},\ IH_{ll}^{\mathsf{nLhflattenll}(ll)} \text{ and } H_S^{\forall_i.\, i<|(l::ll)|\to k<|(l::ll)_i|\to\bot}.$$

6.1.2 The Program Extracted from FPH

In order to apply A-Translation, we first need to verify that the assumptions are in the class $\mathcal{D}$ and that the goal is a $\mathcal{G}$-formula, according to Definition 3.5.

We have

- $D_1 := \forall_i c(i) < r \in \mathcal{D}$, since its kernel is atomic, so it is definite
- $D_2 := k \cdot r < n \in \mathcal{D}$, since it is an atomic formula,

- $GG_1 := k < |l| \in \mathcal{G}$, since it is atomic,
- $G_2 := (\forall_m.\, \forall_i(i < |l| \to c(l_i) = m) \to \bot) \to \bot \in \mathcal{G}$, because
  - $i < |l| \in \mathcal{D}$ and $c(l_i) = m \in \mathcal{I}_G \Rightarrow$
  - $i < |l| \to c(l_i) = m \in \mathcal{I}_G \subset \mathcal{G} \Rightarrow$
  - $\forall_i(i < |l| \to c(l_i) = m) \in \mathcal{G} \Rightarrow$
  - $\forall_i(i < |l| \to c(l_i) = m) \to \bot \in \mathcal{R}_D \Rightarrow$
  - $\forall_m.\, \forall_i(i < |l| \to c(l_i) = m) \to \bot \in \mathcal{R}_D$

  and $\bot \in \mathcal{G}$.

We can therefore proceed by applying Lemma 3.6. Clearly, this only affects G_2, since the other formulas do not contain $\bot$. From the above analysis for G_2 and the inductive proof of Lemma 3.6 it should be clear that transforming G by (3.4) amounts to using (3.1): $((R^F \to F) \to \bot) \to R$ on the relevant definite subformula $R := \forall_i(i < |l| \to c(l_i) = m) \to \bot$. Since for this we will need to perform a case distinction for $i < |l|$, we apply Lemma 3.4 and as a result the extracted term will contain an if-operator.

Further, we apply the modified realizability and extract a program by the rules from Definition 3.10. As with the other case studies, we use the "recipe" suggested by Theorem 3.5 and obtain the following program

$$P_{\mathrm{FPH}}(\mathsf{GC}) := \lambda k, n, r, c.\, \mathcal{R}_1\mathsf{GC}\, F_1\, G_1\, k\, (\lambda i.\, \mathsf{GC}_i),$$

with

$$\begin{aligned}
F_1 &:= \lambda k', f.\, (\mathsf{nil}^{\mathtt{N}}) \\
G_1 &:= \lambda l, ll, g, k', f.\, \mathcal{R}_2\, k'\, F_2\, G_2\, M \\
F_2 &:= \lambda i, n_0, n_1, l_1, l_2.\, [\mathsf{if}\ i\ l_2\ (\lambda n_2.\, l_1)] \\
G_2 &:= \lambda i, h, i', n_0, n_1, l_1, l_2.\, [\mathsf{if}\ i'\, l_2\, (\lambda n_2.\ \ h\ n_2\ n_0\ n_1\ l_1\ l_2)] \\
M &:= |l|\ (k' \cdot |ll|)\ |flatten(ll)|\ (f0)\ (g\, k'\, (\lambda i.\, f\, \mathsf{S}i))
\end{aligned}$$

where $f^{\mathtt{N}\Rightarrow\mathtt{LN}}$, $g^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{LN})\Rightarrow\mathtt{LN}}$ and $h^{\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}}$.

Here, $\mathcal{R}_1 : \mathcal{R}^{\sigma_1}_{\mathtt{L}(\rho)}$ and $\mathcal{R}_2 : \mathcal{R}^{\sigma_2}_{\mathtt{N}}$ with

$$\begin{aligned}
\rho &:= \mathtt{LN},\ \sigma_1 := \mathtt{N} \Rightarrow (\mathtt{N} \Rightarrow \mathtt{LN}) \Rightarrow \mathtt{LN} \\
\sigma_2 &:= \mathtt{N} \Rightarrow \mathtt{N} \Rightarrow \mathtt{N} \Rightarrow \mathtt{LN} \Rightarrow \mathtt{LN} \Rightarrow \mathtt{LN}.
\end{aligned}$$

Analysis of the extracted program. In order to understand the functionality of the extracted program, we unfold the recursions on the generic data k, n, r, c. According to the values collected in GC, we have for

$$\mathcal{R}_1 \mathsf{GC}\, F_1\, G_1\, k\, (\lambda i.\ \mathsf{GC}_i)$$

the following situations[1].

- $\mathsf{GC} = \mathsf{nil}$ then $P_{\mathrm{FPH}}\, k\, n\, r\, c = (\mathsf{nil}^{\mathtt{N}})$.
- $\mathsf{GC} = (l :: ll)$ in this case,

  $$P_{\mathrm{FPH}}\, k\, n\, r\, c = \mathcal{R}_2\, k\, F_2\, G_2\, M\, \mathsf{S},$$

  where S is the substitution $[l := l, ll := ll, g := \mathcal{R}_1\, ll\, F_1\, G_1, k' := k, f := \lambda i.\ \mathsf{GC}_i.]$ and we have the following two subcases,

  $$\begin{aligned} P_{\mathrm{FPH}(\mathsf{GC})}\, 0\, n\, r\, c &= [\mathsf{if}\ |l|\ (P_{\mathrm{FPH}(\mathsf{cdrGC})}\, 0\, n\, r\, c)\ (\lambda n_2.\ \mathsf{GC}_0)] \\ P_{\mathrm{FPH}(\mathsf{GC})}\, \mathsf{S}k\, n\, r\, c &= [\mathsf{if}\ |l| \\ &\qquad P_{\mathrm{FPH}(\mathsf{cdr}(\mathsf{GC}))}\, k\, n\, r\, c \\ &\qquad (\lambda n_2.\ \ G_1\, \mathsf{nil}^{\mathtt{N}}\, (\mathsf{cdrGC})\, g\, k\, [|ll| := n_2])] \end{aligned}$$

  As expected, the program reflects the decisions in the implementation:

- the main computation is performed, as expected, by $GroupbyCols(c, n)$ (GC), which is responsible for the grouping of objects by colors. Thus, the first recursion proceeds according to the results returned by GC
- $\mathcal{R}_1$ corresponds to the use of (nLhflattenll) and the recursion step G_1 performs the computation according to ($Lem_{nPlusLt}$)
- $\mathcal{R}_2$ corresponds to the use of ($Lem_{nPlusLt}$) and performs a recursion on the constant k. This is an expected consequence of the way in which induction is used for proving ($Lem_{nPlusLt}$) and is in fact rather disturbing. We give in what follows the results of an alternative proof of this auxiliary lemma, in which we use induction on m_2 and n_2; consequently, we obtain the associated operator $\mathcal{R}'_2 : \mathcal{R}_{\mathtt{N}}^{\sigma'_2}$, with the new type:

  $$\sigma'_2 := \mathtt{N} \Rightarrow \mathtt{N} \Rightarrow \mathtt{LN} \Rightarrow \mathtt{LN} \Rightarrow \mathtt{LN}$$

[1] For a detailed unfolding with application of β-reduction rules, see the Appendix A.1.

We make it a point here to underline how sensitive program extraction is to the techniques used in the proof. The program terms reflect with accuracy the proof terms, as it is the case with tail recursion when using A-Translation. On top of this, the refined A-Translation adds a specific behavior resulting from the treatment of $\bot$, which triggers the use of continuations in the computation.

The resulting program is $P'_{\mathrm{FPH}}(\mathsf{GC})$, which we present below.

- The induction step for $(Lem_{nPlusLt})$ has the associated term G_2 in which the recursive call is performed by h. The parameters taken by h are gathered in the substitution term M and they have the following meaning:
  - $l :+: ll$ is the list returned by GC
  - the length of l corresponds to the number of objects of the current color
  - $k' \cdot |ll|$ stands for the result of multiplying the constant k with the number of remaining colors, after deleting the head component of the list returned by GC
  - $|flatten(ll)|$ corresponds to the number of objects remaining in the current list
  - $f0$ is the first element in the list returned by GC and $\lambda i.\ f\mathsf{S}i$ is the element at position i in GC.

The program associated to FPH using the modified proof of $(Lem_{nPlusLt})$ is:

$$P'_{\mathrm{FPH}}(\mathsf{GC}) := \lambda k, n, r, c.\ \mathcal{R}_1 \mathsf{GC}\, F_1\, G'_1\, k\, (\lambda i.\ \mathsf{GC}_i),$$

where

$$\begin{aligned}
F_1 &:= \lambda k', f.\ (\mathsf{nil}^{\mathtt{N}}) \\
G'_1 &:= \lambda l, ll, g, k', f.\ \mathcal{R}'_2\, k' \cdot |ll|\, F'_2\, G'_2\, M' \\
F'_2 &:= \lambda i, l_1, l_2.\ [\mathsf{if}\ i\, l_1\, (\lambda n_2.\ l_2)] \\
G'_2 &:= \lambda i, h', i', l_1, l_2.\ [\mathsf{if}\ i'\ (h'\ 0\ l_1\ l_2)\ (\lambda n_2.\ h'\ n_2\ l_1\ l_2)] \\
M' &:= |flatten\, ll|\ f0\ (g\, k'\, (\lambda i.\ f\mathsf{S}i))
\end{aligned}$$

and $h'^{\mathtt{N}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}}$.

The change in the induction proof of $(Lem_{nPlusLt})$ affects the recursion operator associated to it. Thus, $\mathcal{R}'_2$ operates on $k' \cdot |ll|$, i.e., the number

of objects in the current list. As a consequence, the step function h' is parameterized only by the number of colors and the remaining list, whereas the number of objects in the head of the list and the result of $k' \cdot |ll|$ are dropped from its arguments.

Following the change in the proof of $(Lem_{nPlusLt})$, the recursion is performed as long as the list ll is non-empty, as the following unfolding of P'_{FPH} shows:

- Case $\mathsf{GC} = \mathsf{nil}$

$$P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c = (\mathsf{nil}^{\mathbb{N}})$$

- Case $\mathsf{GC} = (l :: ll)$

$$P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c = \mathcal{R}'_2\, k' \cdot |ll|\, F'_2\, G'_2\, M'\, \mathsf{S},$$

where $\mathsf{S} := [l := l, ll := ll, g := \mathcal{R}_1\, ll\, F_1\, G'_1, k' := k, f := \lambda i.\ \mathsf{GC}_i.]$

Further,

If $k \cdot |ll| = 0$ then

$$P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c = [\mathsf{if}\ |flatten\, ll|\ \mathsf{GC}_0\ (\lambda n_2.\ P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c)]$$

If $k \cdot |ll| \neq 0$ then

$$P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c = [\mathsf{if}\ |flatten\, ll|\ ((\mathcal{R}'_2\, k' \cdot |ll|\, F'_2\, G'_2)\ 0\ l_1\ l_2)\ (\lambda n_2.\ (\mathcal{R}'_2\, k' \cdot |ll|\, F'_2\, G'_2)\ n_2\ l_1\ l_2)]\, \mathsf{S}'$$

where $\mathsf{S}' := [l_1 := f0,\ l_2 = g\, k'\, (\lambda i.\ f\mathsf{S}i)] \cup \mathsf{S}$.
With the abbreviations

$$\begin{aligned} T_2 &:= \mathcal{R}'_2\, k \cdot |ll|\, F'_2\, G'_2 \text{ and} \\ Case(x) &:= T_2\ x\ \mathsf{GC}_0\ (\mathcal{R}_1\, ll\, F_1\, G'_1)\, k\, (\lambda i.\ \mathsf{GC}_{\mathsf{S}i}) \\ &= T_2\ x\ \mathsf{GC}_0\ P'_{\mathrm{FPH}(\mathsf{cdr}(\mathsf{GC}))}\, k\, n\, r\, c. \end{aligned}$$

we have

$$P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c = [\mathsf{if}\ |flatten\, ll|\ Case(0)(\lambda n_2.\ Case(n_2))],$$

The unfolding for $P'_{\mathrm{FPH}(\mathsf{GC})}$ illustrates the runs based on the values for $k \cdot |lh|$ and $|flatten\, ll| \neq 0$. In case $k \cdot |lh| = 0$, but ll is not empty ($|flatten\, ll| \neq 0$), the solution can be found already at GC_0. Otherwise, if the list ll is empty, the program proceeds recursively, but investigates the

lists only from n_2 onwards. In case $k \cdot |lh| \neq 0$, GC is also investigated recursively, but again from n_2 onwards in case $|flatten\, ll| \neq 0$.

The programs $P_{\mathrm{FPH}}(\mathsf{GC})$ and $P'_{\mathrm{FPH}}(\mathsf{GC})$ are both in *tail recursive form* - which illustrates the observation from Section 4.1 that tail recursion is a beneficial side-effect of using A-Translation. This can be identified in the fact that each of the step functions g and h from G_1 and respectively G_2 occur in tail position. At each step of the program a verification is performed as to whether the remaining sublists of GC still comply with the requirement that the sum of their lengths coincides with the number of elements. This reflects the computation by continuations strategy. It would be otherwise unnecessary to perform this check at each step, but this can be viewed as having equipped the program also with a verification procedure. By this, we are guaranteed that each step of the program performs valid operations on the list.

The Computational Role of Negations. The main computation of $P'_{\mathrm{FPH}}(\mathsf{GC})$ depends on the result of (nLhflattenll). The verification whether the current list ll satisfies the requirement $k \cdot r < n$ is in fact computed by the term extracted from $(Lem_{nPlusLt})$. This performs the calculations $k \not< |l| \to k \cdot |ll| \not< |flatten\, ll| \to \mathsf{S}k \cdot |ll| \not< |l| + |flatten\, ll|$, with $|ll| = r$ and $|l| + |flatten\, ll| = n$. These lemmas have computational terms associated to their proofs only when the logical falsity occurs in the specifications. This justifies the need to "manually" insert the double negations, which allow to unravel the construction hidden in the classical proof. The refined A-translation comes into effect only when $\bot$ is inserted in the relevant positions. Since the construction hidden in the proof cannot be recovered from the use of $\mathsf{efq}^{\bot}$, by requiring the underlying system to be the minimal logic, we shift the use of $\bot$ from $\mathsf{efq}^{\bot}$ to the formulas whose atoms are double negated. These atoms will therefore contribute to the computations performed in order to produce the end result - hence, they are involved in the construction of the witness realizing the goal formula. The mechanism for a systematic detection of these relevant atoms - called L-critical - is done in Chapter 8.

6.2 The Erdös-Szekeres theorem

The theorem presented in this chapter is an interesting combinatorial property. It is based on the result of FPH, combining constructive and classical proof strategies.

Theorem 6.1 (Erdös-Szekeres)**.** *Let $n \in \mathtt{N}$ be given and let s be a sequence over $\mathtt{N}$ of length $n^2 + 1$. There exists either an increasing or a decreasing subsequence of s of length at least $\mathsf{S}n$.*

Proof. In the following, we give an informal sketch of the proof using the Finite Pigeonhole Principle.

Step one: The construction. Let $f_{s,i}$ and $g_{s,i}$ denote a longest increasing and (respectively) decreasing subsequence of s ending in s_i. To each s_i, we associate the pair $(|f_{s,i}|, |g_{s,i}|)$. Note that there might be more subsequences of the same length ending in s_i, as illustrated in Example 5; we consider the first of them.

Step two: The false assumption. We will prove the claim by contradiction. In addition to the given hypothesis, we also assume that all increasing and decreasing subsequences of s are of length at most n:

$$\forall_i.\ |f_{s,i}| \leq n \wedge |g_{s,i}| \leq n \tag{6.5}$$

Step three: Finite pigeonhole principle. The assumption (6.5) implies there exist at most n^2 distinct pairs $(f_{s,i}, g_{s,i})$ associated to the $n^2 + 1$ elements of the sequence.

We can interpret "distinct" as referring to the colors associated to the pairs. Therefore, we have n^2 colors for the $n^2 + 1$ objects. If this is the case, then by the Finite Pigeonhole Principle

$$\tilde{\exists}_{i<j}(|f_{s,i}|, |g_{s,i}|) = (|f_{s,j}|, |g_{s,j}|), \tag{6.6}$$

which means that for some i and j, $f_{s,i} = f_{s,j}$ and $g_{s,i} = g_{s,j}$.

Step four: The contradiction. We compare s_i and s_j.

If $s_i \leq s_j$, then the longest increasing subsequence for s_j must be at least by one element (s_j) longer than $f_{s,i}$, i.e., $f_{s,i} < f_{s,j}$. This contradicts (6.6) from Step 3.
Observe that, as shown in Example 5, it is not necessarily the case that $f_{s,i} \subset f_{s,j}$.

If $s_i \geq s_j$, then $g_{s,i} < g_{s,j}$, by the same reasoning as in the previous case and this comes in contradiction with (6.6).

□

Example 5. *Let* $n := 10$ *and* $s := \{7, 8, 2, 4, 10, 3, 9, 4, 5, 11\}$.

For $i \in \{1, 10\}$ *we have the following set of pairs* $(|f_{s,i}|, |g_{s,i}|)$

$$\{(1,1), (2,1), (1,2), (2,2), (3,1), (2,3), (3,2), (3,3), (4,3), (4,1)\}$$

Consider the increasing subsequences associated to $s_6 = 9$. *The maximal length for such sequences is* 3 *and we have multiple options:* $\{7, 8, 9\}$, $\{2, 4, 9\}$, $\{2, 3, 9\}$. *We take* $f_{s,6} = \{7, 8, 9\}$.

We also have $s_3 = 4 < s_6 = 9$. *Consequently,* $|f_{s,3}| = 2 < |f_{s,6}| = 3$, *although* $f_{s,3} = \{2, 4\} \not\subset f_{s,6} = \{7, 8, 9\}$.

If we take $s_1 = 8$ *and* $s_9 = 11$, *then* $f_{s,2} = \{7, 8\}$ *and* $f_{s,10} = \{2, 4, 4, 5, 11\}$, *so* $f_{s,2}$ *and* $f_{s,10}$ *are also disjoint. However,* $|f_{s,2}| = 2 < |f_{s,10}| = 5$.

Any increasing sequence including $f_{s,2}$ *which ends in* 11 *is of length smaller than* $5 = |f_{s,10}|$*:* $\{7, 8, 9, 11\}$ *or* $\{7, 8, 10, 11\}$.

6.2.1 Formalization Issues

The property to prove is the following (classical) existence statement:

$$\forall_{n^{\mathtt{N}}, s^{\mathtt{L}}}.\ |s| = n^2 + 1 \to \tilde{\exists}_i.\ \mathsf{S}n \leq |f_{s,i}| \ \tilde{\vee}\ \mathsf{S}n \leq |g_{s,i}| \qquad (\mathrm{ES}_{\ \tilde{\vee}})$$

Before presenting the formalized proof of this statement, let us discuss a few issues and the decisions that we made in approaching them.

First issue: the disjunction.

Recall that we regard disjunction only as an abbreviation. We can represent " $\tilde{\vee}$ " in

$$\tilde{\exists}_i.\ \mathsf{S}n \leq |f_{s,i}| \ \tilde{\vee}\ \mathsf{S}n \leq |g_{s,i}|. \qquad (\mathrm{Goal}_{\mathrm{ES}})$$

as either of the following:

$$\tilde{\exists}_{p^{\mathtt{B}}}.(p \to \mathsf{S}n \leq |f_{s,i}|) \ \tilde{\wedge}\ (\neg p \to \mathsf{S}n \leq |g_{s,i}|) \qquad (\tilde{\vee}_1)$$

$$(\mathsf{S}n \leq |f_{s,i}| \to \bot) \to \mathsf{S}n \leq |g_{s,i}| \qquad (\tilde{\vee}_2)$$

$$|f_{s,i}| < \mathsf{S}n \to |g_{s,i}| < \mathsf{S}n \to \bot, \qquad (\tilde{\vee}_3)$$

In choosing the above formulations, the main concern was to minimize the number of $\bot$.

It is clear that there is much flexibility in how we unfold the weak disjunction. For instance, $(\ \tilde{\vee}_3)$ may be formulated in a fashion similar to $(\ \tilde{\vee}_2)$ as $|f_{s,i}| < \mathsf{S}n \to \mathsf{S}n \leq |g_{s,i}|$. However, we have opted for $(\ \tilde{\vee}_3)$, which is closer to the false assumption (6.5) from Step 2 in the informal presentation of the proof. At the cost of having reversed $\leq$, we have gained the symmetry.

Solution. We have chosen ($\tilde{\vee}_3$), which reads "It cannot be the case that both $|f_{s,i}|$ and $|g_{s,i}|$ are smaller than $\mathsf{S}n$" and is the closest to the informal proof. As we shall see when discussing the fourth issue in Section 6.2.1, this maps well to the use of (FPH).

Second issue: the construction of $f_{s,i}$ and $g_{s,i}$.

For each s_i in the given sequence we need to find the longest increasing/decreasing sequence ending in s_i. This is the "Longest increasing subsequence" problem, known to be solvable in time O(n log n).

Solution. For this constructive component of the classical proof we will give directly the algorithm.

In order to compute the longest sequence $f_{s,\mathsf{S}i}$ associated to s_i, we define the function $\mathsf{Bar}f$, which constructs the list $\overline{f}_{s,\mathsf{S}i} := (f_{s,1}, ..., f_{s,\mathsf{S}i})$ where each $f_{s,\mathsf{S}j}$ with $0 \leq j \leq i$ is

1. a sequence ending with s_j
2. monotonously increasing
3. a longest sequence with the above two properties.

We have broken down the definition of $Barf$ into smaller functions, according to the properties that it needs to fulfill. Since we regard them as built-in functions, we give them in infix notation. For the same reason, we allow a generic type α in the definition, whenever this is possible.

- $l^{\mathtt{L}\alpha}$ filter $\psi^{\alpha\Rightarrow\mathtt{B}}$ selects from the given list l the elements that fulfill the filtering criterion ψ. filter has the return type $\mathtt{L}\ \alpha$.

  The purpose is to filter $(f_{s,1}, ..., f_{s,n})$ such that it fulfills 1. and 2. above, i.e., the last element of $f_{s,i}$, $1 \leq i \leq n$ must be smaller than s_n.

  Its functionality is described by

  $$(\mathsf{nil}\ \alpha)\,\mathsf{filter}\,\psi := (\mathsf{nil}\ \alpha)$$
  $$(x :: l)\,\mathsf{filter}\,\psi := \mathsf{if}\ (\psi x)\ \mathsf{then}\ ((x :: l)\,\mathsf{filter}\,\psi)\ \mathsf{else}\ (l\,\mathsf{filter}\,\psi)$$

- $l^{\mathtt{L}\alpha}$ maxwrt $\chi^{\alpha\Rightarrow\mathtt{N}}$ extracts from the list l the greatest element with respect to the measure function χ.

  The intention is to compare the lengths of the increasing subsequences returned by filter and select the longest among them.

Its functionality is described by

$$(\mathsf{nil}\ \alpha)\ \mathsf{maxwrt}\ \chi := (\mathsf{Inhab}\ \alpha)$$
$$(x :: l)\ \mathsf{maxwrt}\ \chi := (\lambda y.\ \mathsf{if}\ (\chi x \leq \chi y)\ \mathsf{then}\ y\ \mathsf{else}\ x)\ (l\ \mathsf{maxwrt}\ \chi),$$

where Inhab is a generic object inhabiting the *nil* list.

We want to guarantee that it determines indeed the maximal element of the list:

$$\forall_{l,i}.\ i < |l| \rightarrow |l_i| \leq |l\ \mathsf{maxwrt}\ |||\qquad (max|l|)$$

We can now define $\mathsf{Bar}f(s^{\mathtt{LN}}, i^{\mathtt{N}})^{\mathtt{LLN}}$, in order to construct $\overline{f}_{s,i} := (f_{s,1}..., f_{s,i})$. The new component $f_{s,\mathsf{S}i}$ of $\overline{f}_{s,\mathsf{S}i}$ is determined as follows:

- compute a sublist $\widehat{f}_{s,i}$ by filtering out all lists in $\overline{f}_{s,i}$ which do not fulfill the requirement $\lambda l.\ \mathsf{last}(l) \leq s_i$
- select from $\widehat{f}_{s,i}$ the list which is maximal w.r.t. the length $|\cdot|$.

This functionality is expressed by

$$\begin{aligned}
\mathsf{Bar}f(s, 0) &:= (\mathsf{nil}\ \mathtt{L}\ \mathtt{N})\\
\mathsf{Bar}f(s, \mathsf{S}i) &:= \mathsf{Bar}f(s, i)\mathtt{+\!+}(\widehat{f}_{s,i}\ \mathsf{maxwrt}\ \lambda l|l| :: s_i):,\\
\text{where } \widehat{f}_{s,i} &:= \mathsf{Bar}f(s, i)\ \mathsf{filter}\ \lambda l.\mathsf{last}(l) \leq s_i\\
&\quad \text{and } \mathsf{maxwrt} \text{ binds stronger than } ::.
\end{aligned}$$

In a very symmetric way, we construct $\mathsf{Bar}g(s^{\mathtt{LN}}, i^{\mathtt{N}})^{\mathtt{LLN}}$, by filtering $\overline{g}_{s,i} := (g_{s,1}..., g_{s,i})$. according to the criterion $\lambda l.\ s_i < \mathsf{last}(l)$.

Third issue: the coloring to use with the FPH

We define the coloring used to count the *distinct* pairs (f_i, g_i). Let n and $M \subset \mathtt{N}$ be such that the function

$$\mathsf{col} : \{\ (a, b)\ |\ a, b \in \mathtt{N}, a < n \wedge b < n\ \} \rightarrow M \qquad (6.7)$$

is bijective, i.e.,

- $\mathsf{col}(a_1, b_1) = \mathsf{col}(a_2, b_2) \rightarrow a_1 = a_2 \wedge b_1 = b_2$
- $|M| := |\{\ (a, b)\ |\ a, b \in \mathtt{N} \wedge a, b < n\ \}| = n \times n$

We take

$$\mathsf{col}(a, b) := a \cdot n + b \tag{6.8}$$

This is of course not the only function fulfilling the requirements, but we prefer it for its simplicity.

Lemma 6.2. col *as defined by* (6.8) *meets the requirements that we need in order to use* FPH*.*

Proof. We show that *col* is injective and that its value is at most n^2.

1. Let $a_i, b_i \in \mathtt{N}$, $i = 1, 2$. We can show that:

   $$a_{1,2}, b_{1,2} < n \rightarrow \mathsf{col}(a_1, b_1) = \mathsf{col}(a_2, b_2) \rightarrow a_1 = a_2 \wedge b_1 = b_2. \quad (2\mathsf{Col}_{=})$$

   Let $\mathsf{col}(a_1, b_1) = \mathsf{col}(a_2, b_2)$.

   Then $a_1 \cdot n + b_1 = a_2 \cdot n + b_2 \Leftrightarrow (a_1 - a_2) \cdot n = b_2 - b_1$.

   Since $0 \leq b_1, b_2 < n$, we have $-n < b_2 - b_1 < n$.

   Thus, $(a_1 - a_2) \cdot n = b_2 - b_1 \Leftrightarrow b_2 - b_1 = 0 \wedge a_1 - a_2 = 0$ i.e., $a_1 = a_2 \wedge b_1 = b_2$.

2. We want to show that n^2 is the upper bound of the chosen coloring

   $$\forall_{a,b<n} \mathsf{col}(a, b) < n^2 \tag{ColUB}$$

   By the definition of col and the monotonicity of $\leq$ with respect to multiplication and addition, we have

   $$\mathsf{col}(a, b) = a \cdot n + b \leq (n - 1) \cdot n + n - 1 = n^2 - 1 < n^2.$$

□

Fourth issue: the negative assumption and usage of FPH

In the third step of our informal proof, we have used the false assumption

$$\forall i. |f_{s,i}| \leq n \wedge |g_{s,i}| \leq n \tag{6.5}$$

to infer that there exist at most n^2 distinct pairs, as shown in (ColUB). This property, together with the assumption for ($\mathrm{ES}_{\tilde{\vee}}$) that there are n^2+1 pairs, is part of the hypothesis for FPH.

However, we cannot derive in minimal logic a positive (or irrelevant) property from a negative (relevant) assumption. More precisely, (6.5) cannot be obtained as it is from ($\mathrm{Goal}_{\mathrm{ES}}$):

$$\tilde{\exists}_i.\ \mathsf{S}n \le |f_{s,i}| \ \tilde{\vee}\ \mathsf{S}n \le |g_{s,i}|$$

since, independent of the form in which we unfold $\tilde{\vee}$, the (FalseH) for the Erdös-Szekeres Theorem

$$\forall_i.(\mathsf{S}n \le |f_{s,i}| \ \tilde{\vee}\ \mathsf{S}n \le |g_{s,i}|) \to \bot \qquad \text{(FalseH)}$$

will contain at least one occurrence of $\bot$[2].

Therefore, we need to change the assumption in FPH by explicitly inserting negations, either to say $\forall_i.\ n^2 < c(i) \to \bot$ or $\forall_i.\ (c(i) \le n^2 \to \bot) \to \bot$. Concerning the former alternative, it is impossible to derive in NA^ω $\tilde{\exists}_{i<j\le n} c(i) = c(j)$, since this would amount to inferring $\bot$ from two negative assumptions. More precisely, we would have the following scenario:

$$\text{Given } \neg A \text{ show } \tilde{\exists}_x B.$$

That is, we need to combine $\neg A$ and $\forall_x \neg B$ in order to show $\bot$. For this, we either have to use $\neg A$ to show B (at some fixed x) or use $\forall_x \neg B$ to prove A. Either of them means that a relevant formula should be used to derive an irrelevant formula, which is not possible in minimal logic, since it requires $\mathsf{efq}^\bot$.

Solution. In order to carry out the proof in minimal logic we need to change the assumption in FPH by (manually) inserting a double negation:

$$\forall_i.\ (c(i) \le r \to \bot) \to \bot$$

and explicitely write $\tilde{\vee}$ as suggested by ($\tilde{\vee}_3$):

$$|f_{s,i}| < \mathsf{S}n \to |g_{s,i}| < \mathsf{S}n \to \bot.$$

Modified in this way FPH becomes

$$\forall_{k,n,r,c}.\ \forall_i\ \tilde{\neg}\ \tilde{\neg}(c(i) < r) \to k \cdot r < n \to \\ \tilde{\exists}_l.\ k < |l| \ \tilde{\wedge}\ \forall_i(\mathsf{S}i < |l| \to l_{\mathsf{S}i} < l_i) \ \tilde{\wedge}\ \tilde{\exists}_m \forall_i.\ i < |l| \to c(l_i) = m \qquad (\mathrm{FPH}_{dn})$$

which is easy to prove in NA^ω, following the key ideas of FPH. We give in the next section its proof and the associated realizer.

[2]The $\bot$ comes from viewing $\tilde{\exists}_x A$ as a shortcut for $\forall_x \neg A \to \bot$. Of course, we can shift it around, depending on how we unfold $\tilde{\vee}$, but we cannot completely avoid it.

6.2.2 The Double-negated FPH

We show (FPH_{dn}), which is the modified version of FPH that enables us to show ES in minimal logic.

Remark 6.1. *As with* IPH*, the last property of the list can be formulated as "all elements are of the same color", instead of "there exists a color shared by all elements in the list". By this, we eliminate one* $\tilde{\exists}$ *from the goal of* (FPH_{dn})*.*

We first analyze where our proof of FPH fails for (FPH_{dn}), in order to detect which lemmas need to be modified to accommodate the changes in FPH_{dn}. Since this is clearly connected with changing the assumption $c(i) < r$ into its double negated form, the use of "ColLists" is likely to be problematic. Indeed, we cannot use it without $\mathsf{Stab}^{\perp}$, so we need to adapt it by incorporating the double negation. We turn "ColLists"into:

$$\text{"dnColLists"} : \forall_{c,r,n}.\, \forall_i\, \tilde{\neg}\,\tilde{\neg} c(i) < r \to\ \tilde{\neg}\,\tilde{\neg} |GC| \le r. \qquad (\mathsf{dnColLists})$$

Since "dnColLists" has computational content, we will present its proof.

Proof of (dnColLists)**.** First, let us remark that "dnColLists" is the Gödel-Gentzen Translation of "ColLists", so it can be proved by the same derivation. In what follows, we give a separate proof and sketch a comparison with the proof for "ColLists".

Using the auxiliary property:

$$\max_{UB} : \forall_{n,m,k}.\, n \le k \to m \le k \to \max(m,n) \le k.$$

and the compatibility lemma

$$Lem_{\le -\mathsf{Compat}} := \forall^{nc}_{r,n_1,n_2}.\, n_1 = n_2 \to n_1 \le r \to n_2 \le r$$

we have

$$\begin{aligned}
M_{dnColLists} &:= \lambda c, r, n.\ Ind\ n\ M_{Base}\ M_{Step},\ \text{where}\\
M_{Base} &:= \lambda H_2.\ Lem_{\le -\mathsf{Compat}}\ r\ t_1((),0)\ t_2((),0)\ t_3((),0)\\
&\qquad (\lambda \tilde{H}_2.\ H_2\, 0\ (\lambda H_0^{c(0)<r}.\ \tilde{H}_2\ H_0))\\
M_{Step} &:= \lambda n, IH_2, H_2.\ Lem_{\le -\mathsf{Compat}}\ r\ t_1(\mathsf{GC}, \mathsf{S}n)\ t_2(\mathsf{GC}, \mathsf{S}n)\ t_3(\mathsf{GC}, \mathsf{S}n)\\
&\qquad (\lambda \tilde{G}_1.\ IH_2\ H_2\ (\lambda \tilde{G}_2.\ H_2\ \mathsf{S}n\ (\lambda H_{\mathsf{S}n}.\ \tilde{G}_1\ (\max_{UB}\ \mathsf{S}c(\mathsf{S}n)\ |\mathsf{GC}|\ r\ H_{\mathsf{S}n}\ \tilde{G}_2))))
\end{aligned}$$

The assumptions have the following typing:

$$H_2^{\forall_i \tilde{\neg}\tilde{\neg}(c(i)<r)}, \qquad IH_2^{\forall_i \tilde{\neg}\tilde{\neg}c(i)<r\to \tilde{\neg}\tilde{\neg}|\mathsf{GC}|\leq r}, \qquad H_{\mathsf{S}n}^{c(\mathsf{S}n)<r},$$
$$\tilde{G}_1^{t_1(\mathsf{GC},\mathsf{S}n)\leq r\to\bot}, \qquad \tilde{G}_2^{|\mathsf{GC}|\leq r}, \qquad \tilde{H}_2^{\,t_1((),0)\leq r\to\bot}$$

and we have used the abbreviations

$$t_1(l,i) := \max(|l|, \mathsf{S}c(i)),$$
$$t_2(l,i) := |(\mathsf{ChangeAt}\ l\ c(i)\ i)| \overset{\mathsf{LhChangeAt}}{=} \max(|l|, \mathsf{S}c(i)),$$
$$t_3(l,i) := \mathsf{LhChangeAt}\ l\ c(i)\ i.$$

None of the terms t_1, t_2, t_3 have computational content.

In the above, we have regarded $c(0) < r$ and $\mathsf{S}c(0) \leq r$ as being equivalent and have also used implicitly the normalization of $\max(|()|, \mathsf{S}c(0))$ to $\mathsf{S}c(0)$.

Comparison of the proofs of "ColLists" and "dnColLists". For "ColLists" the typing for the assumptions is $H_1^{\forall_i c(i)<r}, IH_1^{\forall_i c(i)<r\to|\mathsf{GC}|\leq r}$. Its proof is:

$$\begin{aligned} M_{ColLists} &:= \lambda c, r, n.\ Ind\ c\ r\ n\ M_{base}\ M_{step} \\ M_{base} &:= \lambda H_1.\ Lem_{\leq-\mathsf{Compat}} r\ t_1((),0)\ t_2((),0)\ t_3((),0)\ (H_1\ 0) \\ M_{step} &:= \lambda n, IH_1, H_1.\ Lem_{\leq-\mathsf{Compat}}\ t_1(\mathsf{GC},\mathsf{S}n)\ t_2(\mathsf{GC},\mathsf{S}n)\ \ t_3(\mathsf{GC},\mathsf{S}n) \\ &\qquad (\max_{UB}\ \mathsf{S}c(\mathsf{S}n)\ |\mathsf{GC}|\ r\ (H_1\ \mathsf{S}n)\ (IH_1\ H_1)), \end{aligned}$$

We compare the two proofs, by presenting the terms in parallel in Table 6.1. For simplification reasons, we omit from M_{base} and M_{Step} the identical terms corresponding to the use of $Lem_{\leq-\mathsf{Compat}}$.

	ColLists	dnColLists
M	$\lambda c, r, n.\ Ind\ n\ M_{base}\ M_{step}$	$\lambda c, r, n.\ Ind\ n\ M_{Base}\ M_{Step}$
M_{base}	$\lambda H_1.\ H_1\ 0$	$\lambda H_2, \tilde{H}_2.\ H_2\ 0\ \tilde{H}_2$
M_{step}	$\lambda n, IH_1, H_1.$ $\max_{UB}\ \mathsf{S}c(\mathsf{S}n)\ \|\mathsf{GC}\|\ r\ (H_1\ \mathsf{S}n)\ (IH_1\ H_1)$	$\lambda n, IH_2, H_2.$ $\lambda\tilde{G}_1.\ IH_2\ H_2\ (\lambda\tilde{G}_2.\ H_2\ \mathsf{S}n\ (\lambda H_{\mathsf{S}n}.\ \tilde{G}_1$ $(\max_{UB}\ \mathsf{S}c(\mathsf{S}n)\ \|\mathsf{GC}\|\ r\ H_{\mathsf{S}n}\ \tilde{G}_2)))$

Table 6.1: Comparison between the proof terms of ColLists and dnColLists

Formalized proof of FPH_{dn}**.** We can now prove (FPH_{dn}), following the pattern from FPH.

Proof. We use the same auxiliary lemma as for FPH, GroupByCols, nLhflattenII, ObjGrouped, together with $Lem_{\leq -\mathsf{Compat}}$, $\tilde{\exists}^+$ and $Lem_{\leq \mathsf{Times}}$. ColLists needs to be modified to dnColLists . We take:

$$H_3^{\forall_l(k<|l|\to\tilde{\exists}_m\forall_i(i<|l|\to c(l_i)=m)\to\bot)},\ \tilde{H}_2^{\,k<|\mathsf{GC}_i|},$$
$$(H_1')^{\forall_i.\ \tilde{\neg}\,\tilde{\neg} c(i)<r},\ H_2^{k\cdot r<n},\ H_i^{i<|\mathsf{GC}|},\ H^{|\mathsf{GC}|\leq r}$$

The proof term for FPH_{dn} is:

$$\begin{aligned}
M_{\mathrm{FPH}_{dn}} :=& \lambda k, n, r, c, H_1', H_2, H_3.\\
&(\mathsf{dnColLists}\ c\ r\ n\ H_1')\\
&(\lambda H.\\
&\quad (\mathsf{nLhflattenII}\ \mathsf{GC}\ k\ (\lambda i, H_i, \tilde{H}_2.\ (H_3\ \mathsf{GC}_i\ \tilde{H}_2\ N_1)^{\bot}))^{k\cdot|\mathsf{GC}|<|\mathsf{flatten(GC)}|\to\bot}\\
&\quad (Lem_{\leq -\mathsf{Compat}}\ k\ c\ n\ \mathsf{S}n\ |\mathsf{flatten(GC)}|\ \mathsf{ObjGrouped}\ N_2)),
\end{aligned}$$

with

$$\begin{aligned}
&N_1^{\tilde{\exists}_m\forall_j.\ j<|\mathsf{GC}_i|\to c(\mathsf{GC}_{i,j})=m} :=\\
&\quad \tilde{\exists}^+\ i\ (\mathsf{GroupByCols}\ c\ r\ n\ i\ (Lem_{\mathsf{Trans}}\ i\ |\mathsf{GC}|\ r\ H_i\ H)^{i\leq r})^{\forall_j.\ j<|\mathsf{GC}_i|\to c(\mathsf{GC}_{i,j})=i}\\
&N_2^{k\cdot|\mathsf{GC}|\leq n} :=\\
&\quad Lem_{\mathsf{Trans}}\ k\cdot|\mathsf{GC}|\ k\cdot r\ n\ (Lem_{\leq \mathsf{Times}}\ k\ k\ |\mathsf{GC}|\ r\ \mathsf{AxT}\ H)^{k\cdot|\mathsf{GC}|\leq k\cdot r}\ H_2
\end{aligned}$$

□

Extracted Program

For an easier comparison with P_{FPH}, we make the same abbreviations, wherever this is possible. Thus, we take $f^{\mathtt{N}\Rightarrow\mathtt{L}\,\mathtt{N}}$, $g^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{LN})\Rightarrow\mathtt{LN}}$, $h^{\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}}$ and $\mathcal{R}_1 : \mathcal{R}^{\sigma_1}_{\mathtt{L}(\rho)}$ and $\mathcal{R}_2 : \mathcal{R}^{\sigma_2}_{\mathtt{N}}$, with

$$\rho := \mathtt{LN},\ \sigma_1 := \mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{LN})\Rightarrow\mathtt{LN}$$
$$\sigma_2 := \mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}$$

In addition, we use the functionals $C^{(\mathtt{N}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN})\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}}$ and $p^{\mathtt{N}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}}$ and we have one more recursion operator $\mathcal{R}_3 : \mathcal{R}^{\sigma_3}_{\mathtt{N}}$ coming from the proof of "dnColLists", where

$$\sigma_3 := (\mathtt{N}\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN})\Rightarrow\mathtt{LN}\Rightarrow\mathtt{LN}.$$

The program corresponding to (FPH_{dn}) is:

$$P_{\mathrm{FPH}_{dn}} := \lambda k, n, r, c.\ \mathcal{R}_3\ n\ F_3\ G_3\ (\lambda m, l.\ [\mathsf{if}\ (c(m) < r)\ l\ (\mathsf{nil}\ \mathtt{N})])\\ (\mathcal{R}_1\ \mathsf{GC}\ F_1'\ G_1'\ k\ (\lambda i.\ \mathsf{GC}_i)),\ \text{where}$$

$$\begin{aligned}
F_1' &:= \lambda k', f.\ (\mathsf{nil}^{\mathtt{N}})\\
G_1' &:= \lambda l, ll, g, k', f.\ \mathcal{R}_2\ k'\ F_2'\ G_2'\ M'\\
F_2' &:= \lambda i, n_0, n_1, l_1, l_2.\ [\mathsf{if}\ i\ l_2\ (\lambda n_2.\ l_1)]\\
G_2' &:= \lambda i, h, i', n_0, n_1, l_1, l_2.\ [\mathsf{if}\ i'\ l_2\ (\lambda n_2.\ h\ n_2\ n_0\ n_1\ l_1\ l_2)]\\
M' &:= |l|\ (k' \cdot |ll|)\ |\mathsf{flatten}(ll)|\ (f\,0)\ (g\ k'\ \lambda i.\ f\ (\mathsf{S}i))\\
F_3 &:= \lambda p.\ p\ 0\\
G_3 &:= \lambda n', C, p, l.\ C\,p\,(p\,\mathsf{S}n'\,l)
\end{aligned}$$

$P_{(\mathrm{FPH}_{dn})}$ **vs.** P_{FPH}**.** The base and step functions for $\mathcal{R}_1$ and $\mathcal{R}_2$ ($F_i' = F_i, G_i' = G_i,\ i = 1, 2$) are identical in the case of F_{FPH} and $P_{\mathrm{FPH}_{dn}}$. We observe that the last subterm is $P_{\mathrm{FPH}}\,k\,n\,r\,c$ and thus have

$$P_{(\mathrm{FPH}_{dn})} := \lambda k, n, r, c.\ \mathcal{R}_3\ n\ F_3\ G_3\ (\lambda i, l.\ [\mathsf{if}\ (c(i) < r)\ l\ (\mathsf{nil}\ \mathtt{N})])(P_{\mathrm{FPH}}\,k\,n\,r\,c)$$

On the one hand, since we have changed the assumption formula to a (classically equivalent) *relevant* definite form, we have introduced additional $\bot$-predicates. This triggers an application of Lemma 3.6 and as a consequence, "Case Distinction" is necessary, resulting in the test on whether $c(i) < r$.

On the other hand, as we have seen, using the "dnColLists" lemma in order to prove (FPH_{dn}) introduces an additional recursion, since the induction needed to prove the lemma is captured by the supplementary $\bot$.

We simulate below the runs of $P_{\mathrm{FPH}_{dn}}$ on generic values for k, n, r, c.

Analysis of the extracted program. Unfolding the runs of $P_{\mathrm{FPH}_{dn}}$ we have:

$$P_{\mathrm{FPH}_{dn}}\,k\,n\,r\,c := \mathcal{R}_3\ n\ F_3\ G_3\ (\lambda m, l.\ [\mathsf{if}\ (c(m) < r)\ l\ (\mathsf{nil}\ \mathtt{N})])(P_{\mathrm{FPH}}\,k\,n\,r\,c)$$

Due to $\mathcal{R}_3$, we need to distinguish on the values of n:

- $n = 0$. Then $\mathcal{R}_3\ 0\ F_3\ G_3 = F_3 = \lambda p.\ p\ 0$, thus

$$P_{\mathrm{FPH}_{dn}}\,k\,0\,r\,c := \mathsf{if}\ (c(0) < r)\,\mathsf{then}\,(P_{\mathrm{FPH}}\,k\,0\,r\,c)\,\mathsf{else}\,(\mathsf{nil}^{\mathtt{N}})$$

- $\mathsf{S}n$. In this case, we have $\mathcal{R}_3\ \mathsf{S}n\ F_3\ G_3 = G_3\ n\ (\mathcal{R}_3\ n\ F_3\ G_3) = (\lambda m, C, p, l.\ C\,p\,(p\,\mathsf{S}m\,l)) = \lambda p, l.\ (\mathcal{R}_3\ n\ F_3\ G_3)\,p\,(p\,\mathsf{S}n\,l)$ and therefore

$$P_{\mathrm{FPH}_{dn}}\,k\,\mathsf{S}n\,r\,c := (\mathcal{R}_3\ n\ F_3\ G_3)\,(\lambda m, l.\ [\mathsf{if}\ (c(m) < r)\ l\ (\mathsf{nil}^{\mathbb{N}})])\\ (\mathsf{if}\ (c(\mathsf{S}n) < r)\ \mathsf{then}\ (P_{\mathrm{FPH}}\,k\,\mathsf{S}n\,r\,c)\ \mathsf{else}\ (\mathsf{nil}^{\mathbb{N}}))$$

$\mathcal{R}_3$ is extracted from the proof of (dnColLists). The purpose of this lemma is to validate the fact that the list constructed by GC has at most as many objects as r (the number of colors). By double negating this specification we incorporate this test in the extracted program, such that the search for the list of k objects is performed as long as this condition is fulfilled.

6.2.3 Formalization of the Erdös-Szekeres Theorem

We first show the following lemma, which we need in the fourth step of the proof. The properties that it specifies are not trivial, as one is tempted to think, by considering that if $s_i \leq s_j$ then $f_{s,\mathsf{S}i} \subset f_{s,\mathsf{S}j}$. This is not necessarily the case, as illustrated by the (counter-) Example 5 , in which $f_{s,\mathsf{S}i}$ is computed according to our definition of $\mathsf{Bar}f$.

Lemma 6.3. *Let s be a finite sequence of natural numbers and for each $i < |s|$ let $f_{s,\mathsf{S}i}$ and $g_{s,\mathsf{S}i}$ be a longest increasing, respectively decreasing, sequences ending in s_i. Then*

$$\forall_{i,j}.\ i < j \to s_i \leq s_j \to |f_{s,\mathsf{S}i}| < |f_{s,\mathsf{S}j}| \qquad (s_i \leq s_j)$$

$$\forall_{i,j}.\ i < j \to s_j \leq s_i \to |g_{s,\mathsf{S}i}| < |g_{s,\mathsf{S}j}| \qquad (s_j \leq s_i)$$

Proof. The lemma does not have computational content, so we give only its informal proof, with the intention of validating its claims.

We show only $(s_i \leq s_j)$, since $(s_j \leq s_i)$ is symmetrical.

Assume that for fixed s, i and j, $H_1 : i < j$ and $H_2 : s_i \leq s_j$. We want to prove that $|f_{s,\mathsf{S}i}| < |f_{s,\mathsf{S}j}|$.

Let $\widehat{f}_{s,i}$ be the result of filtering $\mathsf{Bar}f(s,i)$ according to the criterion $\lambda l.\ \mathsf{last}(l) \leq s_i$. Then

- For any k, if $k > i$ and under the assumption H_2, we have $f_{s,\mathsf{S}i} \in \widehat{f}_{s,k}$. Therefore, $|f_{s,\mathsf{S}i}| \leq |\widehat{f}_{s,k}\ \mathsf{maxwrt}\ \lambda l|l||$.

- In case $k = j$, $\widehat{f}_{s,j}$ is the filtered list used to construct $f_{s,\mathsf{S}j}$.

Since $f_{s,\mathsf{S}j} = (\widehat{f}_{s,k}\ \mathsf{maxwrt}\ \lambda l|l|) :: s_{\mathsf{S}j}$, it follows that $|f_{s,\mathsf{S}i}| + 1 \leq |f_{s,\mathsf{S}j}|$. □

The Erdös-Szekeres Theorem. The formula that we prove, corresponding to ($\tilde{\forall}_3$) by which we have rewritten in Section 6.2.1 $\tilde{\forall}$ in (ES $_{\tilde{\forall}}$) is:

$$\forall_{n^{\mathtt{N}},s^{\mathtt{L}}}.\ |s| = \mathsf{S}n^2 \to \tilde{\exists}_i.\ |\overline{f}_{s,i}| \le n \to |\overline{g}_{s,i}| \le n \to \bot, \tag{ES}$$

Formalized proof. We fix n and s and assume

$$H_1 : |s| = \mathsf{S}n^2 \text{ and } H_2 : \forall_i.\ (|\overline{f}_{s,i}| \le n \to |\overline{g}_{s,i}| \le n \to \bot) \to \bot.$$

The goal is $\bot$.

We use (FPH$_{dn}$) with $k = 1$, $n^2 + 1$ objects (the number of elements in the sequence s), n^2 colors (the number of distinct pairs $(f_{s,i}, g_{s,i})$) and the coloring as defined by (6.8), where we take (a, b) to be $(|f_{s,i}|, |g_{s,i}|)$. In order to apply (FPH$_{dn}$), we first verify that its requirements are fulfilled.

In what follows, we will use the assumptions:

$$H_3 : \mathsf{col}(|f_{s,i}|, |g_{s,i}|) \le n^2 \text{ and } H_4(h, i) : |\overline{h}_{s,i}| \le n, \text{ with } h \in \{f, g\}.$$

- $0 < \mathsf{col}(|f_{s,i}|, |g_{s,i}|)$ follows from the definition (6.8) of col and the provable property $0 < |f_{s,i}|, |g_{s,i}|$.
- We need to show $\forall_i\ \tilde{\neg}\ \tilde{\neg} H_3$. We have the derivation:

$$\dfrac{\dfrac{\dfrac{}{H_2\ i} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{col}(|f_{s,i}|, |g_{s,i}|) \le n^2}\ (\mathsf{ColUB})\ |f_{s,i}|\ |g_{s,i}|}{\bot}\ \neg H_3}{H_4(f,i) \to H_4(g,i) \to \bot}\ H_4(f,i)}{\bot}\ H_4(g,i)}{}$$

- $n^2 \le n^2 + 1$ is trivial.

FPH$_{2,dn}$ provides us therefore with i and j be such that

$$i < j \le n^2 + 1 \wedge \mathsf{col}(|f_{s,i}|, |g_{s,i}|) = \mathsf{col}(|f_{s,j}|, |g_{s,j}|). \tag{H_5}$$

Assume $H_4(f, i)$, $H_4(g, i)$, $H_4(f, j)$ and $H_4(g, j)$. By ($2\mathsf{Col}_=$) we infer

$$|f_{s,i}| = |f_{s,j}| \wedge |g_{s,i}| = |g_{s,j}|. \tag{H_6}$$

We make the case distinction on s_i and s_j.

Case $s_i \le s_j$**.** By Lemma$(s_i \le s_j)$, it follows that $|f_{s,i}| < |f_{s,j}|$ and by the left conjunct of (H_6) this implies $|f_{s,i}| < |f_{s,i}| = \mathrm{F}$. Using efq we infer $\bot$.

Case $s_j \leq s_i$ This case follows the same argument, using $(s_j \leq s_i)$ and the right conjunct of (H_6). □

The λ-term corresponding to the proof of ES is M_{ES}, where the assumption variables are typed as follows:

$$v_1^{\forall_i.\, \mathsf{S}i<|l|\to l_{\mathsf{S}i}<l_i}, v_2^{\tilde{\exists}_m \forall_i.\, i<|l|\to \mathsf{col}(s,n,l_i)=m}, v_3^{\forall_i.\, i<|l|\to m=\mathsf{col}(s,n,l_i)}$$
$$w_h^{|\mathsf{Bar}h(s,\mathsf{S}i)|\leq n} w_1^{1<|l|}, w_2^{\mathsf{col}(s,n,i)<n^2\to\bot}, u_{4,h\,k}^{H_4(h,k)}, h \in \{f,g\} i \in \{0,1\},$$
$$c_{siLesj} : s_{l_1} \leq s_{l_0} \text{ and } c_{sjLesi} : s_{l_0} \leq s_{l_1}$$

We implicitly use the transitivity for $0 < 1 < |l|$ and the lemmas:

$$Lem_{\neq} :\forall_{i,j}.\, \mathsf{S}i \leq j \to i = j \to \bot$$
$$Lem_{Case\leq} :\forall_{n_1,n_2}.\, (n_1 \leq n_2 \to \bot) \to (n_2 \leq n_1 \to \bot) \to \bot$$

and make the abbreviations:

$$Lem_1 :\forall_{i,s,n}.\, (\mathsf{ColUB})(|\mathsf{Bar}f(s,\mathsf{S}i)|\ |\mathsf{Bar}g(s,\mathsf{S}i)|)$$
$$Lem_2 :\forall_{i,j,s,n}.\, (2\mathsf{Col}_=)\ |\mathsf{Bar}f(s,\mathsf{S}i)|\ |\mathsf{Bar}g(s,\mathsf{S}i)|\ |\mathsf{Bar}f(s,\mathsf{S}j)|\ |\mathsf{Bar}g(s,\mathsf{S}j)|$$

With this,

$$M_{\mathrm{ES}} := \lambda n, s, u_1^{H_1}, u_2^{H_2}.\, T_1\ \lambda l, w_1, v_1, v_2.\, (Lem_{Case\leq}\ l_{1,s}\ l_{0,s})\ T_{21}\ T_{22},$$

where

$$T_1 := (\mathrm{FPH}_{dn})\ 1\ \mathsf{S}n^2\ n^2\ \mathsf{col}(s,n)\ \lambda i, w_2.\, u_2\ i\ \lambda w_f, w_g.\, w_2\ (Lem_1\ i\ s\ n\ w_f\ w_g)\ \mathrm{T}$$
$$T_{21} := \lambda c_{siLesj}.\, u_2\ l_1\ \lambda u_{4,f1}, u_{4,g1}.\, u_2\ l_0\ \lambda u_{4,f0}, u_{4,g0}.\, T_3(f, (s_i \leq s_j))$$
$$car(Lem_2\ l_1\ l_0\ s\ n\ u_{4,f1}\ u_{4,g1}\ u_{4,f0}\ u_{4,g0} \tilde{\exists}^-_{1,1}\ l\ s\ n\ v_2\ T_4)$$
$$T_{22} := \lambda c_{sjLesi}.\, u_2\ l_1\ \lambda u_{4,f1}, u_{4,g1}.\, u_2\ l_0\ \lambda u_{4,f0}, u_{4,g0}.\ T_3(g, (s_j \leq s_i))$$
$$cdr(Lem_2\ l_1\ l_0\ s\ n\ u_{4,f1}\ u_{4,g1}\ u_{4,f0}\ u_{4,g0}\ \tilde{\exists}^-_{1,1}\ l\ s\ n\ v_2\ T_4)$$
$$T_3(h, D) := Lem_{\neq}\ |\mathsf{Bar}h\ s\ \mathsf{S}l_1|\ |\mathsf{Bar}h\ s\ \mathsf{S}l_0|\ (D\ s\ l_1\ l_0\ (v_1\ 0)\ w_1\ c_D),$$
$$D \in \{(s_i \leq s_j), (s_j \leq s_i)\}$$
$$T_4 := \lambda m, v_3.\, Lem_{=Trans}\ \mathsf{col}(s,n,l_1)\ m\ \mathsf{col}(s,n,l_0)\ (v_3\ 1\ w_1)\ (v_3\ 0\ w_1)$$

6.2.4 Extracted Program

We consider the following terms

$$T_f(s,i) := (\mathsf{Bar}f(s,i)\mathsf{filter}(\lambda l.\, l_0 \leq s_i))\mathsf{maxwrt}(\lambda l.\, |l|)$$
$$T_g(s,i) := (\mathsf{Bar}g(s,i)\mathsf{filter}(\lambda l.\, s_i < l_0))\mathsf{maxwrt}(\lambda l.\, |l|).$$

$T_f(s,i)$ thus determines which list from $\overline{f}_{s,i}$ ends in an element smaller[3] than s_i and is of maximal length. $T_g(s,i)$ applies the dual operation for $\overline{g}_{s,i}$. These terms are the components by which the increasing or decreasing subsequence of length greater than n is computed. For $h \in \{f,g\}$,

$$|T_h(s,i)| = |h_{s,i}| \text{ and } \mathsf{S}|T_h(s,i)| = |h_{s,\mathsf{S}i}|.$$

Let us regard the extracted terms and for a better understanding leave first the program component corresponding to (FPH_{dn}) unfolded.

FPH_{dn} not expanded The term corresponding to $P_{\mathrm{FPH}_{dn}}$ in $P'_{(\mathrm{ES})}$ is left unexpanded. Denoted by cdnFPH, this term takes the arguments as specified in the proof: $k := 1, n := \mathsf{S}(n^2), r := n^2$ and the coloring $\lambda n_1.\ |T_f(l,n_1)| \cdot n + |T_g(l,n_1)| = \lambda n_1.\ |f_{l,n_1}| \cdot n + |g_{l,n_1}|$.

We leave the (Minlog) extracted term unchanged structurally, but make a few abbreviations in order to improve its readability. Let in the following $D_h(i,j)$, $h \in \{f,g\}$, abbreviate the term

$$\begin{array}{l}
\mathsf{if}\ (\mathsf{S}|T_f(l,j)| \le n) \\
\quad \mathsf{if}\ (\mathsf{S}|T_g(l,j)| \le n) \\
\quad\quad \mathsf{if}\ (\mathsf{S}|T_f(l,i)| \le n) \\
\quad\quad\quad \mathsf{if}\ (\mathsf{S}|T_h(l,i)| \le n) \\
\quad\quad\quad\quad 0 \\
\quad\quad\quad\quad i \\
\quad\quad\quad i \\
\quad\quad j \\
\quad j.
\end{array}$$

and $D_1(i,j)$ the term

$$\begin{array}{l}
\mathsf{if}\ (\mathsf{S}|T_f(l,i)| \le n) \\
\quad \mathsf{if}\ (\mathsf{S}|T_g(l,i)| \le n) \\
\quad\quad j \\
\quad\quad i \\
i
\end{array}$$

[3]Since the lists are reversed, the elements are in fact monotonically decreasing and the comparison with s_i is made considering the head of the list

With $\mathsf{S}|T_h(s,i)| = |h_{s,\mathsf{S}i}|, h \in \{f,g\}$, we can interpret $D_h(i,j)$ as a test on whether either of $f_{s,\mathsf{S}i}$ or $g_{s,\mathsf{S}i}$ is of length at least n. Should the answer be in either case positive, the corresponding index is returned; otherwise, $D_h(i,j)$ returns 0. Likewise, $D_1(i,j)$ returns i if $|h_{s,\mathsf{S}i}| > n$ for $h \in \{f,g\}$ and j if this is not the case.

With these abbreviations, the term extracted from the Erdös-Szekeres Theorem, parameterized by the term cdnFPH, is

$$\begin{aligned} P'_{(\mathrm{ES})} := \lambda n, l.\ \mathsf{cdnFPH}\ 1\ (\mathsf{S}\, n^2)\ n^2\ (\lambda n_1.\ |T_f(l,n_1)| \cdot n + |T_g(l,n_1)|) \\ (\lambda n_1, n_2.\ D_1(n_1,n_2)) \\ (\lambda l', C.\ \mathcal{R}\ l_{l'_1}\ F\ G\ l_{l'_0}\ D_g(l'_0,l'_1)\ D_f(l'_0,l'_1)), \end{aligned}$$

with $C^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$ and $\mathcal{R} := \mathcal{R}^{\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}}_{\mathtt{N}}$ having the base and step case

$$\begin{aligned} F :=& \lambda i, n_0, k.\ n_0, \text{ respectively} \\ G :=& \lambda n_0, g, k, m_1, i.\ [\mathsf{if}\ n_0\ i\ \lambda m_2.\ g\, m_2\, m_1\, i] \end{aligned}$$

Here, l' corresponds the list returned by FPH_{dn} containing the objects of the same color. For ES we are interested in a list of two elements, hence the occurrence in the extracted term of l'_0, l'_1.

Moreover, notice that G is in tail recursive form, since the operator corresponding to the recursion step g occurs in tail position.

In what follows, we expand the term associated with the proof of (FPH_{dn}). This terms is coupled with the computational terms associated to the proof of ES. We can now trace the computation made by (FPH_{dn}) in order to select the two components that have the same "color".

FPH_{dn} expanded We take $p'^{\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}}$, $C'^{(\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}}$, $g'^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$ and $h'^{\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}}$ and let $\mathsf{col_i} := \mathsf{col}(l,n,i)$.

In the program extracted from the proof of the Erdös-Szekeres Theorem, we use the following terms:

$$\begin{aligned} CA :=& \mathsf{ChangeAt}(\mathsf{GroupbyCols}(\lambda i.\ \mathsf{col_i}, n^2),\ \mathsf{col_{Sn^2}},\ \mathsf{S}n^2) \\ D_2 :=& D_1(CA_{m,1}, D_1(CA_{m,0}, 0)). \end{aligned}$$

The extracted program is:

$$\begin{aligned} P_{(\mathrm{ES})} :=& \lambda n, l.\ \mathcal{R}_3\ n^2\ F_3\ G_3(\lambda n_1, n_2.\ D_1(n_1,n_2)) \\ & D_1(\mathsf{S}n^2, \mathcal{R}_1\ CA\ F_1\ G_1\ 1\ (\lambda m.\ \mathcal{R}\ l_{CA_{m,1}}\ F\ G\ l_{CA_{m,0}}\ D_2\ D_2)) \end{aligned}$$

where

$$
\begin{aligned}
F_1 &:= \lambda k', f.\ 0\\
G_1 &:= \lambda l', ll, g', k', f.\ \mathcal{R}_2\ k'\ F_2\ G_2\ M\\
M &:= |l'|\ k \cdot |ll|\ |\mathsf{flatten}(ll)|\ (f\,0)\ (g'\ k\ \lambda i.\ (f\ \mathsf{S}i))\\
F_2 &:= \lambda i, n_0, n_1, m_1, m_2.\ [\mathsf{if}\ i\ m_2\ (\lambda n_2.\ m_1)]\\
G_2 &:= \lambda i, h', i', n_0, n_1, m_1, m_2.\ [\mathsf{if}\ i'\ m_2\ \lambda n_1.\ h'\ n_1, n_0, n_1, m_1, m_2]\\
F_3 &:= \lambda p'.\ p'\,0\\
G_3 &:= \lambda n_1, C', p', k.\ C'\ p'\ (p'\ \mathsf{S}n_1\ k)
\end{aligned}
$$

and

$$
\begin{aligned}
F &:= \lambda i, n_0, k.\ n_0\\
G &:= \lambda n_0, g, k, m_1, i.\ [\mathsf{if}\ k\ i\ (\lambda m_2.\ g\ m_2\ m_1\ i)]
\end{aligned}
$$

and the recursion operators occurring in the term have the following types:

$$(\mathcal{R}_1)^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}_{\mathtt{LLN}}, (\mathcal{R}_2)^{\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}}_{\mathtt{N}}, (\mathcal{R}_3)^{(\mathtt{N}\Rightarrow\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}\Rightarrow\mathtt{N}}_{\mathtt{N}}$$

As expected, the programs reflect closely the initial classical proofs. The use of FPH to derive the contradiction the Erdös-Szekeres Theorem is reflected in the fact that the main computations of P_{ES} are performed by the term associated to $P_{\mathrm{FPH}_{dn}}$. As long as sequences $\overline{f}_{s,i}$ and $\overline{g}_{s,i}$ of length at most n are found and used to validate FPH, the search continues recursively, since for such sequences we reach a contradiction. As a consequence of taking $k := 1$ in FPH_{dn} the corresponding program outputs two objects of identical color instead of a list of k objects. The types are therefore simplified, as it is the case with the program constants C, g and h.

6.3 Summary of the Chapter

In this chapter we have analyzed the proof of the Erdös-Szekeres Theorem (ES), specifying that any given sequence of n^2+1 elements contains a monotonically increasing/decreasing subsequence of length at least $\mathsf{S}n$. Since the proof of the theorem relies on the result known as the Pigeonhole Principle (FPH), we have also investigated an indirect proof of this lemma.

The chapter serves as a case study for the application of the refined A-Translation in order to extract the algorithms from the classical proofs. We were interested in investigating not only the outcome of the transformation method, but mostly on which conditions need to be fulfilled in order to

apply it. Due to the restrictions imposed on the definite/goal formulas and to the fact that we need to limit ourselves to NA^ω, special care had to be taken with respect to negations. This concerns on the one hand the proof of FPH, which cannot be shown in minimal logic, unless it is weakened by the insertion of double negations. Further, FPH cannot be used for the proof of ES unless it is adapted by further insertion of double negations. We have addressed all these issues in this chapter and have discussed in detail our choice for which atoms are double negated. We have illustrated the effects of this on the extracted program and have commented on the fact that the negations are the actual carriers of the computational content. Therefore, each proof component that performs actual calculations needs to have the logical falsity in the specification, such that there is a correlation between the restriction of the working system to NA^ω and the application of the refined A-Translation. More precisely, $\bot$ is the element that helps us recover the computational content and its insertion is necessary on the one hand in order to avoid $\mathsf{efq}^\bot$ and on the other in order to recover the computational content. This is to be expected, since $\mathsf{efq}^\bot$ cannot provide a constructive information, so the logical negation needs to be shifted in the positions where it is relevant for the extracted term. In Chapter 8 we give a systematic procedure by which we detect which atoms need to be double negated in order to carry out the proofs in NA^ω.

Another idea for further investigation is to consider the alternative representations of $\tilde{\vee}$, such as $(\tilde{\vee}_1)$. It would be interesting to compare the programs extracted from this formalization and the ones that we have obtained by $(\tilde{\vee}_3)$ presented in this chapter.

Chapter 7

Dickson's Lemma

Dickson's Lemma (DL) represents a key component in guaranteeing the termination of Buchberger's Algorithm for computing the Gröbner basis of a given ideal. Due to the importance of the Gröbner Basis Theory - which has applications in a wide spectrum of domains - Dickson's Lemma and Buchberger's Algorithm have been extensively analyzed from the perspective of program extraction/verification. Such is the case with systems like Mizar (see (Schwarzweller, 2005)), Coq (see (Coquand and Persson, 1999)) and ACL2 (see (Martin-Mateos et al., 2003)).

However, whereas in such systems (e.g. Coq) the emphasis is on verifying that the given algorithm is correct, our purpose is to extract the program from the *classical proof* of DL and for this we use the refined A-Translation method. As we will point out, one cannot associate a computable functional to DL directly, but this is possible for a Π^0_2-corollary.

In our literature survey, we have come across various formulations and we investigate in Section 7.3 the connection among them. The proof that we will analyze in this chapter relies on the Minimum Principle, which we present in Section 7.2. We give the formalization and extracted program of a Π^0_2-corollary of DL in Section 7.4.1, while emphasizing in particular the use of negations and the connection between the refined A-Translation and the continuation passing style. In Section 7.4.2 we sketch a generalization of this corollary, which is also a Π^0_2 formula and thus suited for the A-Translation. Future work ideas are gathered in Section 7.5 and in the concluding Section 7.6, and are grouped in two research directions: on one hand we are interested in the applications of Dickson's Lemma and on the other in the strength of the analyzed corollary with respect to the statement proving the termination of Buchberger's Algorithm.

7.1 Terminology

In this section we overview the main notions related to the Gröbner Basis Theory, which we will further need when analyzing the different versions of Dickson's Lemma. We also need the terminology when discussing the role played by Dickson's Lemma with respect to Buchberger's Algorithm.

Let in the following K be a field. We consider the ring of polynomials in the variables $X_1, \ldots, X_n$ with coefficients in K and denote it by $K[X_1, \ldots, X_n]$ or $K[\underline{X}]$.

Let $p \in K[\underline{X}]$ be a multivariate polynomial over K, i.e. a polynomial in the variables $X_1, \ldots, X_n$,

$$p = \sum_{\alpha_1, \ldots \alpha_n \in \mathtt{N}} a_{\alpha_1, \ldots \alpha_n} X_1^{\alpha_1} \ldots X_n^{\alpha_n} =: \sum_{\underline{\alpha} \in \mathtt{N}^n} a_{\underline{\alpha}} \underline{X}^{\underline{\alpha}}.$$

Definition 7.1 (Monomial, term, coefficient)**.** *We have the following terminology*[1]

1. *The power product* $m_{\underline{\alpha}} = X_1^{\alpha_1} \cdot \ldots \cdot X_n^{\alpha_n} =: \underline{X}^{\underline{\alpha}}$*, or* m *in short, is called* monomial *in the indeterminates* $X_1, \ldots, X_n$*. Let in the following* M *denote the set of all monomials and* $M(p)$ *denote the monomials occurring in the polynomial* p*.*

2. $t = a_{\underline{\alpha}} \underline{X}^{\underline{\alpha}}, a_{\underline{\alpha}} \neq 0$*, is a* term *of* p*.*

3. $\mathrm{coeff}(m, p) := a_{\underline{\alpha}} \in K$ *is called the* coefficient *of the term* t *in the polynomial* p*. We have:* $t = \mathrm{coeff}(m, p) \cdot m$*.*

Example 6. *Let* $p = 2x^2y^3 + 7xy^4 + 3y \in \mathbb{N}[x, y]$*.*

- $M(p) = \{x^2y^3, xy^4, y\}$ *are the* monomials *occurring in* p
- $\mathrm{coeff}(xy^4, p) = 7$
- *The* terms *in* p *are* $2x^2y^3$*,* $7xy^4$ *and* $3y$*.*

Definition 7.2 (Polynomial/Monomial degrees)**.** *Let* $X_1^{\alpha_1} \cdot \ldots \cdot X_n^{\alpha_n}$ *be a monomial in* $K[\underline{X}]$ *and* $p \in K[\underline{X}]$ *be some polynomial in the field.*

- *The monomial total degree of* $\underline{X}^{\underline{\alpha}}$ *is defined as* $\alpha_1 + \alpha_2 + \ldots + \alpha_n$*.*
- *The polynomial total degree ,* $deg(p)$*, is taken to be the maximum of the monomial total degrees for the monomials of* p*.*

[1] It is often the case in the literature that the notion of *term* and that of *monomial* are exchanged. We use here the notions as given in (Cox et al., 1992).

Definition 7.3. *We consider the* natural partial order $\leq^n$ *on* $\mathtt{N}^n$*:*

$$(m_1, ..., m_n) \leq^n (k_1, ..., k_n) \text{ iff } m_i \leq k_i, \forall i \in 1, ..., n.$$

Definition 7.4. *A linear order* $\leq$ *on* M *(or, equivalently, on the set* $\mathbb{I} \subset \mathtt{N}$ *of exponents) is a* monomial order *if it satisfies the following conditions:*

(i) If $m_1 \leq m_2$ *then* $m_1 \cdot s \leq m_2 \cdot s$*, for all* $m_1, m_2, s \in M$*.*

(ii) $\leq$ *is a well-ordering, i.e. every nonempty subset of* $\mathbb{I}$ *has a smallest element under* $\leq$*.*

Example 7. $\leq$ *is a* lexicographical (or lexical) order *on the set* M *if the following holds:*
$X_1^{\alpha_1} \cdot \ldots \cdot X_n^{\alpha_n} \leq X_1^{\beta_1} \cdot \ldots \cdot X_n^{\beta_n}$ *iff* $(\alpha_1 \ldots \alpha_n) = (\beta_1 \ldots \beta_n)$ *or there exists some* $i \in \{1, ..., n\}$ *s.t.* $\alpha_j = \beta_j$, $1 \leq j < i$ *and* $\alpha_i < \beta_i$*.*

Remark 7.1. *The lexicographical order is a monomial order.*

From now on, when otherwise not specified, the default order is taken to be the lexicographical order.

Definition 7.5. *Let* p *be a polynomial and* $a_{\underline{\alpha}}$ *the coefficients of its terms. Let further* $\underline{\beta} := max\{\underline{\alpha} \in \mathtt{N}^{\mathtt{N}}, a_{\underline{\alpha}} \neq 0\}$*.*

- lm(p)$= \underline{X}^{\underline{\beta}}$ *is called the* leading monomial *of* p *(also referred to in the literature as the leading power product of* p*)*
- lt(p) $= a_{\underline{\beta}} \underline{X}^{\underline{\beta}}$ *is referred to as the* leading term *of* p
- lc(p) $= a_{\underline{\beta}}$ *is the* coefficient of the leading monomial *of* p*.*

For I a set of polynomials, we denote by *lt(I)* the set of leading terms of the elements of I, i.e.

$$lt(I) = \{lt(p), \text{ for some } p \in I\}.$$

Definition 7.6. *Let* $m_1, m_2 \in M$*. We say that* m_1 *divides* m_2 *and write* $m_1 | m_2$ *if there exists an* $s \in M$ *s.t.* $m_1 \cdot s = m_2$*.*

Remark 7.2. *The notions* leading term *and* leading monomial *can be used interchangeably, since we can assume without loss of generality that the leading term has coefficient* 1*. For this we divide the polynomial by its leading coefficient.*

Definition 7.7 (Monomial Ideal)**.** *Let $\mathbb{I} \subset \mathtt{N}^n, n \in \mathtt{N}$ be a set of indices, possibly infinite.*

1. *We call B a* monomial basis *of an ideal, if it is a basis consisting only of monomials, i.e. if $B = \{\underline{X}^{\underline{\alpha}}, \underline{\alpha} \in \mathbb{I}\}$.*
2. *An ideal $I \subset K[\underline{X}]$ generated from a monomial basis is referred to as a* monomial ideal*.*

Remark 7.3. *If $B = \{\underline{X}^{\underline{\alpha}}, \underline{\alpha} \in \mathbb{I}\}$ is a monomial basis for I, then all $p \in I \subset K[\underline{X}]$ are of the form*

$$p = \sum_{\underline{\alpha} \in \mathbb{I}} p_{\underline{\alpha}} \underline{X}^{\underline{\alpha}}, p_{\underline{\alpha}} \in K[\underline{\alpha}].$$

We write $I =< \underline{X}^{\underline{\alpha}}, \underline{\alpha} \in \mathbb{I} >$.

7.2 The Minimum Principle

In this section we overview the generalized induction scheme in its classically equivalent form known as the "Minimum Principle". We will use this in the proof of the equivalent statements of Dickson's Lemma and in Section 7.4 in the independent proof of the Π^0_2-corollary.

Let $m^{\mathtt{N} \Rightarrow \mathtt{N}}$ be a measure function and P a predicate. The following non-constructive axiom is known as the "Minimum Principle":

$$\forall_{m,P}.\ \tilde{\exists}_x P(x) \to \tilde{\exists}_x.\ P(x) \mathbin{\tilde{\wedge}} \forall_y.\ m(y) < m(x) \to \neg P(y), \qquad \text{(MinPr)}$$

with $\tilde{\wedge}$ and $\tilde{\exists}$ the classical (weak) operators as given in Section 2.2.

(MinPr) states that any non-empty set has a minimal element and is classically equivalent to the course-of-values induction. To see this, we abbreviate $\forall_y.\ m(y) < m(x) \to Q$ to $\forall_{y|m(y)<m(x)} Q$ and for readability omit the implicit quantification $\forall_{P,m}$. Using contraposition (in the step marked below with $(\circ)$) and given that $\tilde{\exists}$ is only an abbreviation, we have the following chain of (classically valid) transformations:

$$\begin{aligned}
&\tilde{\exists}_x P(x) \to \tilde{\exists}_x.\ P(x) \mathbin{\tilde{\wedge}} \forall_{y|m(y)<m(x)} \neg P(y) \equiv \\
\equiv&(\forall_x \neg P(x) \to \bot) \to \forall_x \neg (P(x) \mathbin{\tilde{\wedge}} \forall_{y|m(y)<m(x)} \neg P(y)) \to \bot \\
\overset{(\circ)}{\equiv}&\forall_x \neg (P(x) \mathbin{\tilde{\wedge}} \forall_{y|m(y)<m(x)} \neg P(y)) \to \forall_x \neg P(x) \\
\equiv&\forall_x (P(x) \to \forall_{y|m(y)<m(x)} \neg P(y) \to \bot) \to \forall_x \neg P(x) \\
\equiv&\forall_x (\forall_{y|m(y)<m(x)} \neg P(y) \to \neg P(x)) \to \forall_x \neg P(x).
\end{aligned}$$

By taking $R(x) := \neg P(x)$, this can be rewritten to

$$\forall_x.\, \forall_x(\forall_{y|m(y)<m(x)} R(y) \to R(x)) \to R(x),$$

which is *the generalized induction scheme* or, in the case when the measure function is the identity function on $\mathtt{N}$, the *course-of-values induction*.

Associated computational content. For $\tau(x) = \alpha$ and $\tau(m) = \alpha \to \mathtt{N}$, the general induction scheme

$$\mathsf{GInd}_{m,x,R} := \forall_x(\forall_{y|m(y)<m(x)} R(y) \to R(x)) \to R(x)$$

has the type

$$\tau(\mathsf{GInd}_{m,x,R}) = (\alpha \to \mathtt{N}) \to \alpha \to (\alpha \to (\alpha \to \tau) \to \tau) \to \tau.$$

The associated general recursion operator is $\mathcal{F}$, defined as in (Schwichtenberg and Wainer, 2011),

$$\mathcal{F}(m, x, G) = Gx(\lambda y.\mathsf{if}\ (m(y) < m(x))\ \mathsf{then}\ \mathcal{F}(m, y, G)\ \mathsf{else}\ \varepsilon) \tag{7.1}$$

7.3 Equivalent Formulations

While surveying the literature on the topic "Dickson's Lemma in the context of the Gröbner Bases Theory", we have come across various formulations and approaches for formalizing the lemma. We have investigated how these formulations connect, in order to determine the strength of the Π^0_2-corollary of DL presented in (Berger et al., 2002), which represented the staring point of our survey.

The first formulation of the property known as "Dickson's Lemma" has been published in (Dickson, 1913) and is as follows:

Lemma 7.1. *Any set S of functions of the type*

$$F \;=\; x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n},\ e\text{'s integers } \geq 0 \tag{7.2}$$

contains a finite number of functions $F_1, \ldots, F_j$ such that each function F of the set S can be expressed as a product $F_i \cdot f$, where f is of the form (7.2)*, but is not necessarily in the set S.*

Lemma 7.2. *The following statements are equivalent:*

(1) *Let* $S \subset \mathtt{N}$ *be unbounded,* $n \in \mathtt{N}$ *and* $\mathbf{f}_1, ..., \mathbf{f}_n : S \to \mathtt{N}$. *Then there exists* $M \subset S$ *unbounded s.t.*

$$\forall_{i,j \in M}.\ i < j \to \mathbf{f}_k(i) \leq \mathbf{f}_k(j), \tag{7.3}$$

for all $k = 1, ..., n$.

In other words, given a finite number of infinite sequences of natural numbers, there exists an unbounded set M *such that each of these sequence increases on* M. *(Berger et al., 2002; Raffalli, 2004)*

(2) *Given an infinite sequence* $\{f_k : k \in \mathtt{N}\}$ *of n-tuples of natural numbers, there exists* $M \subset \mathtt{N}$ *unbounded s.t.* $\forall_{i,j \in M}.\ i < j \to f_i \leq^n f_j$, *where* $\leq^n$ *is given by Definition 7.3. (Martin-Mateos et al., 2003)*

(3) *Let* $n \in \mathtt{N}$ *and let* $S := \{m_k : k \in \mathtt{N}\}$ *be an infinite sequence of monomials in the variables* $\{X_1, ..., X_n\}$. *There are finitely many monomials* $m_1, ..., m_l$ *in this sequence (for some* $l \in \mathtt{N}$*), s.t.* $\forall_{m_j \in S} \exists_{i \in \{1,...,l\}} m_i | m_j$.

(4) *Let* $S \subset T$, *with* $T \neq \emptyset$ *a set of monomials. Then there exists* $B \subset S$ *finite s.t. for all* $s \in S$ *there exists* $t \in B$ *with* $t|s$. *In this case we call "*$|$*"* *a* Dickson partial order *on T. (Becker and Weispfenning, 1993)*

(5) *Each monomial ideal has a finite basis. (Cox et al., 1992)*

The formulation (3) is very similar to (7.2), so all of the above are seen as variants of Dickson's Lemma.

A couple of remarks:

- (Berger et al., 2001) treats in detail a corollary of (1), in which 2 functions and a 2-element set M are considered. We will also analyze this case, while bringing some new insight on the formalization issues and on the resulting associated algorithm.
- The formulation (5) is used in (Cox et al., 1992) in the context of Gröbner Bases Theory. A simplification similar to the corollary presented in (Berger et al., 2001) is not possible and, most importantly, is not sufficient in order to prove the termination of Buchberger's Algorithm, or the other results in the theory.

We split the proof of Lemma 7.2 in two and show the equivalences as depicted in Figure 7.1. Due to our particular interest in (1) and (5), we prove their equivalence separately, as marked in the figure.

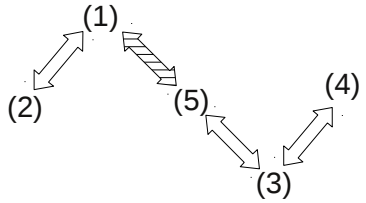

Figure 7.1: Equivalences showed for Lemma 7.2

Note that we can identify each monomial $\underline{X^e} = X_1^{e_1} \cdot X_2^{e_2} \cdot ... \cdot X_n^{e_n}$ with the tuple $\langle e_1, e_2, ..., e_n \rangle$ of the exponents, following the idea from (Martin-Mateos et al., 2003). This way,

$$\underline{X^e} | \underline{X^f} \text{ iff } \langle e_1, e_2, ..., e_n \rangle \leq^n \langle f_1, f_2, ..., f_n \rangle.$$

We can therefore reformulate (2) as:

$$\forall_S \exists_{M \subset S \text{ infinite}} \forall_{f_i, f_j \in M}. \, i < j \rightarrow f_i | f_j, \tag{2'}$$

where $S := \{f_k : k \in \mathtt{N}\}$ is an infinite set of monomials in the indeterminates $\mathbf{f_1}, ..., \mathbf{f_n}$.

Proof. (1) $\Leftrightarrow$ (2) (1) and (2) differ only in the formulation, as we can intuitively view the former as reading the matrix below and the latter its transpose. More precisely, for (1), the column k depicts the values of $\mathbf{f}_k$ on the unbounded set S. For (2) the same matrix has at the intersection of the line i and column k the exponent of the indeterminate $\mathbf{f}_k$ in the monomial f_i.

The bullets mark the generic points in which the k functions are proved to increase for at least 2 values (i and j) (for (1)) and, respectively, the case when at least for some i, j we have $f_i | f_j$ (for (2)):

	$\mathbf{f}_1$	$\cdots$	$\mathbf{f}_n$
f_1	$\cdot$	$\cdot$	$\cdot$
$\vdots$	$\cdots$	$\cdots$	$\cdots$
f_i	$\bullet$	$\bullet$	$\bullet$
$\vdots$	$\cdots$	$\cdots$	$\cdots$
f_j	$\bullet$	$\bullet$	$\bullet$
$\vdots$	$\cdots$	$\cdots$	$\cdots$

(3) $\Leftrightarrow$ (4) (4) is just a reformulation of (3).

(3) $\Rightarrow$ (5) Let $I = \langle \underline{X}^{\underline{\alpha}} \,|\, \underline{\alpha} \in \mathtt{N}^{\mathtt{N}} \rangle$ be a monomial ideal in $K[\underline{X}]$ and suppose that the system of generators, $S = \{\, \underline{X}^{\underline{\alpha}} \mid \underline{\alpha} \in \mathtt{N}^{\mathtt{N}} \,\}$, is infinite. By (3) there exists a finite set of monomials $B \subset S$, such that for all $\underline{X}^{\underline{\alpha}} \in S$ there exists $\underline{X}^{\underline{\beta}} \in B$ with $\underline{X}^{\underline{\beta}} | \underline{X}^{\underline{\alpha}}$. But then the monomial $\underline{X}^{\alpha}$ belongs to the ideal generated by B (the converse is in fact also provably true). Thus, the finite set B generates the same ideal as S and is a system of generators for I. By taking the linearly independent monomials in B we have the finite basis that we need.

(5) $\Rightarrow$ (3) Let $S := \{m_k : k \in \mathtt{N}\}$ and consider the ideal I generated by S. By (5), I has a finite basis B, so in particular any monomial in S is divisible by some monomial in B. There is therefore a subset B' of S of the same cardinality as B which generates all S. Thus, for all $s \in S$ there exist some $t \in B'$ with $t|s$.

(1) $\Leftrightarrow$ (5) We study the connection between the formulation in (Berger et al., 2002) and the one in (Cox et al., 1992) in detail, in order to understand whether the results from (Berger et al., 2001) can be connected with the applications of Dickson's Lemma in the Gröbner Basis Theory, and in particular, for Buchberger's Algorithm.

Correlation between (1) **and** (5)**.** Without loss of generality, we can restrict our attention to the simplified case of two functions for (1) and correspondingly monomials in two indeterminates for (5).

"$\Rightarrow$" Let $\mathbf{f}_1, \mathbf{f}_2 : S \to \mathtt{N}$, with $S \subset \mathtt{N}$ unbounded, and a monomial ideal $I = \langle f_k = X_1^{\mathbf{f}_1(k)} X_2^{\mathbf{f}_2(k)}, k \in \mathtt{N} \rangle$ be given. We want to construct a finite basis B for I.

By (1), there exists an unbounded set M, s.t. (7.3) is fulfilled. We select k_1 as the minimal element in M and have:

$$\forall_{i \in M} (\mathbf{f}_1(k_1), \mathbf{f}_2(k_1)) \leq^2 (\mathbf{f}_1(i), \mathbf{f}_2(i)).$$

Let $S_1 = S \setminus M$. We have two situations:

S_1 **finite.** Then for $S_{1,k_1} := S_1 \cup \{k_1\}$, we have

$$\forall_{j \in S} \, \exists_{i \in S_{1,k_1}} (\mathbf{f}_1(i), \mathbf{f}_2(i)) \leq^2 (\mathbf{f}_1(j), \mathbf{f}_2(j)),$$

i.e., every monomial in I is generated by the monomials in the set $\{X_1^{\mathbf{f}_1(i)}X_2^{\mathbf{f}_2(i)}, i \in S_{1,k_1}\}$. The linearly independent monomials in this set form the finite basis B.

S_1 **infinite.** We apply (1) with $S := S_1$ and obtain an infinite subset M_1 of S_1 for which

$$\forall_{i,j\in M_1}.i < j \rightarrow (\mathbf{f}_1(i), \mathbf{f}_2(i)) \leq^2 (\mathbf{f}_1(j), \mathbf{f}_2(j)).$$

By the Minimum Principle (MinPr) we obtain k_1 as the minimal element in M_1. As in the previous step, we take $S_2 = S_1 \setminus M_1$ and make a case distinction on whether S_2 is finite or infinite. In the first case, we construct B and in the second, we proceed further.

We continue this process until some $S_n := S_{n-1} \setminus M_{n-1}$ is finite, $n > 1$ and $S_0 := S, M_0 := M$. The finite basis for I is therefore

$$B \subset \{X_1^{\mathbf{f}_1(i)}X_2^{\mathbf{f}_2(i)}, i \in \{k_1, ..., k_n\} \cup S_n\}.$$

It remains to show that there is some n such that S_n is finite, i.e., that the process stops after n steps.

Suppose that this is not the case and that we can construct infinitely many such S_i's. For each S_i, we can find by hypothesis (1) an infinite set M_i and thus collect the M_i-minimal elements in an infinite set

$$K := \{k_i | k_i \text{ the minimal element of } M_i\}.$$

Therefore, we can apply again (1), for $S := K$ and obtain that at least 2 elements in K have the property (1):

$$\exists_{k_i,k_j}.\, k_i < k_j \rightarrow f_1(k_i) \leq f_1(k_j) \wedge f_2(k_i) \leq f_2(k_j). \tag{7.4}$$

However, $S_j \subset S_i \setminus M_i$, so $k_j \notin M_i$ and by (7.4) k_j is above k_i (w.r.t. $\leq^2$). This comes into contradiction with the fact that M_i collects all elements above k_i.

"$\Leftarrow$" Let $\mathbf{f}_1, \mathbf{f}_2 : S \rightarrow \mathtt{N}$, with $S \subset \mathtt{N}$ unbounded. Consider the ideal

$$I_S = \langle m_k, k \in S \rangle, \text{ where } m_k = X_1^{f_1(k)}X_2^{f_2(k)}.$$

By (5) I_S is finitely generated, i.e. there exists a finite set

$$N_0 = \{i_1, ..., i_n\} \subset S, \text{ s.t. } B_{N_0} = \{m_i, i \in N\} \text{ is a basis for } I_S.$$

Therefore, $\forall_{m_k \in I_S} \exists_{m_i \in B_{N_0}}\ m_i | m_k$, or

$$\forall_{k_0 \in S} \exists_{i_0 \in N_0}\ m_i | m_k. \tag{7.5}$$

Let $S_1 := S \setminus N_0$ and $I_{S_1} = \langle m_k, k \in S_1 \rangle$. We apply (5) for I_{S_1} and obtain that there is some finite $N_1 \subset S_1$ s.t. $B_{N_1} = \{m_i, i \in N_1\}$ is a finite basis for I_{S_1}. Thus, $\forall_{k_1 \in S_1} \exists_{i_1 \in N_1} m_{i_1} | m_{k_1}$.

On the other hand, since $N_1 \subset S_1 \subset S$, we have from (7.5) that

$$\forall_{i_1 \in N_1} \exists_{i_0 \in N_0}\ m_i | m_{i_1}.$$

Thus,

$$\forall_{k_1 \in S_1} \exists_{i_1 \in N_1} \exists_{i \in N}.\ \ m_i | m_{i_1} \wedge m_{i_1} | m_{k_1}. \tag{7.6}$$

We can continue by taking $S_2 := S_1 \setminus N_1$ and $I_{S_2} = \langle m_k, k \in S_2 \rangle$. For this, by (5) there exists a finite set $N_2 \subset S_2$, s.t. $B_{N_2} = \{m_i, i \in N_2\}$ is a basis for I_2. By a similar reasoning as the one leading to (7.6), we obtain

$$\forall_{k_2 \in S_2} \exists_{i_2 \in N_2} \exists_{i_1 \in N_1, i_0 \in N_0}.\ \ m_{i_0} | m_{i_1} \wedge m_{i_1} | m_{i_2} \wedge m_{i_2} | m_{k_2}. \tag{7.7}$$

We can clearly continue this process, since whenever we take a subset of S_n, this generates an ideal, for which (5) is applicable. That is, suppose that for some $n \geq 1$, we have

$$\forall_{k_n \in S_n} \exists_{i_n \in N_n \ldots i_1 \in N_1, i_0 \in N_0}.\ \ \forall_{j<n} m_{i_j} | m_{i_{j+1}} \wedge m_{i_n} | m_{k_n}. \tag{7.8}$$

By taking $S_{n+1} = S_n \setminus N_n$ and $I_{S_{n+1}}$, we obtain a finite N_{n+1} s.t. B_{N_n} is a basis for $I_{S_{n+1}}$. Thus, we have:

$$\forall_{k_{n+1} \in S_{n+1}} \exists_{i_{n+1} \in N_{n+1} \subset S_n}.\ m_{i_{n+1}} | m_{k_{n+1}}.$$

By (7.8), this means that there is a sequence of $n+1$ monomials ordered by the "$|$"-relation. Thus, if we collect all these indexes in a set, this would be an infinite set M for which (1) holds. I.e.,

$$\exists_{M \text{infinite}} \forall_{k,l \in M}.\ l < k \rightarrow (f_1(l), f_2(l)) \leq^2 (f_1(k), f_2(k)). \tag{7.9}$$

□

The formulation required for the Gröbner Bases Theory. The refined A-Translation is not modular. More precisely, we cannot transform a non-constructive property and associate an extracted term to it, unless this can be formulated as a Π^0_2-formula. Thus, since neither (1) nor (5) are Π^0_2-formulas, we cannot use the A-Translation to directly interpret them. For this reason, we restrict our attention in Section 7.4 to a weakening of (1) to a Π^0_2-corollary. The corollary shows the existence of a *finite subset* of the unbounded set M for a finite number of functions and is however not sufficient in order to show the termination of Buchberger's Algorithm. For the algorithm, it is necessary that the finite basis covers an *infinite* space, the ideal, by the linear combinations formed with its elements. This property is, as demonstrated in this section, equivalent to (1). This problem does not have a computational solution, in the sense that it is impossible to verify that this infinite space is indeed covered by the given basis. On the other hand, (5) cannot be weakened to an equivalent form of the Π^0_2-corollary associated with (1). As we have seen in the proof of this equivalence, it is essential and necessary to know from (1) that at each step an *unbounded* set M_i can be constructed and this cannot be formulated as a Π^0_2 property.

7.4 A Π^0_2-Corollary of Dickson's Lemma

As aforementioned, (1) is not a Π^0_2 formula and thus A-Translation cannot be applied to it. Therefore, we weaken it to a constructive principle and present in this section a corollary in which we treat the case of two functions which are proven to simultaneously increase in two points. We analyze on the extracted program and compare it to a brute-force one. This investigation will provide us with useful information regarding the application of A-Translation, such as the role of negations and the fact that the extracted term adheres to the continuation passing style. Further, we sketch the generalization to a finite number of points, which follows as a consequence of the provable fact that any given functions increase monotonically on an unbounded subset of their infinite domain.

7.4.1 A Simplified Statement.

Formulation and proof.

Proposition 7.1. *Any two functions of natural numbers increase simultaneously for at least two distinct values. This can be stated as the Π^0_2-statement:*

$$\forall_{f,g}{}^{\tilde{\exists}}{}_{i,j}.\ i < j \mathbin{\tilde{\wedge}} f(i) \leq f(j) \mathbin{\tilde{\wedge}} g(i) \leq g(j). \qquad (DL_2)$$

We give an independent proof, following the lines of (Berger et al., 2002).

Notation 7.1. *$M(x)$ abbreviates $x \in M$.*

Proof. We fix arbitrary f and g and define as in (Berger et al., 2001) the set

$$M := \{\, x \mid \forall_{y \geq x} f(x) \leq f(y) \,\}.$$

Using (MinPr) with $m := f$ and $P(x) := T$, we can infer that $M \neq \emptyset$. Therefore, we can apply (MinPr) again with $m := g$ and $P(x) := M(x)$, to obtain the M-minimum w.r.t. the measure g. From

$$\tilde{\exists}_x.\ M(x) \mathbin{\tilde{\wedge}} \forall_y.\ g(y) < g(x) \to \neg M(y), \tag{7.10}$$

we have the desired i. In order to obtain j, it suffices to take the next value above i in M, since

- for every $j \geq i$ in M, $f(i) \leq f(j)$, by the definition of M
- i is minimal w.r.t. g, since by (7.10), $M(j) \to g(j) \not< g(i)$.

We obtain by the Minimum Principle on $P(x) := x > i$ and $m := f$ that

$$\tilde{\exists}_x.\ x > i \mathbin{\tilde{\wedge}} \forall_y.\ f(y) < f(x) \to y \not> i. \tag{7.11}$$

For x fulfilling (7.11), it follows easily that $x \in M$, since for any $y \geq x$, we can infer that $y > i$. By (7.11) this implies that $f(x) \leq f(y)$. Thus, such an x has exactly the properties required for j. □

Negations

The Minimum Principle (MinPr) provides an x with the properties $P(x)$ and $\forall_y.\ m(y) < m(x) \to\ \tilde{\neg} P(y)$ or the (classically) equivalent $\forall_y.\ P(y) \to m(y) < m(x) \to \bot$. In order to use this together with the negative assumptions from (DL_2) we need to eliminate the logical falsity $\bot$ by $\mathsf{efq}^\bot$ or $\mathsf{Stab}^\bot$. As discussed in Section 2.3.2 neither of them is allowed in NA^ω.

More precisely, we would need the lemma

$$\vdash (f(y) < f(x) \to \bot) \to f(x) \leq f(y),$$

and its proof requires $\mathsf{efq}^\bot$.

We can overcome this, by noticing that $f(x) \leq f(y)$ is equivalent to $f(y) < f(x) \to \mathrm{F}$. To match this negative version with (MinPr), we take the logical falsity and rewrite (DL_2) to its classically equivalent:

$$\forall_{f,g} \tilde{\exists}_{i,j}.\ i < j \mathbin{\tilde{\wedge}} (f(j) < f(i) \to \bot) \mathbin{\tilde{\wedge}} (g(j) < g(i) \to \bot). \tag{DLCor}$$

Formal proof.

For P a unary predicate, we use the abbreviations

$$R(m,x) := \forall_y.\, m(y) < m(x) \to \tilde{\neg} P(y)$$
$$Q(m,x) := R(m,x) \to \tilde{\neg} P(x) = \forall_y.\, m(y) < m(x) \to \tilde{\neg} P(y) \to \tilde{\neg} P(x)$$

and thus (MinPr) is $\forall_m.\ \tilde{\exists}_x P(x) \to \forall_x Q(m,x) \to \bot$ or its (classically) equivalent by contraposition $\forall_{m,x}.\, \forall_x Q(m,x) \to \tilde{\neg} P(x)$.

For the proof of Proposition 7.1, we have $m_1 := f, m_2 := g, m_3 := f$ and $P_1(n) := \mathrm{T}, P_3(n) := \forall_m.\, n < \mathsf{S}m \to fm < fn \to \bot, P_3(i,n) := i < n$. GInd is the general induction scheme introduced on page 129. The assumption variables have the types:

$$u_k^{\tilde{\exists}_n P_k(n)}, u_{3+k}^{\forall_n Q_k(n,m_k)}, k \in \{1,2,3\}$$
$$H_{1,\mathsf{MP}}^{R_1(n))}, H_1^{P_1(n)},\ H_{2,\mathsf{MP}}^{R_2(n)}, H_2^{P_2(i)},\ H_{3,\mathsf{MP}}^{R_3(i,n)}, H_3^{P_3(i,j)}$$

and we use the following abbreviations

$$T_1 := \tilde{\exists}_1^- \ (\lambda u_1, u_4.\ u_1\ \lambda n.\ \mathsf{GInd}_1\ f\ n\ u_4)\ (\lambda w^{\forall_n(T\to\bot)}.\ w\ 0\ \mathrm{T})$$
$$T_2 := \tilde{\exists}_2^- \ (\lambda u_2, u_5.\ (u_2\ \lambda n.\ \mathsf{GInd}_2\ f\ g\ n\ u_5)$$
$$(\tilde{\exists}_2^+\ n\ \lambda i, v_1^{n<i+1}, v_2^{fi<fn}.\ H_{1,\mathsf{MP}}\ i\ v_2\ \mathrm{T}))$$
$$T_3 := \tilde{\exists}_3^- \ (\lambda u_3, u_6.\ (u_3\ \lambda l.\ \mathsf{GInd}_3\ i\ f\ l\ u_6)\ \tilde{\exists}_3^+\ \mathsf{S}i\ \mathrm{T}).$$

Each of these terms corresponds to an application of the Minimum Principle, as described in the informal proof. The term $M_{(\mathrm{DLCor})}$ associated to the proof of the corollary of Dickson's Lemma is:

$$M_{(\mathrm{DLCor})} := \lambda f, g.\ T_1\ (\lambda n, H_{1,\mathsf{MP}}, H_1.\ T_2)\ (\lambda i, H_{2,\mathsf{MP}}, H_2.\ T_3)$$
$$\lambda j, H_{3,\mathsf{MP}}, H_3.\ (\tilde{\exists}^+\ i\ j\ H_3\ (H_2\ j\ H_3))$$
$$\lambda v_3^{gj<gi}.\ H_{2,\mathsf{MP}}\ j\ v_3$$
$$(\lambda m, v_4^{j<m+1}, v_5^{fm<fj}.\ H_{3,\mathsf{MP}}\ m\ v_5\ (Lem_{\mathsf{Trans}}\ H_3\ v_4)),$$

where we have used the lemmas for $\tilde{\exists}$-introduction and elimination and the transitivity lemma $Lem_{\circ\mathsf{Trans}}$ with $\circ \in \{<, \leq\}$.

Extracted Program

All three subformulas of the goal are atomic - positive or negated, so they are goal formulas, in the sense of Definition 3.5. The relevant conjuncts

$f(j) < f(i) \to \bot$ and $g(j) < g(i) \to \bot$ require the transformation by (3.4) from Lemma 3.6. Since they are in the form $D_0 \to G$, we will apply Lemma 3.4 (Case Distinction) to $f(j) < f(i)$ and $f(j) < f(i)$. This extracts to the conditionals on the values of f and g.

Let $F_i^{\mathtt{N}\Rightarrow\mathtt{N}\times\mathtt{N}}$, $i \in \{0,1\}$ and $H^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N}\times\mathtt{N})\Rightarrow\mathtt{N}\times\mathtt{N}}$. The program extracted from the proof of the corollary of Dickson's Lemma is

$$P_{\mathrm{DL}} := \lambda f, g.\ (\mathcal{F}_1)_{\mathtt{N}}^{\mathtt{N}\times\mathtt{N}}\, f\, 0\, G_1, \text{ with}$$

$$\begin{aligned}
G_1 &:= \lambda n.\ (\mathcal{F}_2)_{\mathtt{N}}^{(\mathtt{N}\Rightarrow\mathtt{N}\times\mathtt{N})\Rightarrow\mathtt{N}\times\mathtt{N}}\, g\, n\, G_2,\\
G_2 &:= \lambda n_0, H, F_1.\ (\mathcal{F}_3)_{\mathtt{N}}^{\mathtt{N}\times\mathtt{N}}\, f\, (\mathsf{S}n_0)\, G_3,\\
G_3 &:= \lambda n_1, F_0.\ [\mathsf{if}\ (fn_1 < fn_0)\ (F_1\, n_1)\ [\mathsf{if}\ (gn_1 < gn_0)\ (Hn_1F_0)\ (n_0, n_1)]],
\end{aligned}$$

with $\mathcal{F}^{\sigma}_{i,\mathtt{N}}$ the general recursion operators associated to the GInd.

With the reduction rules (7.1), we can unfold[2] this to

$$\begin{aligned}
&P_{\mathrm{DL}}(k, T_1, T_2, T_3) =\\
&\quad \lambda f, g.\ \mathsf{if}\ (f\mathsf{S}k < fk)\ \mathsf{then}\ T_1(k)\, \mathsf{S}k\\
&\qquad\qquad \mathsf{else\ if}\ (g\mathsf{S}k < gk)\ \mathsf{then}\ T_2(k)\, \mathsf{S}k\, T_3(\mathsf{S}k),\\
&\qquad\qquad\qquad \mathsf{else}\ (k, \mathsf{S}k)
\end{aligned}$$

$$\begin{aligned}
\text{with}\qquad T_1(k) &:= \lambda y.\mathsf{if}\ (f(y) < f(k))\ \mathsf{then}\ \mathcal{F}_1(f, y, G_1)\ \mathsf{else}\ \varepsilon && (7.12)\\
T_2(k) &:= \lambda y.\mathsf{if}\ (g(y) < g(k))\ \mathsf{then}\ \mathcal{F}_2(g, y, G_2)\ \mathsf{else}\ \varepsilon && (7.13)\\
T_3(k) &:= \lambda y.\mathsf{if}\ (f(y) < f(k))\ \mathsf{then}\ \mathcal{F}_3(f, y, G_3)\ \mathsf{else}\ \varepsilon. && (7.14)
\end{aligned}$$

T_3 gets evaluated if $\neg(f\mathsf{S}k < fk)$ and $g\mathsf{S}k < gk$, i.e., if k is suitable for f, but not for g. If this is the case, then

$$T_P := T_2(k)\, \mathsf{S}k\, T_3(\mathsf{S}k) = \mathcal{F}_2(g, \mathsf{S}k, G_2)\, T_3(\mathsf{S}k).$$

Unfolding again, we obtain (see Appendix A.2 for details):

$$\begin{aligned}
T_P := \mathsf{if}\ (f\, \mathsf{S}(\mathsf{S}k) < f\, \mathsf{S}k)\ \mathsf{then}\ T_3(\mathsf{S}k)\, \mathsf{S}(\mathsf{S}k)\\
\mathsf{else\ if}\ (g\mathsf{S}(\mathsf{S}k) < g\mathsf{S}k)\ \mathsf{then}\ T_2(\mathsf{S}k)\ \mathsf{S}(\mathsf{S}k)\, T_3(\mathsf{SS}k)\\
\mathsf{else}\ (\mathsf{S}k, \mathsf{S}(\mathsf{S}k)),
\end{aligned}$$

so $T_P = P_{\mathrm{DL}}(\mathsf{S}k, T_3, T_2, T_3)\, f\, g$.

[2]For a detailed presentation of this unfolding please refer to Appendix A.2.

Summarizing the above unfoldings, the program is

$$\begin{aligned} P_{DL}(k, T_1, T_2, T_3) := \lambda f, g.\ &\mathsf{if}\ (f\mathsf{S}k < fk) \\ &\quad \mathsf{then\ if}\ (f\mathsf{S}k < fk) \\ &\qquad \mathsf{then}\ \mathcal{F}_1(f, \mathsf{S}k, G_1) \\ &\qquad \mathsf{else}\ \varepsilon \\ &\quad \mathsf{else\ if}\ (g\mathsf{S}k < gk) \\ &\qquad \mathsf{then}\ P_{DL}(\mathsf{S}k, T_3, T_2, T_3)\, f\, g \\ &\qquad \mathsf{else}\ (k, \mathsf{S}k), \end{aligned}$$

where the recursive call to P_{DL} corresponds $T_P := \mathcal{F}_2(g, \mathsf{S}k, G_2)\, T_3(\mathsf{S}k)$.

Interpretation. The recursion operators illustrate the functionality of the Minimum Principle: $\mathcal{F}_1$ investigates whether the current value k is the "minimal" value of f and in case of a positive answer, it moves onward to the second recursion. The second operator tests whether the value is suitable also for g and if this is the case (i has been found), then it searches for the second value (j) for f. Notice that in order to find j, the program investigates the successor of i.

In the above unfoldings (see Appendix A.2) we have used the substitutions

$$\begin{aligned} &F_0 := T_3(\mathsf{S}k),\ F_1 := T_2(k), H := T_1(k) && \text{for } P_{DL} \\ &F_0 := T_3(\mathsf{SS}k),\ F_1 := T_2(\mathsf{S}k), H := T_3(\mathsf{S}k) && \text{for } T_P\ , \end{aligned}$$

where the terms T_i's are given by (7.12) - (7.14).

Hence, via the parameters F_0, F_1 and H, the term G_3 directs the computation to look for further values or to return the answer $(k, \mathsf{S}k)$, which represents the instance for (n_0, n_1) satisfying the required properties on f and g. However, the real computation is "on hold" until the pair (n_0, n_1) is available and happens only when G_3 is computed. In case suitable values for $(k, \mathsf{S}k)$ are found, this pair is returned.

Continuation passing style. What did we mean in the above by "the computation is *on hold*"? In what way are we able to associate a program to a proof which uses the property that minimal elements exist in infinite sequences? The Minimum Principle is non-constructive in essence, so how come that we are able to involve it in a computable functional returning the values at which the functions increase?

First, let us make the observation that the computation is clearly possible only when we search for a *finite* number of points with the desired property. Thus, in order to computing the two values, the first question to ask is: "Are the i and j indeed the minimal elements of the two functions?". The answer is "Not necessarily!". In fact, it is impossible to guarantee this property and the important fact is that is not required in order to answer the question "What are the two values on which both f and g are increasing?".

The essential aspect which enables to output the correct result is the interpretation given to the logical falsity. $\bot$ plays the role of a hole, which puts the current computation on hold until the result is known. Thus, the program searches for a "small enough" i, rather than a minimal one. As soon as a small enough i is detected, in the sense that there exists a j above it with the property that the first function increases in these two indices, the hole for the first recursion is filled and the program exits the current computation. Further, a check is performed, whether this pair satisfies the property for the second function. Should an appropriate pair (i, j) be found, the last recursion stops as well, fills in the corresponding hole and the program exits. Should the pair satisfying f be unsuited for g the program returns computing the pair for f, by *continuing* with the i above the last investigated j.

The actual computation is performed by the term G_3, which tests whether the current values fulfill the desired properties and directs the search further in case they fail for one or the other function. Thus, the extracted program does not guarantee that the (i, j) that are picked are minimal with respect to all the values that function takes (so infinitely many), but rather that each such pair is "good enough".

Experiments

We depict in Figure 7.2 some runs on randomly generated sequences. We represent the values of f and g in the lists l_f and l_g. P_{DL} denotes the program extracted by the A-Translation and P_{BF} be the brute-force program.

For the brute-force program, the following strategy was considered: f and g are investigated in parallel at points (i, j) with $j \leq |l_f| = |l_g|$ and $i \leq j$. The first suitable pair - i.e., for which $f(i) \leq f(j)$ and $g(i) \leq g(j)$ is picked. A concrete run for this brute-force algorithm is presented in Figure 7.3(b).

Let us analyze a run in more detail on some input data.

Example 8. *Assume that we start with the sequences:*

$l_f = (80, 20, 53, 5, 48, 43, 75, 37, 18, 65), l_g = (73, 48, 11, 46, 34, 85, 56, 41, 28, 48).$

l_f	l_g	P_{BF}	P_{DL}
(76,59,48,62,92,93,29,40,17)	(93,87,76,99,97,47,7,61,12)	(1 . 3)	(2 . 3)
(78,7,55,37,89,66,21,47,27)	(51,71,32,51,89,67,7,2,66)	(0 . 4)	(3 . 4)
(41,33,53,56,44,25,48,12,73)	(60,59,33,77,49,45,60,4,2)	(0 . 3)	(2 . 3)
(36,43,64,17,17,31,4,86,12)	(93,9,27,14,0,15,13,88,25)	(1 . 2)	(1 . 2)
(14,98,51,45,10,98,94,45,7)	(90,9,45,39,15,45,19,82,48)	(1 . 5)	(4 . 5)

Figure 7.2: Average runs of the programs obtained with A-Translation vs. the brute-force method

Figure 7.3 presents the step-wise computation for this example, opposing the runs for A-Translation (subfigure 7.3(a)) to the runs for the brute-force method (subfigure 7.3(b)). The first column represents the indices in the current run, while the second and third column output the values at these indices for f and g, respectively. The pair fulfilling the requirements is marked in bold.

(i,j)	(fi,fj)	(gi,gj)
(0, 1)	(80, 20)	
(1, 2)	(20, 53)	(48, 11)
(2, 3)	(53, 5)	
(1, 3)	(20, 5)	
(3, 4)	(5, 48)	(46, 34)
(4, 5)	(48, 43)	
(3, 5)	(5, 43)	(46, 85)

(a) The program obtained after A-translation

(i,j)	(fi,fj)	(gi,gj)
(0, 1)	(80, 20)	(73, 48)
(0, 2)	(80, 53)	(73, 11)
(1, 2)	(20, 53)	(48, 11)
(0, 3)	(80, 5)	(73, 46)
(1, 3)	(20, 5)	(48, 46)
(2, 3)	(53, 5)	(11, 46)
(0, 4)	(80, 48)	(73, 34)
(1, 4)	(20, 48)	(48, 34)
(2, 4)	(53, 48)	(11, 34)
(3, 4)	(5, 48)	(46, 34)
(0, 5)	(80, 43)	(73, 85)
(1, 5)	(20, 43)	(48, 85)

(b) The brute-force method

Figure 7.3: The runs for Example 8

As we can see, although the brute force algorithm finds the first suitable pair, i.e. the smallest values for which the property is satisfied, it also needs more runs. We expect to have better computational complexity for P_{DL} since computing with continuations optimizes the search, as follows:

- the pair (i, j) is investigated whether it is suitable for f. This results

as a consequence of taking $H := T_1(k)$ in P_{DL};

- should (i, j) be unsatisfying for f, the search is resumed for f, without investigating the values at g. The next pair of indexes (i', j') considered are the last i at which g failed and the last j at which f failed;
- should (i, j) be appropriate for f, the values are tested for g;
- in case they fail for g, the search continues with $i' := j$ and $j' := i + 1$. This follows from resuming the computation with $H := T_3(k)$ in T_P. Otherwise, the hole for $\perp$ is filled and the result is returned.

Let us trace another example to illustrate our argument.

Example 9. *We take* $l_f = (41, 52, 18, 64, 55, 28)$ *and* $l_g = (71, 41, 77, 54, 19, 50, 82)$ *- the run is presented in Figure 7.4.*

(i,j)	(fi,fj)	(gi,gj)
(0, 1)	(41,52)	(71,41)
(1, 2)	(52,18)	
(0, 2)	(41,18)	
(2, 3)	(18,64)	(77,54)
(3, 4)	(64,55)	
(2, 4)	(18,55)	(77,19)
(4, 5)	(55,16)	
(2, 5)	(18,16)	
(5, 6)	(16,28)	(50,82)

Figure 7.4: An example for the run of P_{DL}.

Conclusion. At the cost of not computing the least values at which the functions are found to increase, the A-translated program performs less steps and the search is faster. This is a consequence of computing with continuations, which characterizes the programs extracted from A-translated classical proofs.

7.4.2 Generalization

Proposition 7.1 can be viewed also as a corollary of the stronger version (7.3), which follows as a consequence of the following property:

Lemma 7.3. *Given an unbounded set $S \subset \mathtt{N}$ and $f : S \to \mathtt{N}$, the set*

$$S_f := \{\, m \mid S(m) \land \forall_n.\, S(n) \to m < n \to f(m) \le f(n) \,\}$$

of the left f-minima w.r.t. S is unbounded as well.

Proof idea. We fix an arbitrary m with $S_f(m)$. Using the fact that S is unbounded, we can find an n s.t. $m < n$ and $S_f(n)$. For this we apply (MinPr) with measure f to $\{\, n \mid S(n) \land m < n \,\}$. The detailed proof of Lemma 7.3 can be found in (Berger et al., 2002). □

With this property, we can show (7.3):

Lemma 7.4. *Let $S \subset \mathtt{N}$ be unbounded, $n \in \mathtt{N}$ and $f_1, ..., f_n : S \to \mathtt{N}$. Then there exists a finite $M \subset S$, such that*

$$\forall_{k<n} \forall_{i,j \in M}.\, i < j \to f_k(i) \le f_k(j),$$

Proof. By induction on n. For details, please refer to (Berger et al., 2002). □

Implementation idea. We represent the functions $f_1, ..., f_n : S \to \mathtt{N}$ by a generic function f with signature $\mathtt{N} \Rightarrow \alpha \Rightarrow \mathtt{N}$, where the first parameter maps the indices.

For each of the functions, we define the corresponding set S_{k+1} to be the left f_{k+1}-minima w.r.t. to S_k, i.e.,

$$\begin{aligned} S_1 &:= \{\, m \mid \forall_n.\, m < n \to\ \tilde{\neg} f_1 n < f_1 m \,\} \\ S_{k+1} &:= \{\, m \mid S_k(m) \land \forall_n.\, S_k(n) \land m \le n \to\ \tilde{\neg} f_{k+1} n < f_{k+1} m \,\}, k \ge 1. \end{aligned}$$

Thus, the set S_{k+1} is built from S_k and consists only of those points in S_k where in addition f_{k+1} increases.

We start from the premises:

- S is unbounded[3],
$$\forall_{f,m} \tilde{\exists}_n.\, m < n \mathbin{\tilde{\land}} S(n) \qquad (S_\infty)$$
- $S_{\mathsf{S}k}$ can be constructed from S_k by

$$\forall_{f,m,k}.\, (S_k(m) \land \forall_n.\, S_k(n) \to m < n \to\ \tilde{\neg} f_{\mathsf{S}k} n < f_{\mathsf{S}k} m) \to S_{\mathsf{S}k}(m) \quad (\mathsf{S}_\to)$$

[3]For simplification, we can consider $S := \mathtt{N}$.

Since both clauses have computational content, due do the presence of $\bot$, they will occur as parameters in the extracted program.

Proposition 7.1 is now an easy consequence of Lemma 7.4. Moreover, it is possible to infer that:

$$\forall_{k,f}\tilde{\exists}_{i,j}.\ (i<j)\ \tilde{\wedge}\ \forall_{m\leq k}.\ f_{\mathsf{S}m}j < f_{\mathsf{S}m}i \to \bot. \qquad (\mathrm{DL}_k)$$

In what follows, we present the program associated with the proof of this generalized version (DL_k).

Extracted program. Let $\tau := \mathtt{N}\times\mathtt{N}\times\alpha\times\alpha$. We take a^α, b^α (with α some arbitrary type) and $F^{\mathtt{N}\Rightarrow\alpha\Rightarrow\tau}$, $H^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\alpha\Rightarrow\tau)\Rightarrow\tau}$. Further, we have the typing $\mathcal{R}_{\mathtt{N}}^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\alpha\Rightarrow\tau)\Rightarrow\tau}$ for all the recursion operators involved and $\mathcal{F}_{\mathtt{N}}^{\alpha\Rightarrow\tau}$ for all the general recursion operators. The auxiliary assumptions have computational content, so are typed as $cS_{\to}^{\alpha,\tau}$ and $cS_{\infty}^{\alpha,\tau}$. We abbreviate the terms to:

$$\begin{aligned}
f_1 &:= \lambda n, F_1.\ cS_\infty\, f\, n(\lambda m.\ \mathcal{F}_1\ f1\ m\ G_1\ \mathrm{T})\\
G_1 &:= \lambda n', F_2, a_1.\ F_1\, n'\, (cS_\to\, f\, n'\, 0\, (a_1, F_2))\\
g_1 &:= \lambda n, H, m, F_3.\ H\ m\ (\lambda n'.\ \mathcal{F}_2\ (f\ \mathsf{S}\mathsf{S}n)\ n'\ G_2\ \mathrm{T})\\
G_2 &:= \lambda n'', F_4, a_2.\ F_3\, n''\, (cS_\to\, f\, n''\, (\mathsf{S}n)\ (a_2, F_4))\\
f_2 &:= \lambda m, F_5.\ cS_\infty\, f\, m\, (\lambda n'.\ \mathcal{F}_3\ f1\ n'\, G_3\mathrm{T})\\
G_3 &:= \lambda n'', F_6, a_2.\ F_5\, n''\, (cS_\to\, f\, n''\, 0\, (a_2, F_6))\\
g_2 &:= \lambda m, H', n', F_7.\ H'\, n'\, (\lambda n''.\ \mathcal{F}_4\, (f\ \mathsf{S}\mathsf{S}m)\, n''\, G_4\mathrm{T})\\
G_4 &:= \lambda n''', F_8, a_3.\ F_7\, n'''\, (cS_\to\, f\, n'''\, (\mathsf{S}m)\, (a_3, F_8)).
\end{aligned}$$

Due to the similarity, we can write, with $i = 1, 2$:

$$\begin{aligned}
f_i &:= \lambda n, F_i.\ cS_\infty\, f\, n(\lambda j.\ \mathcal{F}_1\ f1\ j\ G_1\ \mathrm{T})\\
G_i &:= \lambda k, F_{i+1}, a.\ F_i\, k\, (cS_\to\, f\, k\, 0\, (a, F_{i+1}))\\
g_i &:= \lambda n, H, m, F_{i+2}.\ H\ m\ (\lambda j.\ \mathcal{F}_{i+1}\ (f\ \mathsf{S}\mathsf{S}n)))\ j\ G_{i+2}\ \mathrm{T})\\
G_{i+2} &:= \lambda k, F_{i+3}, b.\ F_{i+2}\, k\, (cS_\to\, f\, k\, (\mathsf{S}n)\ (b, F_{i+3}))
\end{aligned}$$

With this, the program associated to the derivation of the corollary (DL_k) from Lemma 7.4 is:

$$P_{\mathrm{DL}_k} := \lambda f, k.\ (\mathcal{R}_1\ k\ f_1\ g_1)\ 0\ (\lambda n, a.\ (\mathcal{R}_2\ k\ f_2\ g_2)\ n\ (\lambda m, b.\ (n, m, a, b)))$$

As can be seen from the step terms g_1 and g_2 for $\mathcal{R}_1$ and respectively $\mathcal{R}_2$, these are in tail recursive form, since both H and H' are applied in head position in the step functions. However, no similar concept has been developed for the general recursion operator and it would be interesting to investigate whether the same observation can be made for $G_i, i \in \{1, ...4\}$.

7.5 Applications of Dickson's Lemma

In this section, we overview some of the applications of the strong version of Dickson's Lemma in the Gröbner Basis Theory, as presented in (Cox et al., 1992). Although these are not formulated as Π^0_2-statements, we envisage for further investigation to weaken them such that that they fit the requirements of A-Translation. We are interested in using Dickson's Lemma as a parameter in these proofs and thus abstracting from the fact that its strong version is not a Π^0_2-statement.

7.5.1 Further Terminology

We give in the following a few concepts necessary to illustrate some of the applications of Dickson's Lemma.

Definition 7.8 (Reducibility modulo)**.** *Let f, p be given. If there exists $m \in M(f)$ s.t. $lm(p)|m$, let $t := \text{coeff}(m, f) \cdot m$ (i.e., t is the corresponding term to m in f). We define:*

$$\lambda_{p,m}(f) := \frac{\text{coeff}(m, f) \cdot m}{lt(p)} = \frac{t}{lt(p)}.$$

We say that:

(i) *f reduces to g modulo p by eliminating t if there is m as above and $g = f - \lambda_{p,m}(f) \cdot p$. Notation: $f \underset{p}{\rightarrow} g[m]$.*

(ii) *f* reduces to *g* modulo *p if for some m, $f \underset{p}{\rightarrow} g[m]$. We denote this by $f \underset{p}{\rightarrow} g$.*

(iii) *f* reduces to *g* modulo *P if for some $p \in P$, $f \underset{p}{\rightarrow} g$. This will be denoted by $f \underset{P}{\rightarrow} g$.*

(iv) *f* is reducible modulo *p (P) if for there exists $g \in K[\underline{X}]$ s.t. $f \underset{p}{\rightarrow} g$ (respectively, $f \underset{P}{\rightarrow} g$).*

Theorem 7.1 (Division Theorem for multivariate polynomials)**.** *Let polynomials $g_1, \ldots, g_l \in K[\underline{X}]$ be fixed. Every $f \in K[\underline{X}]$ can be expressed as:*

$$f = p_1 g_1 + \ldots + p_l g_l + r \tag{7.15}$$

where $p_i, r \in K[\underline{X}]$ and either $r = 0$ or else it is the case that none of the terms in r is divisible by any of $lt(g_1), \ldots, lt(g_l)$.

Proof. The theorem can be interpreted as an existence theorem, in which, given $g_1, \ldots, g_l$, we need to show that for any $f \in K[\underline{X}]$, there exist polynomials $p_1, \ldots, p_l \in K[\underline{X}]$ with the property (7.15).

Instead of proving the theorem by giving the division algorithm and showing that it is correct and it terminates, we incorporate the algorithmic content in our proof. Namely, we show that $p_1, \ldots, p_l$ exists by constructing them. We consider the term ordering $\geq$ to be a priory fixed and $g_1, ..., g_l$ to be ordered by $lt(g_1) \geq \ldots \geq lt(g_l)$. With the notation from Definition 7.8, $\lambda_{g,lt(f)}(f) := \frac{lm(f)}{lm(g)}$ and we denote by $R_{f,g} = f - \lambda_{g,lt(f)}(f) \cdot g$. Notice that $lm(f) \notin R_{f,g}$.

We proceed by induction on the $lm(f)$, the monomial with the highest sequence of exponents, w.r.t. the term ordering which we fixed.

(IH): For any g with $lt(g) \leq lt(f)$, where f is the polynomial given in the theorem, we have a pair p_i, r, $i \in \{1, ..., l\}$ such that:

$$g = p_1 g_1 + \ldots + p_l g_l + r,$$

with r fulfilling the condition in Theorem 7.1.

Suppose now for f that there is a g_j for some $j \in \{1, \ldots, s\}$ s.t. $lt(g_i) \mid lt(f)$.

If this is not the case, let $r = lt(f)$ and $f' = f - r$. Clearly, we have $lt(f') \leq lt(f)$, so we can apply the induction hypothesis on f' and obtain $p'_1, \ldots, p'_l$ such that:

$$f' = p'_1 g_1 + \ldots + p'_l g_l + r',$$

with r' satisfying the condition in Theorem 7.1. Thus,

$$\begin{aligned} f &= f' + lt(f) \\ &= p'_1 g_1 + \ldots + p'_l g_l + r' + r \end{aligned}$$

where the rest is $r + r'$. Since none of the terms in r or r' is divisible by any of $lt(g_1), \ldots, lt(g_l)$, the same holds for $r + r'$, so this requirement of the theorem is fulfilled.

Otherwise, if there is a g_j as above, we compute $\lambda_{g_1,lt(f)}(f)$ and let $f' = R_{f,g_j}$. By the remark on R_{f,g_j}, $lt(f) \notin f'$ and it is also the case that $lt(f') \leq lt(f)$. Thus, we can apply (IH) for f' and obtain $p'_1, \ldots, p'_l$ such that:

$$f' = p'_1 g_1 + \ldots + p'_l g_l + r',$$

where $r' = 0$ or else none of the terms in r' is divisible by any of $lt(g_1), \ldots, lt(g_l)$.

With this,

$$\begin{aligned} f &= f' + \lambda_{g_j, lt(f)}(f) \cdot g_j \\ &= p'_1 g_1 + \ldots + p'_l g_l + r' + \frac{lm(f)}{lm(g_j)} \cdot g_j \\ &= p'_1 g_1 + \ldots + (p'_j + \frac{lm(f)}{lm(g_j)}) \cdot g_j + \ldots + p'_l g_l + r' \end{aligned}$$

□

Remark. Whereas for univariate polynomials, the quotient and rest are uniquely determined by the division, in the multivariate polynomial field this is not the case anymore. We give a counterexample below.

Example 10. *Let* $g_1, g_2, f \in \mathbb{R}[X_1, X_2]$.

$$\begin{aligned} g_1 &= {X_1}^2 X_2 + X_1 \\ g_2 &= X_1 X_2 - X_2^2 \\ f &= X_1^3 X_2 + X_1 X_2. \end{aligned}$$

If we divide f first by g_1*, and then by* g_2*, we get* $f = {X_1}^2 g_1 + X_2 g_2$.
If we choose first g_2 *and then* g_1*, then we obtain* $f = X_1 g_2 - g_1 - {X_1}^2 + {X_2}^2$.

From the first decomposition in this example, we conclude that f is in the ideal $< g_1, g_2 >$, whereas this is not obvious in the second division. Thus, it can happen that a polynomial is in the ideal generated by some polynomials, and yet, it would not give the rest 0 by division to those polynomials.

7.5.2 Dickson's Lemma in the Gröbner Bases Theory

The theory which has evolved around ideals is confronted with decision problems such as Ideal Description, Ideal Membership Problem or Nullstellensatz. These problems gain in complexity when one regards ideals over multivariate fields. The Gröbner Basis Theory offers positive answers to these questions.

In particular, Dickson's Lemma can be used to answer the following:

Ideal Description Problem. Let I be an ideal in $K[\underline{X}]$. Is it possible to find $f_1, ..., f_k \in K[\underline{X}]$, such that $I =< f_1, ..., f_k >$?

Clearly, the question can be answered positively. Furthermore, it can be used to decide the ideal membership:

Ideal Membership Problem. Given $f_1, ..., f_k \in K[\underline{X}]$, decide whether some $f \in K[\underline{X}]$ is in the ideal $< f_1, ..., f_k >$.

Whereas in the one variable case deciding the ideal membership problem comes down to computing the GCD of the given polynomials and determining whether f divided by the GCD gives the rest 0, in the multivariate case the problem is not as simple. This is because in the division algorithm for multivariate polynomials (see Theorem 7.1), the rest is *in general* not uniquely determined.

However, by choosing the appropriate basis for the ideal $< f_1, ..., f_k >$, one can achieve the uniqueness property. Such bases are the Gröbner Bases, for the computation of which (Buchberger, 1965) proposed his well-known algorithm. The termination of the algorithm is guaranteed by the ascending chain condition, which at its turn is proven using Hilbert's Basis Theorem and this follows from Dickson's Lemma, as shown in what follows.

Theorem 7.2 (Hilbert Basis Theorem)**.** *Let I be an ideal of $K[\underline{X}] \neq \{0\}$. Then there exist finitely many polynomials $g_1, \ldots, g_t \in I$ generating I, i.e. I has a finite basis.*

Proof. If $I = \{0\}$, then it is generated by $\{0\}$.

Otherwise, by Dickson's Lemma (4), the monomial ideal generated by $lt(I)$ has a finite basis, i.e.

$$\langle lt(I)\rangle \;=\; \langle lm(g_1), \ldots, lm(g_n)\rangle \;=\; \langle lt(g_1), \ldots, lt(g_n)\rangle,$$

for $g_1, \ldots, g_n \in I$ (finitely many).

Let now $f \in I$ some polynomial. By the Division Theorem 7.1, if f is divided by $g_1, \ldots, g_n$, then

$$f \;=\; p_1 g_1 + \ldots + p_n g_n + r,$$

where $p_1, \ldots, p_n, r \in K[\underline{X}]$ and either $r = 0$, or else no monomial in r is divisible by any of $lt(g_1), \ldots, lt(g_n)$. If the later was the case, since $r = f - p_1 g_1 - \ldots - p_n g_n \in I$, then $lt(r) \in \langle lt(I)\rangle$. However, from $\langle lt(I)\rangle = \langle lt(g_1), \ldots, lt(g_n)\rangle$ it would follow that at least some $lt(g_i)$ from the basis divides $lt(r)$, which is a contradiction.

Thus, $r = 0$ and $f = p_1 g_1 + \ldots + p_n g_n$, hence $f \in \langle lt(g_1), \ldots, lt(g_n)\rangle$. □

In order to apply the refined A-translation to the above applications of Dickson's Lemma it is necessary to find meaningful Π^0_2-corollaries of these theorems. We leave this for further investigation. It is possible to A-translate these applications, even when they use DL, unless we envisage a modular translation and program extraction.

7.6 Summary of the Chapter

Dickson's Lemma has raised our interest due to its application in the Gröbner Bases Theory and we envisaged the ambitious goal of synthesizing Buchberger's Algorithm. During our investigations it turned out that the classical proof of Dickson's Lemma cannot be A-translated in its general form, since it is not a Π^0_2-formula. Neither the simplified corollary which we have presented, nor its generalizations, suffice in order to guarantee the termination of Buchberger's Algorithm. In order to justify this observation we have compared in Section 7.3 various formulations of Dickson's Lemma and have pointed out which one is necessary in order to prove that the algorithm computing the Gröbner Basis of a given ideal terminates.

The limitation imposed by refined A-Translation resides on the one hand in restricting the working system to NA^ω and on the other in its lack of modularity. For this reason, it would be interesting to analyze the strong version of Dickson's Lemma by a different method, such as bar recursion or Dialectica and use its extracted term as a parameter in the theorem specifying the termination of Buchberger's Algorithm. We do not yet know of related work interpreting the classical proof Dickson's Lemma with appropriate methods.

The program extracted from the Π^0_2-corollary of Dickson's Lemma is based on existing work, but is presented from a new perspective. We have added the treatment of the Minimum Principle with its associated realizer: the general recursion operator. Further, we have illustrated the observation from Chapter 4 that the programs obtained from refined A-translated proofs adhere to the continuation passing style. We have also emphasized the role of negations with respect to the restrictions imposed by working in minimal logic and by requiring the formulas to be in the definite/goal classes.

Future work

As observed in Chapter 4, the induction proofs transformed by A-Translation result in general in tail recursive procedures. It would be interesting to extend the notion of tail recursion in the case of the general recursion operator and investigate whether the course-of-values induction used in the proof of Dickson's Lemma in the form of Minimum Principle has the same property.

Another interesting line for further investigation is the analysis of the programs associated with the equivalent formulations presented in Section 7.3 or their weakening to Π^0_2-formulas, where such meaningful corollaries exist.

We have seen in Section 7.5 that, according to the equivalences in Section 7.3, a Π^0_2-form of Dickson's Lemma is too weak to be used in the context

of the Gröbner Bases Theory, where the generators need to cover an infinite space - the ideal. On the other hand, A-Translation can only be applied to Π^0_2-formulas, so we cannot provide in this way an independent realizer for Dickson's Lemma, so that we further use this for its applications. However, we have already encountered situations - for instance in the case of IPH in Chapter 5 - where the intermediate lemmas/axioms could be used, without requiring them to be Π^0_2-formulas. If they are replaced by their proofs before applying the refined A-Translation, then whether or not they are Π^0_2-formulas is not relevant anymore. Thus, one can further investigate whether some of the applications of the strong form of Dickson's Lemma are Π^0_2-formulas or have interesting Π^0_2-corollaries.

On the other hand, as suggested by (Seisenberger, 2003), one can couple A-translation with modified bar recursion (see (Berger and Oliva, 2006) for more on this), in order to provide realizers for the assumptions which are not definite formulas. This idea can be used to find the realizer for Dickson's Lemma independently and then couple this with the extraction from Π^0_2-statements in which the lemma is used. This way, even though the refined A-Translation cannot be used directly on Dickson's Lemma, its computational content could be exploited by finding applications which are compatible with the refined A-Translation.

Related Classical and Constructive Principles. We have presented in Section 7.3 equivalent formulations of Dickson's Lemma. This has enabled us to determine the strength of the formulation (7.3) and, more importantly, of the Π^0_2-corollary that we have analyzed.

Following some interesting discussions with Paulo Oliva and Josef Berger, there seem to be tight connections with well-known classical and constructive principles. We would like to investigate this and determine the relationship between DL and principles such as Principle of Omniscience (LPO), Axiom of (Dependent) Choice (AC, respectively DC), Double negation shift (DNS) or Weak König's Lemma (WKL). It would be interesting to bridge also Stolzenberg's Principle from Chapter 5 with Dickson's Lemma or its Π^0_2-corollary, following the results from (Berardi, 2006) and compare the computational content extracted by A-Translation from suited forms of these principles. In particular, we are interested whether the following are true:

- $IHP \Leftrightarrow \sigma^0_1 - LLPO$.
- $DL \Rightarrow \sigma^0_1 - LPO$.
- $LPO + AC \Rightarrow DL$.

Chapter 8

Refined Double-Negation

The refinement of the Friedman-Dragalin Translation presented in Chapter 3 can be applied, as we have seen in the previous chapters, only to the restricted classes of formulas given by Definition 3.5, called definite and goal. These restrictions are imposed by Lemma 3.6, such that for these formulas it is possible to replace some of the logical negations ($\bot$) by the arithmetical falsity (F), while still keeping $\bot$ in relevant positions. The purpose of $\bot$, seen as a placeholder, is to enable the recovery of the constructive content hidden in the classical proofs, by allowing computationally relevant formulas to be substituted for it. A complete characterization of the definite/goal formulas is still an open problem, but in this chapter we aim at shedding some light in this respect, by showing how to insert negations, in order to turn other formulas into definite or goal. By this we extend the domain of applicability of the A-Translation.

An important contribution of this chapter is to further exploit the idea of minimal insertion of negations, in order to find the NA^ω-provable formulas corresponding to intuitionistically valid ones. The NA^ω-validity is a necessary condition in order to apply the refined A-Translation, but this introduces a further restriction, by prohibiting the use of $\mathsf{efq}^\bot$. However, as we have seen in Chapter 2 (Section 2.3.2), efq is NA-provable, so a careful distinction needs to be made for the two forms of negations.

We begin by overviewing in Section 8.1 the distinction between arithmetical and logical falsity and describe the influence exerted by this distinction on the axioms efq and Stab. This also serves as a justification as to why the working system needs to be minimal logic. Working in NA^ω makes the double negation translation superfluous, at the price of restricting the class of formulas to which the refined A-Translation can be applied. We therefore

try to loosen these conditions, by introducing in Section 8.3 a refinement of the Gödel-Gentzen translation. We also give sufficient conditions such that some formulas are turned into NA^{ω}-provable ones by this (minimal) insertion of double negations.

The theory developed in this chapter evolves around the notion of L-critical predicate symbols, as introduced in Section 8.2. Based on this notion, we propose to double negate only atoms containing predicate symbols. We show in Section 8.3 that this is a correct translation, preserving provability.

We illustrate our claims by the examples treated in the Chapters 5 -7. These have constituted the starting point in formulating the refinement of the double negation translation, by which we insert negations in a controlled and minimal manner. In Chapters 5 -7 the insertion of negations has been done manually and not systematically, but based on the expertise gained from these case studies from the previous chapters we give in Section 8.3 a general method for the detection of the formulas that need double negation. In Section 8.5 we propose a further refinement of the double negation translation, by which we minimize even more the number of logical falsifies inserted in order to achieve NA^{ω}-provability. For this, we consider a mix of $\bot$ and F, whenever we use relation symbols that have a complement.

In what follows, we use the notion of definite/goal formulas as introduced by Definition 3.5.

8.1 Negations

As we have already pointed out when describing the system NA^{ω} in Section 2.2, we distinguish between two forms of negations:

- F, the arithmetical falsity, defined in terms of the inductive predicate for the Leibnitz equality as $F := Eq(\mathrm{ff}, \mathrm{tt})$
- $\bot$, taken to be a predicate variable for the purpose of A-Translation, such that it plays the role of a placeholder for arbitrary formulas, envisaging the substitution by the strong existential goal.

This distinction triggers a corresponding differentiation in the treatment of formulas and axioms involving negations, as described further.

efq and Stab

In this section we overview the axioms efq, Stab and their logical variant $\mathsf{efq}^{\bot}$ and $\mathsf{Stab}^{\bot}$.

Let B be some arbitrary formula. The proofs carried out in this work are in minimal logic, so in our system NA^ω the logical axioms ${\mathsf{efq}^\bot}_B$ and ${\mathsf{Stab}^\bot}_B$ are not permitted. However, their arithmetical counterparts efq_B and Stab_B are provable (see Lemma 2.1) and are therefore allowed in NA^ω. In the following, we give an intuition in order to understand why it is necessary to exclude the logical forms, by analyzing the way in which A-Translation affects these axioms.

Let A be any formula. We recall the notation used in Chapter 3:

$$B^A := B[\bot := A].$$

efq_B Under A-Translation, $\mathsf{efq}_B^A = (\mathrm{F} \to B)^A = F \to B^A = \mathsf{efq}_{B^A}$. Since this is (provably) valid by Lemma 2.1, we allow efq_B in NA^ω.

Stab_B Stab_B^A is permitted in NA^ω, because

$$\mathsf{Stab}_B^A = [((B \to \mathrm{F}) \to \mathrm{F}) \to B]^A = ((B^A \to \mathrm{F}) \to \mathrm{F}) \to B^A = \mathsf{Stab}_{B^A}.$$

Hence, Stab_B^A also remains provable in minimal logic after A-translation.

${\mathsf{efq}^\bot}_B$ The A-translation of the logical variant of ex-falso axiom is

$$\mathsf{efq}_B^A := (\bot \to B)^A = A \to B^A,$$

where A can be any formula. In particular, if $A := \mathrm{T}$, then any formula B^T would be provable in the system(!).

Thus, we cannot accept ${\mathsf{efq}^\bot}_B$ in NA^ω.

${\mathsf{Stab}^\bot}_B$ $((B \to \bot) \to \bot) \to B$ is *not* accepted in our system either, since

$$\mathsf{Stab}_B^\mathrm{T} = (((B^T \to \mathrm{T}) \to T) \to B^T) \leftrightarrow B^T,$$

which makes any formula B^T true when Stab_B is true (!). Moreover, should ${\mathsf{Stab}^\bot}_B$ be allowed, ${\mathsf{efq}^\bot}_B$ can be easily derivable from it.

This analysis makes it clear that we need to work in the minimal setting - i.e., intuitionistic logic without ${\mathsf{efq}^\bot}_A$ - when we want to apply the refined A-Translation without further concerns. This constraint has the benefit that as a consequence, in NA^ω the Double-Negation Translation becomes superfluous. In what follows, we identify some classes of Π^0_2-formulas which are intuitionistically valid, but for which it is possible to construct by a minimal insertion of negations their NA^ω-provable classical equivalents.

8.2 Critical Predicate Symbols

(Berger et al., 2002) identifies the atoms containing some special predicate symbols, called *L-critical*, within a class of formulas that are not definite/goal formulas. In what follows, we prove that double negating these atoms is sufficient in order to turn these formulas into definite/goal ones.

The notion of "positive subformulas" is as given by Definition 2.8.

Definition 8.1 (L-critical predicate symbols)**.** *We consider* Π_2^0*-formulas:*

$$\forall_x H_1 \to ... \to \forall_x H_n \to \forall_y(\vec{G} \to \bot) \to \bot, \tag{8.1}$$

with $H_i, i \in \{1, ..., n\}$ *and* G *quantifier-free and take* L *to be*

$$L := \{H_1, ..., H_n, \vec{G} \to \bot\}. \tag{8.2}$$

Let C_i *be quantifier-free and* R_i *predicate symbols occurring in* L*. Then* L*-critical predicate symbols are specified by*

- $\bot$ *is* L*-critical*
- *if* $(\vec{C}_1 \to R_1) \to ... \to (\vec{C}_m \to R_m) \to R$ *is a positive subformula of* L *and if some* R_i *is* L*-critical, then* R *is* L*-critical* $(*)$*.*

An atom formed with L*-critical predicate symbols is called an* L-critical atom*.*

Remark 8.1. *The set of* L*-critical predicate symbols is the least set of predicate symbols containing* $\bot$ *and closed under* $(*)$*.*

Let us illustrate the intended meaning of Definition 8.1 by some simple and intuitive examples.

Example 11. *We consider a fixed set* L *and* A_i', A_i*,* B, B_i*,* $i = 1, 2$ *atomic formulas different from* $\bot$*, such that the formulas* $C_1, ..., C_4$ *are positive subformulas of* L*. In the following formulas the conclusion* B *is an* L*-critical atom:*

- $C_1 := (\vec{A} \to \bot) \to B$
- $C_2 := \vec{A'} \to (\vec{A} \to \bot) \to B$
- $C_3 := (\vec{A} \to \bot) \to \vec{A'} \to B$*.*

C_2 and C_3 indicate that the order of premise does not play any role. Notice that the premise needs to contain a negation in order for the conclusion to be L-critical. However, in

- $C_4 := ((\vec{A} \to \bot) \to \vec{A'}) \to B$.

A' is not L-critical, since it occurs in a negative formula of L, B is not L-critical either.

Definition 8.1 identifies the situations in which the premise of an implication is relevant, whereas the conclusion is an atom different from $\bot$, so is an irrelevant formula. Such formulas are also not definite. We illustrate this by the following example:

Example 12. *We consider C_1, C_2, C_3 as in Example 11.*

- *C_1 is not a definite formula, because it is neither relevant, nor is $\tilde{\neg}\vec{A}$ in $\mathcal{I}_G$.*
- *Since C_1 is not a definite formula, neither is $C_2 = \vec{A'} \to C_1$.*
- *C_3 is also not definite, since it is not in $\mathcal{R}_D$ and since $\vec{A} \to \bot \notin \mathcal{I}_G$.*
- *Since C_3 is neither in $\mathcal{R}_D$, nor in $\mathcal{D}$, the formula $H := C \to C_3$ is also not definite, for any formula C.*
- *C_3 is however an irrelevant goal formula, since its premise is in $\mathcal{R}_D$ and its conclusion is clearly irrelevant. Thus $C_3 \to C$ is a definite formula, when C is definite.*

C_4 is a definite formula, since its premise is an irrelevant goal formula.

(Berger et al., 2002) proposes to repair some of the situations in which we need the formulas to be definite/goal. In case they are not as such, but contain L-critical atoms, it is possible to transform them into definite/goal formulas, by double negating the atoms formed with L-critical predicate symbols different from $\bot$. By carrying this out recursively over the formulas, we obtain definite/goal formulas, as shown in what follows.

Note: By definition all atomic formulas belong to both $\mathcal{D}$ and $\mathcal{G}$.

In order to understand why the L-critical predicate symbols were defined only for positive subformulas of L, we formulate the following consequence of Definition 3.5, using Definition 2.8 for the notions of positive/negative subformulas:

Lemma 8.1. *For $\mathcal{D}$ and $\mathcal{G}$ the classes of definite, respectively goal formulas, we have that:*

(1) each positive subformula of a $\mathcal{D}$-formula is definite.

(2) each negative subformula of a $\mathcal{D}$-formula is goal.

(3) each positive subformula of a $\mathcal{G}$-formula is goal.

(4) each negative subformula of a $\mathcal{G}$-formula is definite.

Proof. We show each claim by case distinction on the subformulas and treat (1) and (2) simultaneously. The proof for (3) and (4) is very similar, so we leave it as an exercise.

Let A be a definite formula.

(1) We show that its positive subformulas are definite.

Case P. P is a positive subformula of itself and definite.

Case $\forall_x B(x)$. By Definition 3.5, $B(x)$ is also a positive subformula of A and thus by the induction hypothesis it is definite. Then, by Definition 2.8 $\forall_x B(x)$ is definite.

Case $B \to C$. If $B \to C$ is a positive subformula of A, then B is a negative subformula and C a positive subformula of $B \to C$. Using the induction hypothesis, we have from (2) that $B \in \mathcal{G}$ and from (1) that $C \in \mathcal{D}$. Thus, by Definition 3.5, $B \to C$ is a definite formula.

(2) We show that the negative subformulas of A are in $\mathcal{G}$. If the negative subformula of A is

P, then this is a negative subformula of itself and in $\mathcal{G}$.

$\forall_x B(x)$, then by Definition 2.8 $B(x)$ is also a negative subformula of A and thus by the induction hypothesis it is in $\mathcal{G}$. Then, by Definition 3.5 $\forall_x B(x)$ is also in $\mathcal{G}$.

$B \to C$, then B is a positive subformula and C a negative subformula of A. Using the induction hypothesis, we have $A \in \mathcal{D}$ (by (1)) and $B \in \mathcal{G}$ (by (2)). By Definition 3.5, $A \to B$ is therefore a goal formula.

□

Example 13. *(Intuitive view of Lemma 8.1)*

Consider the formula $D := (A \to B) \to C$ to be a definite formula.

By Definition 2.8, A is a positive subformula, B a negative subformula and C a positive subformula of D. The notions of positive and negative subformula are thus alternating.

Likewise, for D to be definite, by Definition 3.5, A needs to be a definite, B a goal (such that $A \to B$ is in $\mathcal{G}$) and C a definite formula. Hence, definite/goal subformulas are alternating in the same manner as the positive/negative subformulas.

Thus, Lemma 8.1 can be viewed as mapping these notions - positive/negative formulas to definite/goal formulas.

8.3 Refined Double Negation

We introduce a special form of double negation, refining the Gödel-Gentzen Translation, in order to turn formulas containing L-critical predicate symbols into definite/goal formulas.

In what follows, we consider as in Definition 8.1 a Π^0_2 formula A for which we have a fixed set L of its quantifier-free assumptions ($\tilde{\exists}$ being unfolded and the actual goal being $\bot$).

Definition 8.2. *[Refined Double-Negation $\cdot^{rg_L}$]*
We call the double negation applied only to the atoms formed by L-critical predicate symbols the refined double-negation translation.
The refined double-negation translation $\cdot^{rg_L}$ is given inductively by:

$$\begin{aligned}
\bot^{rg_L} &:= \bot,\\
\psi^{rg_L} &:= \begin{cases} \tilde{\neg}\,\tilde{\neg}\psi, & \psi \text{ is an L-critical atom}\\ \psi, & \psi \text{ is not L-critical}\end{cases}\\
(\psi_1 \wedge \psi_2)^{rg_L} &:= \psi_1^{rg_L} \wedge \psi_2^{rg_L},\\
(\psi_1 \to \psi_2)^{rg_L} &:= \psi_1^{rg_L} \to \psi_2^{rg_L},\\
(\forall_x \psi)^{rg_L} &:= \forall_x \psi^{rg_L}.
\end{aligned}$$

8.3.1 Obtaining Definite/Goal Formulas

We show in this section how to obtain definite and goal formulas by double negating critical atoms. We illustrate the method on one of the examples shown in previous chapters.

Lemma 8.2. *The refined negative translation turns every positive subformula of L into a definite formula, and every negative subformula of L into a goal formula.*

Proof. Let D be a positive subformula of L with

$$D = (\vec{C}_1 \to R_1) \to ... \to (\vec{C}_m \to R_m) \to R, \tag{8.3}$$

with R, R_i predicate symbols.

We show by induction on formulas that

$$D^{rg_L} = (\vec{C}_1^{rg_L} \to R_1^{rg_L}) \to ... \to (\vec{C}_m^{rg_L} \to R_m^{rg_L}) \to R^{rg_L}$$

is a definite formula. Since D is a positive subformula of L, then so are C_i and by the (IH) every $\vec{C}_i^{rg_L}$ is thus a definite formula. We distinguish the following two situations:

- if some R_i, $1 \leq i \leq m$, is L-critical then R is L-critical, so it appears double negated in D^{rg_L}. Since $\vec{C}_i^{rg_L}$ is definite and quantifier-free and every R_i is atomic, each premise $\vec{C}_i^{rg_L} \to R_i^{rg_L}$ is a goal formula. Thus, by Remark 3.5, $(\vec{C}_i^{rg_L} \to R_i^{rg_L}) \to \cdots \to (\vec{C}_n^{rg_L} \to R_n^{rg_L}) \to \tilde{\neg}\tilde{\neg}R$ is a relevant definite formula.
- if none of the R_i, $1 \leq i \leq m$, is L-critical, then each $\vec{C}_i^{rg_L} \to R_i$ is an irrelevant goal formula. Then, since R is atomic and thus definite $(\vec{C}_1^{rg_L} \to R_1) \to \cdots \to (\vec{C}_n^{rg_L} \to R_n) \to R$ is a definite formula.

□

Example: Infinite boolean tape We illustrate this method on Stolzenberg's Principle presented in Section 5.1. We regard the following formulation of Stolzenberg's Principle (SP):

$$\forall_n (f(n) = 0 \mathbin{\tilde{\vee}} f(n) = 1) \to \tilde{\exists}_{n,m}.\ n < m \mathbin{\tilde{\wedge}} f(n) = f(m).$$

Since the disjunction is not an implicit connector in NA^ω, in Section 5.1.2 we consider rewriting the assumption into the following alternatives:

$$\begin{aligned} A &:\forall_n.\ \tilde{\neg} f(n) = 0 \to \tilde{\neg} f(n) = 1 \to \bot \\ A^{np} &:\forall_n.\ \tilde{\neg} f(n) = 0 \to f(n) = 1. \end{aligned}$$

In order to explore the symmetry so that we can obtain different program in Section 5.1 we have formalized SP using A. It is an easy exercise to verify

that it is also possible to carry out the proof using A^{np}, by slightly changing the proof term for $(\infty_0_or_\infty_1)$.

However, the refined A-Translation cannot be applied to the proof of

$$\forall_n.\ \tilde{\neg}f(n) = 0 \to f(n) = 1 \to \tilde{\tilde{\exists}}_{n,m}.\ n < m\ \tilde{\wedge}\ f(n) = f(m). \tag{8.4}$$

The reason is that A^{np} is not a definite formula, since $\tilde{\neg}f(n) = 0 \notin \mathcal{I}_G$ and $f(n) = 1 \notin \mathcal{R}_D$. In fact, if we define for (8.4) the set

$$L_{\mathrm{SP}} = \{\ \tilde{\neg}f(n) = 0 \to f(n) = 1, n < m \to f(n) = f(m) \to \bot\}$$

then by Definition 8.1 the predicate symbol "=" in $\tilde{\neg}f(n) = 0 \to f(n) = 1$ turns out to be L_{SP}-critical. Thus, $f(n) = 1$ is an L_{SP}-critical atom. If we double negate all atoms in A^{np} containing the L_{SP}-critical "=" we obtain:

$$(A^{np})^{rg_L} : \forall_n.\ \tilde{\neg}\,\tilde{\neg}\,\tilde{\neg}f(n) = 0 \to\ \tilde{\neg}f(n) = 1 \to \bot \in \mathcal{D}.$$

It is however not necessary to double negate all L_{SP}-critical atoms in SP, since we have (also in NA^ω) $\tilde{\neg}\,\tilde{\neg}\,\tilde{\neg}f(n) = 0 \equiv\ \tilde{\neg}f(n) = 0$. The double negation of $f(n) = 1$ gives us A. Likewise, it is not necessary to change $G \to \bot := n < m \to f(n) = f(m) \to \bot$ by the refined double negation translation. In Section 5.1.2 we have given a proof of

$$\forall_n.\ \tilde{\neg}f(n) = 0 \to\ \tilde{\neg}\,\tilde{\neg}f(n) = 1 \to \tilde{\tilde{\exists}}_{n,m}.\ n < m\ \tilde{\wedge}\ f(n) = f(m).$$

We observe that as it is the case with $\bot$, the refined negative translation does not change $\tilde{\neg}R$, even for an L-critical R.

Remark 8.2. *It is an easy exercise to show*

$$\vdash_m (A^{np} \to G) \to A \to G.$$

Since A^{np} clearly follows from A we can thus view $A^{np} \to G$ as a weakening of the form of SP *shown in Section 5.1.*

In conclusion, for formulas A as in (8.1) the refined double negation allows us to change positive formulas D (which are of the kind (8.3) since they are quantifier-free) of L_A into definite formulas. For this it suffices to detect the L_A-critical predicate symbols. The same applies to the negative formulas, which can be turned into goal formulas and this enables us to use the refined A-translation in order to extract programs.

8.3.2 Refined Double Negation of Proofs

In the analysis performed in Chapters 5-7 we have encountered situations where the manual insertion of double negations was necessary in order to carry out the proofs in NA^ω. We have observed that there is a connection with the notion of L-critical atoms, which we will generalize in this section. The claim is that we can eliminate $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$ from the proofs of some Π^0_2-formulas by the refined Double Negation Translation, with the benefit of avoiding Gödel-Gentzen Translation. Negating only certain atoms is the least price to pay, in order to be able to apply the refined A-Translation.

In short, the aim is to repair situations of the kind

$$\begin{aligned}\mathsf{NA}^\omega \cup \{{\mathsf{efq}^\bot}_{\mathrm{F}}, {\mathsf{Stab}^\bot}_{\mathrm{F}}\} &\vdash (B \to \bot) \to B \to \mathrm{F}\\ \text{but } \mathsf{NA}^\omega &\nvdash (B \to \bot) \to B \to \mathrm{F},\end{aligned}$$

where B is a formula in NA^ω such that $\mathsf{NA}^\omega \vdash B$ or B relevant.

Remark 8.3. *Let L be a fixed set of the quantifier-free assumptions in some Π^0_2-formula A and ${\mathsf{efq}^\bot}_{\vec{B}\to R}, {\mathsf{Stab}}_{\vec{B}\to R}$ positive subformulas of L. Then by Definition 8.1, R is an L-critical predicate symbol in ${\mathsf{efq}^\bot}_{\vec{B}\to R}$ and in ${\mathsf{Stab}^\bot}_{\vec{B}\to R}$ and we have:*

(1) $({\mathsf{efq}^\bot}_{\vec{B}\to R})^{rg_L} := \bot \to \vec{B}^{rg_L} \to \tilde{\neg}\,\tilde{\neg} R$, *so trivially*

$$\vdash_m ({\mathsf{efq}^\bot}_{\vec{B}\to R})^{rg_L}.$$

(2) We have

$$({\mathsf{Stab}^\bot}_{\vec{B}\to R})^{rg_L} := \tilde{\neg}\,\tilde{\neg}(\vec{B}^{rg_L} \to \tilde{\neg}\,\tilde{\neg} R) \to \vec{B}^{rg_L} \to \tilde{\neg}\,\tilde{\neg} R,$$

so $\vdash_m ({\mathsf{Stab}^\bot}_{\vec{B}\to R})^{rg_L}$ *by the* NA^ω*-proof*

$$\dfrac{\tilde{\neg}\,\tilde{\neg}(\vec{B}^{rg_L} \to \tilde{\neg}\,\tilde{\neg} R) \qquad \dfrac{\dfrac{\vec{B}^{rg_L} \to \tilde{\neg}\,\tilde{\neg} R \qquad \vec{B}^{rg_L}}{\tilde{\neg}\,\tilde{\neg} R}\to^- \qquad \tilde{\neg} R}{\bot}\to^-}{\bot}\to^+ \quad \tilde{\neg}(\vec{B}^{rg_L} \to \tilde{\neg}\,\tilde{\neg} R)\ \to^-$$

Theorem 8.1 (Refined Double-Negation of Proofs)**.** *Let an NA^ω-proof of the Π^0_2-formula $A := \forall_{\vec{x}} H_1 \to ... \to \forall_{\vec{x}} H_n \to \forall_{\vec{y}}(\vec{G} \to \bot) \to \bot$ be given.*

The refined negative translation of A remains NA^ω-provable, i.e.,

$$\vdash_m A^{rg_L}.$$

Proof. By induction on the derivations.

We consider $L := \{H_1, ..., H_n, \vec{G} \to \bot\}$ and let

$$C := (\vec{C_1} \to R_1) \to ... \to (\vec{C_n} \to R_n) \to R$$

be a positive subformula of L. C can occur in the proof of A in the subderivations M consisting of $\to^+$ and $\to^-$, as follows:

$$\dfrac{\dfrac{v : C \qquad \dfrac{\begin{array}{c}[u_1 \colon \vec{C_1}] \\ | \, N_1 \\ R_1 \end{array}}{\vec{C_1} \to R_1} \to^+ u_1}{(\vec{C_2} \to R_2) \to ... \to (\vec{C_n} \to R_n) \to R} \to^- \qquad \dfrac{\begin{array}{c}[u_n \colon \vec{C_n}] \\ | \, N_n \\ R_n \end{array}}{\vec{C_n} \to R_n} \to^+ u_n}{\begin{array}{c} | \, ... \\ (\vec{C_n} \to R_n) \to R \end{array} \qquad\qquad\qquad\

negation of the selected atoms can be now justified by the application of $\cdot^{rg_L}$, such that these transformations are not ad-hoc, but systematic. In each of the examples presented below it suffices to apply the double negation only to some of the atoms containing L-critical predicate symbols, for a case-specific set L. More precisely, in practice we have that:

Remark 8.4. *For certain intuitionistic proofs, it is not necessary to double negate all occurrences of the L-critical predicate symbols, in oder to translate them into minimal logic proofs.*

This suggests that further improvements of the refined double negation are possible, by which the initial structure of the arithmetical axioms remains unchanged. We will illustrate this by examples.

Remark 8.5. *The Gödel-Gentzen Translation can be viewed as transforming by the refined Double Negation $\cdot^{rg_L}$ formulas consisting of only L-critical atoms. In the next section we will be concerned with using appropriate specifications, such that the number of L-critical predicate symbols is kept to a minimum. This is achieved for instance by rewriting $a \leq b$ to $a < \mathsf{S}b$ or $a < b$ to $a \neq b$.*

8.4 Examples

In the following, we illustrate the above claims on some examples. We first present the problem of finding the least element in a well-founded set, by showing that each such set must increase in two consecutive points. Further, we overview the more complex situations arising in the studies presented Chapters 5, 6 and 7. In each of these examples, the double negation of the L-critical predicate symbols was necessary and sufficient in order to prove in minimal logic the otherwise only intuitionistically valid formulas. In this section, we justify by the notion of L-critical predicate symbols the seemingly ad-hoc decisions of double negating only certain atoms from the previous chapters. We refer the reader to Chapters 5-7 for further details and the context.

Notation 8.1. *$<$, $\leq$ and $=$ bind stronger then $\tilde{\neg}$.*

8.4.1 The "Least" Element in a Well-Founded Set

We first present a "fresh" example, which we hope offers a better understanding of the refined Double Negation due to its simplicity.

We show that each function on natural numbers must increase weakly in (at least) two distinct points, i.e.

$$A := \forall_f(\forall_{m,n}((m < n \to \bot) \to n \le m) \to \tilde{\exists}_k f\, k \le f\, \mathsf{S}k)$$

The function on $\mathtt{N}$ cannot strictly decrease in all points. To show this, we first select from the domain of f the values x such that f increases in all points above x. We collect these values in the set

$$M := \{\, x \mid \forall_{y \ge x} f(x) \le f(y) \,\}.$$

By the (MinPr) we have that such a set is not empty, so for some x its successor belongs to the M, so the requirement in A is fulfilled at this x.

For the formal proof, we instantiate (MinPr) with $m := f$ and $P(k) := \tilde{\exists}_k \mathrm{T}$. For this, we use $\mathsf{GInd}_Q : \forall_{f,n}.\ \forall_j\ \tilde{\neg} Q(j) \to \forall_{b^{\mathbb{B}}}(b \to\ \tilde{\neg}\mathrm{T})$, where $Q(n) := \forall_i(f\, n < f\, i \to\ \tilde{\neg}\mathrm{T})\ \tilde{\wedge}\ \mathrm{T}$. We also use the existence elimination $(\tilde{\exists}^-_{2,1})$ and introduction $(\tilde{\exists}^+_{1,1})$ axioms.

The λ-term of the proof is

$$\begin{aligned}
M_{A^+} &:= \lambda f, u_1.\ (\tilde{\exists}^-_{2,1}\ f\ \lambda u_2.\ \tilde{\exists}^+_{1,1}\ 0\ \mathrm{T}\ \lambda i.\ u_2\ i\ M_1) \\
&\qquad\qquad \lambda k, u_3, u_4, u_5.\ u_5\ k\ M_2^{f\,k \le f\,\mathsf{S}k},\ \text{where} \\
M_1 &:= (\lambda n.\ \mathsf{GInd}_Q\ f\ n\ u_2\ (f\, n < f\, i))^{\forall_n.\ f\,n < f\,i \to\ \tilde{\neg}\mathrm{T}} \\
M_2 &:= u_1\ (f\,\mathsf{S}k)\ (f\,k)\ (\lambda u_6.\ u_3\ \mathsf{S}k\ u_6\ \mathrm{T})^{f\,\mathsf{S}k < f\,k \to \bot},
\end{aligned}$$

with the typed assumption variables: $u_1^{\forall_{m,n}.\ (m<n\to\bot)\to n\le m}$, $u_2^{\forall_n\ \tilde{\neg}Q(n)}$, $u_3^{\forall_n.\ f\,n<f\,k\to\ \tilde{\neg}\mathrm{T}}$, u_4^{T}, $u_5^{\forall_k.\ f\,k\le f\,\mathsf{S}k\to\bot}$ and $u_6^{f\,\mathsf{S}k<f\,k}$.

With the terminology from Theorem 8.1 we have

$$H_1 := (m < n \to \bot) \to n \le m \text{ and } G := f\,k \le f\,\mathsf{S}k\ \Rightarrow\ L_A = \{H_1, G \to \bot\}$$

The proof of H_1 requires $\mathsf{efq}^{\bot}_{\mathsf{S}n\le 0}$, so $\le$ is an L_A-critical predicate symbol. If we double negate it, then

$$H_1^{rg_{L_A}} := (m < n \to \bot) \to (n \le m \to \bot) \to \bot,$$

coincides with the Gödel-Gentzen Translation of H_1, so $H_1^{rg_L}$ is provable by the same proof as H_1. Since $\vdash_m \mathsf{efq}^{\bot}\ \tilde{\to}\ \tilde{\neg}\mathsf{S}n\le 0$ by Remark 8.3, $H_1^{rg_L}$ is provable in minimal logic. In addition, $\vdash_m H_1 \to H_1^{rg_L}$.

Hence,

$$A^{rg_{L_A}} = \forall_f.\ \forall_{m,n}((m < n \to \bot) \to\ \tilde{\neg}\,\tilde{\neg} n \le m) \to \tilde{\exists}_k\ \tilde{\neg}\,\tilde{\neg} f\,k \le f\,\mathsf{S}k$$

is also NA^ω-provable.

However, we can further minimize the $\bot$-occurrences in the minimal logic proof, by observing that $fk \le f\mathsf{S}k \to \bot$ can be interpreted as $f\mathsf{S}k < fk$. The modified version is:

$$A_\bot := \forall_f(\forall_m(m < 0 \to \bot) \to \tilde{\exists}_k(f\,\mathsf{S}k < f\,k \to \bot)$$

With the improved refined double negation, $(m < n \to \bot) \to \tilde{\neg}\,\tilde{\neg} n \le m)$ becomes trivial and $m < 0 \to \bot$ reduces $\mathsf{efq}_\bot$, so it is harmless in NA^ω. Consequently, $A_\bot$ can be proved in minimal logic by:

$$\begin{aligned} M_{A_\bot} := \lambda f\,, v_1.\ (\tilde{\exists}^-_{2,1} f\ \ \lambda u_2.\ \tilde{\exists}^+_{1,1}\ 0\ \mathrm{T}\ \lambda i_1.\ u_2\ i_1 M_1) \\ \lambda k, u_3, u_4, v_2.\ v_2\ k\ \lambda u_6.\ u_3\ \mathsf{S}k\ u_6\ \mathrm{T}, \end{aligned}$$

with the additional typed assumption variables $v_1^{\forall_m(m<0\to\bot)}$ and $v_2^{\forall_k.\ \tilde{\neg}\,\tilde{\neg} f\,\mathsf{S}k<f\,k}$.

8.4.2 The Infinite Pigeonhole Principle (Chapter 5)

In Section 5.2 we give the following specification of IPH:

$$\forall_f.\forall_n f(n) < r \to \tilde{\exists}_q \forall_n \tilde{\exists}_k.\ n \le k\ \tilde{\wedge}\ f(k) = q, \qquad (IPH^+)$$

or, equivalently, $\forall_f.\forall_n f(n) < r \to \forall_n \tilde{\exists}_{q,k}.\ n \le k\ \tilde{\wedge}\ f(k) = q$. For this, we use the induction axiom, the cases axiom and the following lemmas:

$$\begin{aligned} &\forall_{i,j} H_1 : \forall_{i,j}(i < \mathsf{S}j \to (i < j \to \bot) \to (i = j \to \bot) \to \bot) && (Lem_{<\mathsf{S}Cases}) \\ &\forall_{i,j} H_{2,n} : \forall_{i,j} n \le (i \sqcup j), n \in \{i, j\} && (Lem_{n\le(\sqcup)}) \\ &\forall_{i,j,k} H_{3,\circ} : \forall_{i,j,k}(j \circ i \to i \circ k \to j \circ k), \circ \in \{<, \le\} && (Lem_{\circ Trans}) \end{aligned}$$

Additionally, as discussed in Section 5.2.2 and explained below, in the proof of (IPH^+) we need

$$H_4 : \mathsf{efq}^\bot_{f\ (n\sqcup m)<r}$$

Hence, the set of quantifier-free formulas associated to the proof of (IPH^+) is

$$\begin{aligned} L_{\mathrm{IPH}^+} = \{f(n) < r, n \le k \to f(k) = q \to \bot\} \cup \\ \cup\ \{H_1, H_{2,i}, H_{2,j}, H_{3,<}, H_{3,\le}, H_4\}. \end{aligned}$$

Let us overview the situation requiring the use of $\mathsf{efq}^\bot$. In the induction step we want to use

$$\begin{aligned} &(\mathrm{StepH}) && : \forall_n f(n) < \mathsf{S}r \\ &(\mathrm{NegInf})_m && : \forall_k.\ m \le k \to f(k) = r \to \bot, \end{aligned}$$

in order to infer $f\ (n \sqcup m) < r$, where $(n \sqcup m)$ is used as in Section 5.2.2 to denote the maximum of n and m.

(StepH) abbreviates a case distinction, so we use $Lem_{<\mathsf{S}Cases}$ to "unfold" it. We use the substitutions $i := f(n \sqcup m)$, $j := r$ and take $P := \bot$. If we apply this to (StepH) in which we take n to be $(n \sqcup m)$, then it only remains to show that $f(n \sqcup m) = r \to f(n \sqcup m) < r$. From (NegInf) with $k := (n \sqcup m)$ we know that the equality does not hold. Thus, it suffices to show:

$$(f\ (n \sqcup m) = r \to \bot) \to f\ (n \sqcup m) = r \to f\ (n \sqcup m) < r,$$

which amounts to using $\mathsf{efq}^{\bot}{}_{f\ (n \sqcup m) < r}$. This makes *"<" an L_{IPH^+}-critical predicate symbol.*

An analysis of the formulas in L shows that no further predicate symbols are L-critical. Therefore, it suffices to double-negate the atoms containing "<" and we have[1]:

$$\begin{aligned} L^{rg_L}_{\mathrm{IPH}^+} = \{\ \tilde{\neg}\,\tilde{\neg} f(n) < r, n \le k \to f(k) = q \to \bot\} \cup \{\bot \to\ \tilde{\neg}\,\tilde{\neg} f(n \sqcup m) < r\} \cup \\ \cup \{\ \tilde{\neg}\,\tilde{\neg} i < \mathsf{S}j \to (i < j \to \bot) \to (i = j \to \bot) \to \bot\} \cup \\ \cup\ \{H_{2,i}, H_{2,j}, H_{3,\le},\ \tilde{\neg}\,\tilde{\neg} j < i \to\ \tilde{\neg}\,\tilde{\neg} i < k \to\ \tilde{\neg}\,\tilde{\neg} j < k\}. \end{aligned}$$

Thus, $Lem_{<\mathsf{S}Cases}$ and $Lem_{<Trans}$ have been changed by taking the double negation of the L_{IPH^+}-critical predicate symbols. Their refined double negation coincides in this case with the Gödel-Gentzen translation, so we know that $Lem^{rg}_{<\mathsf{S}Cases}$ and $Lem^{rg}_{<Trans}$ are also valid and provable by the same derivations as the original lemmas.

Since it was necessary to double negate only the atoms containing $<$ and we have changed *all* these atoms, all the inference rules in which these atoms occur remain valid. Any other rules remain intact. Further, $(\mathsf{efq}^{\bot}{}_{f(n \sqcup m) < r})^{rg_L}$ is NA^{ω}-provable and thus the refined double-negated translation of IPH^+:

$$(\mathrm{IPH}^+)^{rg_L} : \forall_f.\ \forall_n\ \tilde{\neg}\,\tilde{\neg} f(n) < r \to \tilde{\exists}_q \forall_n \tilde{\exists}_k.\ n \le k\ \tilde{\wedge}\ f(k) = q$$

is also provable in minimal logic.

Remark 8.6. *In Section 5.2.1 we show on page 84 an improved[2] version of* IPH*, in which we use the original variants of the lemmas $Lem_{<\mathsf{S}Cases}$ and $Lem_{<Trans}$. Hence, it turns out that it suffices to double negate the*

[1] We can safely take $i < j \to \bot$ instead of $\tilde{\neg}\,\tilde{\neg} i < j \to \bot$.

[2] Improved with respect to the refined double negation in the sense that even fewer double negations are inserted.

L_{IPH^+}-critical atom $f(n) < r$, which is equivalent modulo the substitution of n by $(n \sqcup m)$ to the atom occurring in the intuitionistically-valid H_4.

For this proof we have:

$$L_{(\mathrm{IPH}_r)} = \{ \tilde{\neg}\,\tilde{\neg} f(n) < r, n \leq k \to f(k) = q \to \bot\} \cup \{\bot \to \tilde{\neg}\,\tilde{\neg} f(n \sqcup m) < r\} \cup \\ \cup \{H_1, H_{2,i}, H_{2,j}, H_{3,<}, H_{3,\leq}, H_4\}$$

8.4.3 Dickson's Lemma (Chapter 7)

In the case study concerning Dickson's Lemma we show the Π^0_2-corollary:

$$\forall_{f,g} \tilde{\exists}_{i,j}.\ i < j \mathbin{\tilde{\wedge}} f(i) \leq f(j) \mathbin{\tilde{\wedge}} g(i) \leq g(j). \qquad (\mathrm{DL}^+)$$

using the Minimum Principle (MinPr), which follows from the generalized induction scheme:

$$\forall_{m,x}.\ \forall_x(\forall_{y|m(y)<m(x)} R(y) \to R(x)) \to R(x)$$

We also use the following lemmas (where the free variables are considered to be universally quantified)

$$\begin{array}{ll} H_{1,\circ} : i < j \to j \circ k \to i < k, \circ \in \{<, \leq\} & (Lem_{\circ Trans}) \\ H_2 : i < \mathsf{S}j \to i \leq j & (Lem_{<\mathsf{S}To\leq}) \\ H_3 : (i < j \to \bot) \to j \leq i & (Lem_{\not<To\leq}) \end{array}$$

Thus,

$$L_{\mathrm{DL}^+} = \{i < j \to f(i) \leq f(j) \to g(i) \leq g(j) \to \bot\} \cup \\ \cup \{H_{1,<}, H_{1,\leq}, H_2, H_3\}$$

By Definition 8.1, H_3 makes $\leq$ *an L_{DL^+}-critical predicate symbol.*

Let us first overview the context from Section 7.4.1 in which the use of H_3 was necessary. We have used the properties for i and j resulting from the application of (MinPr) and have reached an impasse when we had to show in minimal logic that

$$(f(j) < f(i) \to \bot) \to f(i) \leq f(j). \qquad (8.5)$$

This is equivalent to H_3 and its proof by induction requires $\mathsf{efq}^{\bot}_{\mathsf{S}m\leq 0}$.

The use of $Lem_{\leq Trans}$ turns $<$ into an L_{DL^+}-critical predicate symbol as well. However, this can be avoided by noticing that the lemma can be safely replaced in the proof of DL^+ by

$$Lem_{\mathsf{S}Trans} : \forall_{i,j,k}.\ i < j \to j < \mathsf{S}k \to i < k$$

In general we have that:

Remark 8.7. *In order to minimize the number of double negations inserted, it is essential to avoid the alternation of predicate symbols and to keep the lemmas as uniform as possible in this respect.*

The refined double negation of $Lem_{\leq Trans}$ coincides in this case with the Gödel-Gentzen Translation, so its proof remains unchanged, and $H_2^{rg_L}$ can be easily inferred from H_2 ($Lem_{<\mathsf{S}To\leq}$). However, as one can see in the proof of DL^+ in Section 7.4, these auxiliary lemmas on the natural numbers do not need to be changed by the insertion of double negations.

Thus,

$$\begin{aligned} L^{rg_L}_{\mathrm{DL}^+} = \{&i < j \to \ \tilde{\neg}\,\tilde{\neg} f(i) \leq f(j) \to \ \tilde{\neg}\,\tilde{\neg} g(i) \leq g(j) \to \bot\} \cup \\ &\cup \{H_{1,<}, i < j \to j < \mathsf{S}k \to i < k, H_2, (i < j \to \bot) \to \ \tilde{\neg}\,\tilde{\neg} j \leq i\} \end{aligned}$$

$H_3^{rg_L}$ becomes provable in minimal logic, since we have:

$$\vdash_m \mathsf{efq}^{\bot}\ {}_{\tilde{\neg}\,\tilde{\neg}\mathsf{S}m\leq 0}.$$

In Section 7.4 we show in fact an improved form of the refined double negation of DL^+, (DLCor):

$$\tilde{\exists}_{i,j}.\ i < j \ \tilde{\wedge}\ \ \tilde{\neg}(f(j) < f(i)) \ \tilde{\wedge}\ \ \tilde{\neg}(g(j) < g(i))$$

This is based on an observation which we generalize in Section 8.5 that

$$\tilde{\neg}(f(i) \leq f(j)) \text{ is the complement of } f(j) < f(i).$$

Hence, we can take the simple negation of $f(j) < f(i)$ and eliminate some of the (logical) negations. Further, $H_3^{rg_{L_{DL}}}(i < j \to \bot) \to \ \tilde{\neg}\,\tilde{\neg} j \leq i$ becomes $(i < j \to \bot) \to \ \tilde{\neg} i < j$, which trivially holds. We therefore have:

$$\begin{aligned} L_{\mathrm{DL}} = \{&i < j \to \ \tilde{\neg} f(j) < f(i) \to \ \tilde{\neg} g(j) < g(i) \to \bot\} \cup \\ &\cup \{H_{1,<}, i < j \to j < \mathsf{S}k \to i < k, H_2\} \end{aligned}$$

A proof in minimal logic of this simplified refined double negation of DL has been given in Section 7.4.1.

8.4.4 The Erdös-Szekeres Theorem (Chapter 6)

This is the most elaborated example with respect to the insertion of negations. In the proof of the Erdös-Szekeres Theorem, the double negations were necessary from a two-folded perspective. First, as underlined in Section 6.2.1, in order to use FPH in a minimal logic proof of the Erdös-Szekeres

Theorem, the assumption formula in ES requires to be double negated. Accordingly, the assumption in FPH needs to be matched. Initially, the formulation of FPH was unaffected by the double negations, although its auxiliary lemmas needed to be modified, in order for FPH to be NA^{ω}-provable. The proof of FPH is presented in Section 6.1. The refined double-negated version of FPH needed in the proof of ES is provable in minimal logic - as presented in Section 6.2.2, but as a consequence of double negating the L_{ES}-critical atoms in FPH, its auxiliary lemmas (such as "dnColLists") need to be also adapted. In this section, we overview these situations and give a more systematic interpretation, in view of the notion of (L-) critical predicate symbols.

For this, we have to first determine L_{FPH} and L_{ES}. In Chapter 6 we have given an overview of the relevant lemmas used in order to prove FPH and ES - for the purpose of completion, we explicitely state in what follows the other lemmas used in the proofs. We refer the reader to Chapter 6 and to Appendix A.1 for an overview of the other lemmas used. The free variables in these lemmas are universally quantified, unless otherwise specified. We capture these lemmas in the following sets:

$$\begin{aligned}
L^{\mathtt{N}}_{\mathrm{FPH}} &= \{i \circ j \to j \circ k \to i \circ k, i < \mathsf{S}j \to i \leq j\} \cup \\
&\quad \cup \{i_1 \leq j_1 \to i_2 \leq j_2 \to i_1 \cdot i_2 \leq j_1 \cdot j_2\}, \circ \in \{<, \leq\} \\
L^{\mathtt{N}}_{\mathrm{ES}} &= \{i \circ j \to j \circ k \to i \circ k, i = j \to j = i\} \cup \\
&\quad \cup \{\mathsf{S}i \leq j \to i = j \to \bot, (i \leq j \to P) \to (j \leq i \to P) \to P\}, \circ \in \{<, =\}
\end{aligned}$$

FPH

In Section 6.1 we show

$$\forall_{k,n,r,c}.\ \forall_i c(i) < r \to k \cdot r < n \to \tilde{\exists}_l.\ k < |l| \mathbin{\tilde{\wedge}} \tilde{\exists}_m \forall_{i<|l|} c(l_i) = m$$

using the auxiliary properties[3]

$$\begin{aligned}
&H_1 : (i < |ll| \to |ll_i| \leq k) \to |\mathsf{flatten}(ll)| \leq k \cdot |ll| && \text{(Lhflattenll)} \\
&H_2 : i \leq r \to j < |\mathsf{GC}_i| \to c(\mathsf{GC}_{i,j}) = i && \text{(GroupByCols)} \\
&H_3 : c(i) < r \to |\mathsf{GC}| \leq r && \text{(ColLists)} \\
&H_4 : |\mathsf{flatten}(\mathsf{GC})| = \mathsf{S}n && \text{(ObjGrouped)} \\
&H_5 : i \leq r \to \mathsf{S}j < |\mathsf{GC}_i| \to \mathsf{GC}_{i,\mathsf{S}j} < \mathsf{GC}_{i,j} && (\mathsf{GC}\mathit{Distinct})
\end{aligned}$$

[3]We consider the free variables in these lemmas to be implicitly universally quantified.

The proof requires also

$$H_6 : (i < j \to \bot) \to j \leq i, \qquad (Lem_{\not<To\leq})$$

which in turn amounts to using $\mathsf{efq}^{\bot}_{\mathsf{S}j\leq 0}$. For an explanation why $Lem_{\not<To\leq}$ is needed in the proof of FPH, please refer to the discussion on page 101. We collect all these assumptions in L_{FPH^+}:

$$\begin{aligned} L_{\mathrm{FPH}^+} = {} & \{c(i) < r, k \cdot r < n, k < |l| \to (i < |l| \to c(l_i) = m) \to \bot\} \cup \\ & \cup \{H_1, H_2, H_3, H_4, H_5, H_6\} \cup \{(i < j \to \bot) \to j \leq i\} \cup L^{\mathtt{N}}_{\mathrm{FPH}} \end{aligned}$$

where for brevity we have used $GC := \mathsf{GroupbyCols}(c, n)$ as in Chapter 6.

Thus because of H_6 *"$\leq$" is an L_{FPH^+}-critical predicate symbol.*

Double negating the atoms containing $\leq$ triggers the need to change all lemmas in which "$\leq$" appears. This is the case first of all with (Lhflattenll):

$$\forall_{ll,k}.\ (\forall_i.\ i < |ll| \to\ \tilde{\neg}\,\tilde{\neg}|ll_i| \leq k) \to\ \tilde{\neg}\,\tilde{\neg}|\mathsf{flatten}(ll)| \leq k \cdot |ll|$$

Changing (Lhflattenll) means that we need to readapt its proof by double negating all atoms containing "$\leq$". In the proof of (Lhflattenll), we have used the lemmas:

$$\begin{aligned} & H'_1 : \forall_{i,j}.\ i + j = j + i && (Lem_{+Com}) \\ & H'_2 : \forall_{i_1,i_2,j_1,j_2}.\ i_1 \leq j_1 \to i_2 \leq j_2 \to i_1 + i_2 \leq j_1 + j_2 && (Lem_{PlusLt}) \end{aligned}$$

Since "$\leq$" is L_{FPH^+}-critical in (Lhflattenll), we need to adapt H'_2 by considering $Lem^{rg_L}_{PlusLt}$, which is valid because the refined double negation coincides with the Gödel-Gentzen translation.

This is the first case when the negative subformula is not atomic. Let us verify that this complies with the requirements from Theorem 8.1. The negative subformula of (Lhflattenll) is $i < |ll| \to |ll_i| \leq k$ and since $\leq$ is L-critical, we need to require that $<$ is also L-critical. We can avoid this if we use the same "trick" as for Dickson's Lemma: the observation that "$\leq$" is the complement relation to "$<$". If we replace the double negation of "$\leq$" by the simple negation of "$<$", then we obtain (nLhflattenll):

$$\forall_{ll,k}.\ (\forall_{i<|ll|}.\ k < |ll_i| \to \bot) \to k \cdot |ll| < |\mathsf{flatten}(ll)| \to \bot,$$

in which $<$ does not need to be L-critical. In Section 6.1.1 we give the proof of (nLhflattenll) using $Lem^{rg_L}_{PlusLt}$ (labeled in Chapter 6 as $(Lem_{nPlusLt})$):

$$\forall_{i_1,j_1,i_2,j_2}.\ (i_1 < j_1 \to \bot) \to (i_2 < j_2 \to \bot) \to i_1 + i_2 < j_1 + j_2 \to \bot.$$

No further predicate symbol in FPH is L_{FPH^+}-critical. This is a consequence of the fact that certain occurrences of "$\leq$" can be safely changed to "$< \mathsf{S}$", as follows:

- (GroupByCols) (H_2) stated initially as

$$\forall_{i,j}.\, i \leq r \to j < |\mathsf{GC}_i| \to c(\mathsf{GC}_{i,j}) = i$$

turns "=" into an L_{FPH^+}-critical predicate symbol. However, we can safely reformulate the lemma as

$$\forall_{i,j}.\, i < \mathsf{S}r \to j < |\mathsf{GC}_i| \to c(\mathsf{GC}_{i,j}) = i$$

and eliminate this issue.

- $Lem_{\mathsf{GC}distinct}$ (H_5):

$$i \leq r \to \mathsf{S}j < |\mathsf{GC}_i| \to \mathsf{GC}_{i,\mathsf{S}j} < \mathsf{GC}_{i,j}$$

can be changed to

$$i < \mathsf{S}r \to \mathsf{S}j < |\mathsf{GC}_i| \to \mathsf{GC}_{i,\mathsf{S}j} < \mathsf{GC}_{i,j}.$$

- Further, although in the statement of (ColLists) (H_3)

$$\forall_i c(i) < r \to |\mathsf{GC}| \leq r$$

"$\leq$" is L-critical and the atom $|\mathsf{GC}| \leq r$ needs to be double negated, we can avoid this if we formulate (ColLists) as

$$\forall_i c(i) < r \to |\mathsf{GC}| < \mathsf{S}r.$$

Remark 8.8. *It should be clear from the explanation on page 101 (Chapter 6) that replacing $\leq$ by its complement in* (Lhflattenll) *would not be enough in order to carry out the proof of* FPH *in minimal logic. We would still need to eliminate $\bot$ from the assumption corresponding to the unfolding of $\tilde{\exists}$. More precisely, we would have to show $(k < |\mathsf{GC}_i| \to \bot) \to |\mathsf{GC}_i| < \mathsf{S}k$, which amounts to using $\mathsf{efq}^{\bot}$.*

In Section 6.1.1 we show that all the above changes are valid and that the lemmas on natural numbers which we have gathered in $L^{\mathtt{N}}_{\mathrm{FPH}}$ do not need to be changed. We give the proof terms of FPH in which the only double negated atoms are those occurring in (nLhflattenll) and ($Lem_{nPlusLt}$).

Since $i < j \to \bot \to \ \tilde{\neg} i < j$ is trivial, we have correspondingly the following refined double negation of L_{FPH^+}:

$$\begin{aligned} L^{rg_L}_{\mathrm{FPH}^+} = {} & \{c(i) < r, k \cdot r < n, k < |l| \to (i < |l| \to c(l_i) = m) \to \bot\} \cup \\ & \cup \{(i < |ll| \to \ \tilde{\neg} k < |ll_i|) \to \ \tilde{\neg} k \cdot |ll| < |\mathsf{flatten}(ll)|\} \cup \\ & \cup \{i < \mathsf{S}r \to j < |\mathsf{GC}_i| \to c(\mathsf{GC}_{i,j}) = i\} \cup \\ & \cup \{c(i) < r \to |\mathsf{GC}| < \mathsf{S}r, |\mathsf{flatten}(\mathsf{GC})| = \mathsf{S}n\} \cup \\ & \cup \{i < \mathsf{S}r \to \mathsf{S}j < |\mathsf{GC}_i| \to \mathsf{GC}_{i,\mathsf{S}j} < \mathsf{GC}_{i,j}\} \cup \\ & \cup \{(i < j \to \bot) \to \ \tilde{\neg}\, \tilde{\neg} j \le i\} \cup L^{\mathtt{N}}_{\mathrm{FPH}} \end{aligned}$$

Since $\le$ does not occur in the formulation of FPH the initial statement

$$\forall_{k,n,r,c}.\, \forall_i c(i) < r \to k \cdot r < n \to \tilde{\exists}_l.\, k < |l| \mathbin{\tilde{\wedge}} \tilde{\exists}_m \forall_{i<|l|} c(l_i) = m.$$

does not change under the refined double-negation translation. In Section 6.1 we give its NA^ω-proof.

ES

We have presented in Section 6.2 and formalized in Section 6.2.3 a proof of the following formulation of the Erdös-Szekeres Theorem:

$$\forall_{n^{\mathtt{N}},s^{\mathtt{L}}}.\, |s| = n^2 + 1 \to \tilde{\exists}_i.\, \mathsf{S}n \le |f_{s,i}| \mathbin{\tilde{\vee}} \mathsf{S}n \le |g_{s,i}| \tag{8.6}$$

Rewriting $\vee$ according to the interpretation given in Chapter 2 (page 15) turns this into

$$\forall_{n^{\mathtt{N}},s^{\mathtt{L}}}.\, |s| = n^2 + 1 \to \tilde{\exists}_i.\ \tilde{\neg}(\mathsf{S}n \le |f_{s,i}|) \to \ \tilde{\neg}(\mathsf{S}n \le |g_{s,i}|) \to \bot$$

In order to show ES, we use FPH

$$\forall_{k^{\mathtt{N}},n^{\mathtt{N}},r^{\mathtt{N}},c^{\mathtt{N}\Rightarrow\mathtt{N}}}.\, \forall_i c(i) < r \to k \cdot r < n \to \tilde{\exists}_{l,m}.\, k < |l| \mathbin{\tilde{\wedge}} \forall_{i<|l|} c(l_i) = m$$

and the further additional lemmas:

$$\begin{aligned} & H_1 : i < j \to s_i \le s_j \to \mathsf{S}|a_i| \le |a_j| && (s_i \le s_j) \\ & H_2 : a_i \le n \to b_i \le n \to c(a_i, b_i) \le n^2 && (\mathsf{ColUB}) \\ & H_3 : |a_i| \le n \to |b_i| \le n \to |a_j| \le n \to |b_j| \le n \to \\ & \qquad c(a_i, b_i) = c(a_j, b_j) \to |a_i| = |a_j| \wedge |b_i| = |b_j| && (\mathsf{2Col}_=) \end{aligned}$$

As discussed below, in order to carry out the proof we also need H_4, with

$$H_4 : (n \leq m \to \bot) \to m < n. \qquad (Lem_{\not\leq To<})$$

Collecting all these formulas and using the abbreviations $a_k := \mathsf{Bar} f(s, \mathsf{S}k), b_k := \mathsf{Bar} f(s, \mathsf{S}k), k \in \{i, j\}$ and c for the coloring function col, we construct:

$$\begin{aligned} L_{\mathrm{ES}^+} := \{|s| = n^2 + 1, (\ \tilde{\neg}(\mathsf{S}n \leq |f_{s,i}|) \to\ \tilde{\neg}(\mathsf{S}n \leq |g_{s,i}|) \to \bot) \to \bot\} \cup \\ \cup \{c(i) < r, k \cdot r < n, k < |l| \to (i < |l| \to c(l_i) = m) \to \bot\} \cup \\ \cup \{H_1, H_2, H_3, H_4\} \cup L^{\mathtt{N}}_{\mathrm{ES}} \end{aligned}$$

Let us explain why $Lem_{\not\leq To<}$ is needed in order to prove ES. For this, we introduce the abbreviations:

$$\begin{aligned} A &:= \mathsf{S}n \leq |f_{s,i}| & B &:= \mathsf{S}n \leq |g_{s,i}| \\ A^n &:= |f_{s,i}| < \mathsf{S}n \equiv |f_{s,i}| \leq n & B^n &:= |g_{s,i}| < \mathsf{S}n \equiv |g_{s,i}| \leq n \end{aligned}$$

$$\text{and } C(n, r) := c(f_{s,i}, g_{s,i}) \leq n^2,$$

with A and A^n, respectively B and B^n being complementary.

In the informal proof (on page 107), we have reached the desired contradiction using FPH. In order to validate the hypothesis of FPH, we have assumed A^n and B^n and used them to derive $C(n, r)$. For this, it suffices to use (ColUB), with the appropriate NA^ω-valid instance:

$$A^n \to B^n \to C(n, r). \qquad (A^n B^n C)$$

However, our assumption is $(\ \tilde{\neg} A \to\ \tilde{\neg} B \to \bot) \to \bot$, so in order to derive $C(n, r)$ in minimal logic, we need $(Lem_{\not\leq To<})$. The proof by induction of this lemma needs $\vdash_m \mathsf{efq}^\bot\ \tilde{\to}\ \tilde{\neg}_{m<0}$ for the base step, so it is not NA^ω-valid. Its refined double negation

$$\forall_{n,m}.\ (n \leq m \to \bot) \to (m < n \to \bot) \to \bot$$

is provable in minimal logic, since by Remark 8.3 $\vdash_m \mathsf{efq}^\bot\ \tilde{\to}\ \tilde{\neg}_{m<0}$.

$(Lem_{\not\leq To<})$ turns *"<" into an L_{ES^+}-critical predicate symbol.*

Therefore, we need to consider the double negations of the atoms containing it. As the proof in Section 6.2 shows, it suffices however to double negate the critical atom $c(i) < r$. More precisely, we observe that:

- The refined double negation affects only FPH. As in the previous examples, it is important to note that in the proof given in Section 6.2 we

consider an improvement of the refined double negation, by which not all atoms in which "<" occurs are double negated. Indeed, it suffices to consider the atom for which $\mathsf{efq}^{\bot}$ was necessary. The (improved) refined double negation of FPH is (FPH_{dn}):

$$\forall_{k,n,r,c}.\, \forall_i\, \tilde{\neg}\,\tilde{\neg}(c(i) < r) \to k \cdot r < n \to \\ \tilde{\exists}_l.\, k < |l| \,\tilde{\wedge}\, \tilde{\exists}_m \forall_i.\, i < |l| \to c(l_i) = m.$$

For this, we have to revise $L^{rg_L}_{\mathrm{FPH}}$, in order to prove (FPH_{dn}) in minimal logic, since this contains the L_{ES^+}-critical predicate symbol $<$. We have given in Section 6.2.2 a proof of FPH_{dn} from which it is clear that we need to adapt[4] (ColLists), but that (GroupByCols), $Lem_{\mathsf{GC}distinct}$ and (nLhflattenll) can be left unchanged. By Theorem 8.1 "<" is L-critical in $k < |l| \to (i < |l| \to c(l_i) = m) \to \bot$, so "=" should also be L-critical by Definition 8.1. As it turns out these atoms and $k \cdot r < n$ do not occur in the same rules as $c(i) < r$, so they can be left intact. We thus take:

$$\begin{aligned} L^{rg_L}_{\mathrm{FPH}_{dn}} = {} & \{\, \tilde{\neg}\,\tilde{\neg} c(i) < r, k \cdot r < n, k < |l| \to (i < |l| \to c(l_i) = m) \to \bot\} \cup \\ & \cup\, \{(i < |ll| \to\ \tilde{\neg} k < |ll_i|) \to\ \tilde{\neg} k \cdot |ll| < |\mathsf{flatten}(ll)|\} \cup \\ & \cup\, \{i < \mathsf{S}r \to j < |\mathsf{GC}_i| \to c(\mathsf{GC}_{i,j}) = i\} \cup \\ & \cup\, \{\, \tilde{\neg}\,\tilde{\neg} c(i) < r \to\ \tilde{\neg}\,\tilde{\neg} |\mathsf{GC}| < \mathsf{S}r, |\mathsf{flatten}(\mathsf{GC})| = \mathsf{S}n\} \cup \\ & \cup\, \{i < \mathsf{S}r \to \mathsf{S}j < |\mathsf{GC}_i| \to \mathsf{GC}_{i,\mathsf{S}j} < \mathsf{GC}_{i,j}\} \cup \\ & \cup\, \{(i < j \to \bot) \to\ \tilde{\neg}\,\tilde{\neg} j \le i\} \cup L^{\mathtt{N}}_{\mathrm{FPH}} \end{aligned}$$

In the case of (ColLists), double negating the atom $c(i) < r$ imposes the double negation of the atom $|\mathsf{GC}| < \mathsf{S}r$. In this case the refined double negation coincides with the Gödel-Gentzen Translation. Thus, the proof of "dnColLists" can be constructed from the proof of (ColLists); on page 114 we have given an alternative proof, which we compare to that of (ColLists).

- Because of $(s_i \le s_j)$, "$\le$" should be by Definition 8.1 an L_{ES^+}-critical predicate symbol. However, the intended meaning of $i < j$ is $i \neq j$, so that we can take $(s_i \le s_j)$ to be

$$H'_2 : (i = j \to \mathrm{F}) \to s_i \le s_j \to \mathsf{S}|a_i| \le |a_j|.$$

[4] Adapt in the sense of double negating the occurrences of the L_{ES^+}-critical (hence $L^{rg_L}_{\mathrm{FPH}^+}$-critical) predicate symbol "<".

- Since the L_{ES^+}-critical predicate symbol "$<$" does not occur in the formulation of ES, this does not need to be changed.

Remark 8.9. *As we have seen when analyzing the proof of* FPH*, "$\leq$" is an L_{FPH^+}-critical predicate symbol. However, since this does not occur in the formulation of* FPH*, but only in the lemmas used to prove it, and the latter are not explicit in the proof of L_{ES^+}, "$\leq$" is not L_{ES^+}-critical.*

Summing up, we have the following collection of refined double-negated formulas used to prove ES:

$$\begin{aligned} L^{rg_L}_{\mathrm{ES}^+} := \{&|s| = n^2 + 1, (\ \tilde{\neg}(\mathsf{S}n \leq |f_{s,i}|) \to\ \tilde{\neg}(\mathsf{S}n \leq |g_{s,i}|) \to \bot) \to \bot\} \cup \\ &\cup \{\ \tilde{\neg}\,\tilde{\neg} c(i) < r, k \cdot r < n, k < |l| \to (i < |l| \to c(l_i) = m) \to \bot\} \cup \\ &\cup \{(n \leq m \to \bot) \to\ \tilde{\neg}\,\tilde{\neg} m < n\} \cup \\ &\cup \{H_1, H_2', H_3\} \cup L^{\mathtt{N}}_{\mathrm{ES}} \end{aligned}$$

where the latter set is unchanged, since we do not need to modify the lemmas on the natural numbers.

Section 6.2.3 demonstrates that with these changes ES becomes provable in minimal logic.

To sum up, the case studies from previous chapters have revealed that

- the refined double negation is necessary not only in order to obtain definite/goal formulas, but also in order to obtain minimal logic proofs
- it is not necessary to double negate all occurrences of the L-critical predicate symbols; in some cases, it suffices to consider only the atoms for which efq is applied. We have considered in the above examples such improved variants, but we would have obtained NA^ω-derivability by the application of $\cdot^{rg_L}$. Although this would have introduced more negations then these improved variants, the logical falsities that are inserted in this way are clearly fewer then the ones in the Gödel-Gentzen translated proofs, in particular in view of the next observation.
- it is essential to find suitable formulations of the lemmas, such that the variety of predicate symbols occurring in one formula is minimized. If, on the contrary, the predicate symbols are mixed in a subformula, one risks of unnecessarily turning further predicates into L-critical ones, which propagates further in the other premises.

In what follows, we give a further improvement of the double negation, by which we insert only one logical falsity instead of two.

8.5 Further Refinement of the Double Negation

Based on the case studies presented in the previous chapters, we can go even a step further and in some cases insert only one logical falsity ($\bot$), instead of double-negating all predicate symbols occurring in the proof.

In general one negation is not enough for obtaining definite/goal formulas:

Remark 8.10. *The (simple) negation of*

- *a decidable or relevant $\mathcal{D}$-formula is in the $\mathcal{G} \setminus \mathcal{D}$-class.*
- *a formula from $\mathcal{G}$ belongs to $\mathcal{R}_D \setminus \mathcal{G}$ (therefore implicitly also to $\mathcal{D} \setminus \mathcal{G}$).*

It is however possible to mix the logical and arithmetical falseties in the following way:

Remark 8.11.

- *If $\vec{D} \in \mathcal{D}$, then $(\vec{D} \to \mathrm{F}) \to \bot \in \mathcal{D}$.*
- *If $\vec{G} \in \mathcal{I}_G$ decidable, then $(\vec{G} \to \mathrm{F}) \to \bot \in \mathcal{G}$.*

Let us investigate in which situations it is possible to replace the first negation by the arithmetical version, while aiming for the same results as with the refined double-negation translation.

We first make the following observation:

Remark 8.12. (a) *If $\mathsf{NA}^\omega \vdash B \to C$ then*

$$\mathsf{NA}^\omega \vdash ((B \to \mathrm{F}) \to \bot) \to (C \to \mathrm{F}) \to \bot.$$

(b) *If $\mathsf{NA}^\omega \vdash (B \to \mathrm{F}) \to (C \to \mathrm{F})$ then*

$$\mathsf{NA}^\omega \vdash (B \to \bot) \to C \to \bot,$$

provided that B has no free predicate variables.

Proof. (a) Assume $H_a : B \to C$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{H_a} \quad \dfrac{}{u_1 : B}}{C} \to^- \quad \dfrac{}{u_2 : C \to \mathrm{F}}}{\mathrm{F}}}{B \to \mathrm{F}} \to^+ u_1 \qquad \dfrac{}{u_3 : (B \to \mathrm{F}) \to \bot}}{\bot} \to^-$$

(b) Assume $H_b : (B \to \mathrm{F}) \to (C \to \mathrm{F})$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vert}}{H_b} \quad u_1 : B \to \mathrm{F}}{C \to \mathrm{F}} \to^- \quad u_2 : C}{\mathrm{F}} \to^-}{(B \to \mathrm{F}) \to \mathrm{F}} \to^+ u_1}{B} \mathsf{Stab}_B \quad \dfrac{}{u_4 : B \to \bot}}{\bot} \to^-$$

□

We thus define a mixed double negation to be:

Definition 8.3. *Let A^{rg_L} be the refined double-negation translation of a formula A. Then A^{Fg_L} consist in replacing the first logical falsity in the double negation by* F*:*

$$\begin{aligned}
\bot^{rg_L} &:= \bot, \\
\psi^{rg_L} &:= \begin{cases} \tilde{\neg}\neg\psi, \psi \textit{ is an } L\textit{-critical atom} \\ \psi, \psi \textit{ not } L\textit{-critical} \end{cases} \\
(\psi_1 \wedge \psi_2)^{Fg_L} &:= \psi_1^{Fg_L} \wedge \psi_2^{Fg_L}, \\
(\psi_1 \to \psi_2)^{Fg_L} &:= \psi_1^{Fg_L} \to \psi_2^{Fg_L}, \\
(\forall_x \psi)^{Fg_L} &:= \forall_x \psi^{Fg_L}.
\end{aligned}$$

It is easy to see that the following can be proved using $\mathsf{efq}_\bot$:

Proposition 8.1. *If $\mathsf{NA}^\omega \vdash A^{rg_L}$ then $\mathsf{NA}^\omega \vdash A^{Fg_L}$.*

Double negation of relations The mixed negation is of particular importance when the formulas involve a relation $\mathcal{R}$ with an explicit complement. For instance, $x \leq y$ can be seen as an abbreviation of $y < x \to \mathrm{F}$, so its "logical" dual is $y < x \to \bot$ (and variants). The reverse can be also applied: $y < x \to \bot$ has $x \leq y$ as dual.

As a consequence, wherever meaningful, we can substitute the first occurrence of $\bot$ in the refined double-negation of the L-critical formulas by F and replace R by its complement - for instance, $y < x \to \mathrm{F}$ by $x \leq y$.

Example 14. *We overview the situations in Chapters 5-7 in which this observation is applicable.*

- *Dickson's Lemma. We have already exploited this idea and have rewritten* $f(i) \leq f(j)$ *to* $\tilde{\neg} f(j) < f(i)$*, in order to unify the specification with the use of the Minimum Principle. The double negation of* $f(i) \leq f(j)$ *was substituted by* $\tilde{\neg} f(j) < f(i)$*, and as a result, the formulation of the simplified corollary for Dickson's lemma was as in* (DL_2)*.*

- *In the proof of* FPH *from Chapter 6, we use the same "trick" for the property* (nLhflattenll)*. This situation arises when we need to unify* $k < |\mathsf{GC}_i| \to \bot$ *from* (6.4) *with* $|\mathsf{GC}_i| \leq k$*. Since this amounts to using* $\mathsf{efq}^\bot$*, we rewrite the inequality in the lemma* (Lhflattenll) *to*

$$\forall_{ll,k}.\, (\forall_{i<|ll|}.\, k < |ll_i| \to \bot) \to k \cdot |ll| < |\mathsf{flatten}(ll)| \to \bot$$

  and avoid the use of $\mathsf{efq}^\bot$*.*

- *The same idea can be applied to the* IPH *presented in Chapter 5. We can take* $\tilde{\neg} r \leq f(n)$ *instead of the double negation of* $f(n) < r$ *presented in the Subsection 8.4.2 above.*

  This change affects the first recursion occurring in the extracted program - the resulting differences are presented in Table 8.1, where the identical terms are not given explicitely,

IPHcor with $\tilde{\neg}\,\tilde{\neg} f(n) < r$	IPHcor' with $\tilde{\neg} r \leq f(n)$
$M := \lambda f, r, n.$ $\mathcal{R}_1\, r\, M_{B1}\, M_{S1}$ $f\,(\lambda n_1, x.\, [if\,(f n_1 < r)\, x\,(Nil^{\mathtt{N}}, 0)])\, T_1$	$M' := \lambda f, r, n.$ $\mathcal{R}'_1\, r\, M_{B1}\, M_{S1}$ $f\,(\lambda n_1.\, (Nil^{\mathtt{N}}, 0))\, T_1$
$M_{B1} := \lambda f', z, t.\, z\, 0\, (Nil^{\mathtt{N}}, 0)$	$M'_{B1} := \lambda f', y_1, t.\, y_1\, 0$
$M_{S1} := \lambda n_1, h_1, f', z, t.\, t\, n_1$ $(\lambda m, y_2.\, h_1\,(\lambda k.\ f'\,(m \sqcup k))$ $(\lambda k, x.\, z\,(m \sqcup k)\, T_2)\, T_3)$	$M'_{S1} := \lambda n_1, h_2, f', y_1, t.\, t\, n_1$ $(\lambda m, y_2.\, h_2\,(\lambda k.\ f'\,(m \sqcup k))$ $(\lambda k.\, T'_2)\, T_3)$
$T_2 := (\mathcal{R}_2\, n_1\, M_{B2}\, M_{S2}$ $(f'\,(m \sqcup k))\, x\,(y_2\,(m \sqcup k)))$	$T'_2 := \mathcal{R}_2\,(f'(m \sqcup k))\, M_{B2}\, M_{S2}$ $n_1\,(y_1\,(m \sqcup k))\,(y_2\,(m \sqcup k))$

Table 8.1: Comparison between the programs extracted from IPHcor

The typing for the terms is $\mathcal{R}_1^{\tau_1}, \mathcal{R}_1^{'\tau_2}, x^{\mathtt{LN}\times\mathtt{N}}, z^{\gamma_1}, y^{\gamma_2}, t^{\mathtt{N}\Rightarrow\sigma\Rightarrow\mathtt{LN}\times\mathtt{N}},\ h_1^{\tau_1}, h_2^{\tau_2},$

with

$$\gamma_1 := \mathtt{N} \Rightarrow \mathtt{LN} \times \mathtt{N} \Rightarrow \mathtt{LN} \times \mathtt{N},\ \gamma_2 := \mathtt{N} \Rightarrow \mathtt{LN} \times \mathtt{N}$$
$$\sigma := \mathtt{N} \Rightarrow (\mathtt{N} \Rightarrow \mathtt{LN} \times \mathtt{N}) \Rightarrow \mathtt{LN} \times \mathtt{N}$$
$$\tau_1 := (\mathtt{N} \Rightarrow \mathtt{N}) \Rightarrow \gamma_1 \Rightarrow (\mathtt{N} \Rightarrow \sigma \Rightarrow \mathtt{LN} \times \mathtt{N}) \Rightarrow \mathtt{LN} \times \mathtt{N}$$
$$\tau_2 := (\mathtt{N} \Rightarrow \mathtt{N}) \Rightarrow \gamma_2 \Rightarrow (\mathtt{N} \Rightarrow \sigma \Rightarrow \mathtt{LN} \times \mathtt{N}) \Rightarrow \mathtt{LN} \times \mathtt{N}.$$

We observe that in the case of IPHcor' the type of $\mathcal{R}_1'$ has been simplified.

8.6 Summary of the Chapter

This chapter evolves around the desiderate of inserting double negations in a minimal way, such that the refined A-Translation is applied to an enlarged domain of formulas. For this, we need to overcome the restrictions imposed on the definite/goal formulas and the requirement of working in NA^ω. For a certain class of formulas, this can be achieved by inserting double negations. However, since $\bot$ is replaced by formulas with computational content in the process of A-translation, we propose to minimize the insertion of double negations, by identifying the so-called L-critical atoms.

Initially, the L-critical predicate symbols have been proposed in Berger et al. (2002), in order to extend the scope of A-Translation by transforming some formulas into definite/goal. In this chapter we have proven that this can be indeed achieved and have exemplified this on Stolzenberg's Principle treated in Chapter 5.

Further, we exploit the idea from Berger et al. (2002) and propose to double negate the atoms containing L-critical predicate symbols in order to obtain NA^ω-validity. We have shown that for some Π^0_2-formulas this double negation is not only necessary, but also sufficient. In order to further minimize the number of $\bot$, we propose to mix the logical falsity with the arithmetical one, in the cases when it is possible to use the complement of the relations.

Due to the role played by the logical falsety for the refined A-Translation, we can view $\bot$ as pinning down the computational content of the proof. The formulas in which $\bot$ does not occur do not contribute to the extracted programs. If they do need to contribute, then we need to insert double negations - we detect such formulas by identifying the L-critical predicate symbols in the proof. In this way, we shift the use of $\bot$ from the axioms $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$ to the L-critical atoms. This helps unravel the construction hidden in the classical proof, since the L-critical atoms provide information on how the witness for the goal formula is constructed.

We have illustrated our claims on the example of well-founded sets and on the case studies presented in Chapters 5-7. We have given a detailed exposition of the transformations necessary in each case, in order to obtain NA^{ω}-provability in view of Theorem 8.1. As we have seen in these concrete situations, it is possible to further minimize the insertion of double negations. Therefore, we are interested in finding a general frame in which we determine the premises for which the L-critical atoms do not need to be double negations. Also, we think that it should be possible to identify further situations to which Theorem 8.1 is applicable.

Chapter 9

Decoration of Proofs

The refinement of the A-Translation introduced in Chapter 3 allows for a minimization of the relevant negations $\bot$, which helps to eliminate a significant number of redundancies in programs extracted using the standard A-translation traditionally combined with the Gödel-Gentzen Translation. In this chapter we go a step further in the direction of optimizing the extracted terms, by refining the notion of computationally relevant (c.r.) formulas introduced by Definition 3.7.

Our work extends the concept of "non-computational universal quantifiers" introduced by Ulrich Berger in (Berger, 2005) to the implication connector, while proposing a method such that these *nc*-decorations are inserted in an automatic way. While $\forall$ and $\rightarrow$ suffice in order to express formulas in NA^ω (see Chapter 2 for details), an extension of the decorations to $\exists$, $\wedge$ and $\vee$ is also possible, as presented in (Ratiu and Schwichtenberg, 2010).

The operators will carry these additional markings (called decorations), based on the computational relevance of their components: the premise, in case of an implication and the variable on which it is being quantified, in case of an universal formula. As a result, the terms extracted from formulas involving non-computational (nc) operators are simplified and thus further redundancies in the programs can be eliminated.

We begin by introducing the terminology in Section 9.1 and by refining the notions of proof terms and extracted terms, to accommodate the decorated operators. Special care is taken in Section 9.2 in treating the axioms, since we start from the premise that they are a-priori marked and thus have fixed computational content. An automated decoration procedure is described in Section 9.3 and is proven to be optimal and terminating. We

conclude in Section 9.4 with an illustration of the effects of this procedure on some examples: the factorial problem and the list reversal algorithm. While the latter has been already presented in (Berger, 2005), in our work it is used to demonstrate the automatic detection of $\forall^{nc}$.

As aforementioned, the decoration algorithm can be used in a larger context than NA^{ω}. In fact, since it is applied directly in a constructive setting, it is also independent of the translation method by which classical proofs are transformed into constructive ones.

9.1 Terminlogy

In the following we give in Definition 9.1 an intuition over the terminology used in this chapter. In Table 9.1 we introduce formally the decorated versions of the operators in terms of the corresponding inference rules of the Natural Deduction calculus.

Definition 9.1 (Uniform, decorated and extended formulas/derivations)**.** .

- *In the case when the premise of an implication contributes computationally to the extracted term, we mark the implication accordingly as* $\rightarrow^{c}$ *and call it a* decorated implication

  Likewise, if the variable over which it is being universally quantified has computational relevance, then $\forall^{c}$ *is referred to as a* decorated universal quantifier*.*

  The restrictions regarding the introduction of such markings are underlined in Table 9.1.

- *Operators which do not carry any markings are called* uniform *and annotated with the superscript nc - these are the undecorated versions of operators. The nc-flag makes explicit the fact that certain components do not contribute computationally to the extracted program:*

  - *the premise, in case of an implication; in this case, we write* $\rightarrow^{nc}$*.*
  - *the quantified variable, in case of a universally quantified formula; if this is the case, then we write* $\forall^{nc}$*.*

  The nc-marking is subject to the conditions $(\circ)$ *and* $(\diamond)$ *(page 185).*

- *A formula is called* undecorated *or* uniform *when* none *of the operators occurring in this formula carries the computational marking. Thus, the only operators occurring in uniform formulas are* $\rightarrow^{nc}, \forall^{nc}$.

  Likewise, a derivation is called uniform *when all its formulas are undecorated, except for the uninstantiated formulas corresponding to the axioms and theorems.*

- *A formula* A' *is considered to be* an extension of A *(or we say that* A' extends A*), if* A' *is obtained by adding the computational marking to some operator(s) in* A.

  A' *does not need to be a proper extension of* A*: it can happen that none of the operators receives a computational marking in the process of extension.*

  A derivation K *is* an extension *of a given proof* M *if some of the formulas occurring in* M *are extended in* K.

- *Given a proof* M*, the derivation obtained by turning all its the operators into uniform ones is referred to as the* undecorated counterpart *or* proof pattern *of* M *and denoted by* M^U*. Exception from this process are the axioms and theorems, which, as we will see, have pre-assigned markings for the uninstantiated formulas.*

- *A proof* M *is said to be* decoration *of* U *when* $M^U = U$.

Remark 9.1. *Given a proof* M*, its proof pattern* M^U *is not necessarily a valid proof anymore. In practice, in most cases it is not, since for the inference rules involving axioms, the formulas do not match.*

Notation 9.1. *We write* $\overset{\smile}{\rightarrow}$ *and* $\overset{\smile}{\forall}$*, respectively, when we do not need to distinguish between the markings that the operators are carrying. We sometimes omit any notation altogether - so from this point on,*

$$\rightarrow, \overset{\smile}{\rightarrow} \in \{\rightarrow^{nc}, \rightarrow^{c}\} \text{ and } \forall, \overset{\smile}{\forall} \in \{\forall^{nc}, \forall^{c}\}.$$

In Section 3.4, the computational irrelevant formulas were associated the type ε. Using the conventions in (ε red.), we refine the notion of computational type from Definition 3.8 to include the decoration of the operators.

derivation	term
$u\colon A$	u^A
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to^{nc} B} \to^{nc+} u \quad (\circ)$	$(\lambda_{u^A} M^B)^{A \to^{nc} B} \quad (\circ)$
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B} \to^{+} u$	$(\lambda_{u^A} M^B)^{A \to B}$
$\dfrac{\mid M \qquad\qquad \mid N}{A \overset{\smile}{\to} B \qquad\qquad A \quad} \overset{\smile}{\to}^{-}$	$(M^{A \overset{\smile}{\to} B} N^A)^B$
$\dfrac{\mid M}{A} \forall^{nc+} x \quad (\diamond) \text{ and } (*)$	$(\lambda_x M^A)^{\forall^{nc}_x A} \quad (\diamond) \text{ and } (*)$
$\dfrac{\mid M}{A} \forall^{+} x \quad (*)$	$(\lambda_x M^A)^{\forall_x A}(*)$
$\dfrac{\mid M}{\overset{\breve{}}{\forall}_x A \qquad r} \overset{\breve{}}{\forall}^{-}$	$(M^{\overset{\breve{}}{\forall}_x A} r)^{A(r)}$

Table 9.1: Derivation terms for $\overset{\smile}{\to}$ and $\overset{\breve{}}{\forall}$

Definition 9.2 ((Revised) Computational type)**.** *Let P denote a decidable prime formula.We distinguish the types based on the markings of the operators and have the following typing rules:*

$$\begin{aligned}
&\tau(P) := \alpha_P, \textit{ if } P \textit{ is c.r.} && \tau(P) := \varepsilon, \textit{ if } P \textit{ is c.i.}\\
&\tau(A \to^c B) := \tau(A) \Rightarrow \tau(B), && \tau(A \to^{nc} B) := \tau(B),\\
&\tau(\forall^c_{x^\rho} A) := \rho \Rightarrow \tau(A), && \tau(\forall^{nc}_{x^\rho} A) := \tau(A).
\end{aligned}$$

with α_P some generic type variable.

We need to also extend the natural deduction inference rules, in order to incorporate the new operators $\to^{nc}$ and $\forall^{nc}$. The introduction and elimination rules are summarized in Table 9.1 and are based on the definition for *proof terms* and *extracted terms*.

In addition to the (Eigen-) variable condition $x \notin FV(\mathsf{FA}(M^A))(*)$, we require that in the rules $\to^{nc}$ and $\forall^{nc}$ the following are respectively fulfilled

$$x_u \notin FV([\![M]\!]) \tag{$\circ$}$$

$$x \notin FV([\![M]\!]). \tag{$\diamond$}$$

These conditions ensure that the premise of $\to^{nc}$ and the variable quantified by $\forall^{nc}$ have no computational contribution to the extracted term.

We extend Definition 2.9 of proof terms and Definition 3.10 of extracted terms, to make the distinction for the uniform connectors.

Definition 9.3. *We define proof terms and extracted terms simultaneously:*

Proof terms [1]*:*

$$\begin{aligned}
M, N \quad ::= \quad & \mathsf{AxT} : \mathrm{atom}(tt) \mid u^A \mid (M^{A\to B} N^A)^B \mid\\
& (\lambda u^A M^B)^{A\to^c B} \mid (\lambda u^A M^B)^{A\to^{nc} B}(\circ) \mid\\
& (\lambda x^\rho M^{A(x)})^{\forall^c_{x^\rho} A(x)}(*) \mid (\lambda x^\rho M^{A(x)})^{\forall^{nc}_{x^\rho} A(x)}(\diamond) \mid\\
& (M^{\forall_{x^\rho} A(x)} t)^{A(t)}
\end{aligned}$$

Extracted terms *from decorated inference-rules:*

$$[\![u^A]\!] := x_u^{\tau(A)}, \textit{ with } x_u^{\tau(A)} \textit{ uniquely associated to } u^A$$

[1] Although we still regard the induction axioms as part of the proof terms, we present them separately in Section 9.2, since their treatment is more elaborated

and further

$$\begin{array}{llll}
[\![(\lambda u^A M)^{A\to^c B}]\!] & := \lambda x_u^{\tau(A)}[\![M]\!], & [\![(\lambda_{u^A} M)^{A\to^{nc} B}]\!] & := [\![M]\!] \\
[\![M^{A\to^c B} N]\!] & := [\![M]\!][\![N]\!], & [\![M^{A\to^{nc} B} N]\!] & := [\![M]\!] \\
[\![(\lambda x^\rho M)^{\forall^c_x A}]\!] & := \lambda x^\rho [\![M]\!], & [\![(\lambda_{x^\rho} M)^{\forall^{nc}_x A}]\!] & := [\![M]\!] \\
[\![M^{\forall^c_x A} r]\!] & := [\![M]\!] r, & [\![M^{\forall^{nc}_x A} r]\!] & := [\![M]\!]
\end{array}$$

The notion of modified realizability needs to be also adapted, so that it allows a treatment of the uniform operators. We extend below Definition 3.9.

Definition 9.4. *Let P be a decidable prime formula, A be a formula and r a term of type $\tau(A)$, if A is c.r. and ε, if A is c.i.*

$$\begin{array}{ll}
\varepsilon\,\mathbf{mr}\,P := P \\
r\,\mathbf{mr}\,A\to^c B := \forall^{nc}_x.\ x\,\mathbf{mr}\,A\ \to^{nc}\ rx\,\mathbf{mr}\,B \\
r\,\mathbf{mr}\,A\to^{nc} B := \forall^{nc}_x.\ x\,\mathbf{mr}\,A\ \to^{nc}\ r\,\mathbf{mr}\,B \\
r\,\mathbf{mr}\,\forall^c_x A \quad := \forall^{nc}_x\ rx\,\mathbf{mr}\,A \\
r\,\mathbf{mr}\,\forall^{nc}_x A \quad := \forall^{nc}_x\ r\,\mathbf{mr}\,A
\end{array}$$

We make the following observation ((Schwichtenberg and Wainer, 2011)):

Remark 9.2. *For computationally irrelevant (c.i.) A the formulas $A\to^c B$ and $A\to^{nc} B$ are computationally equivalent, in the sense that they computationally imply each other. Likewise for $\forall_x A$ and $\forall^{nc}_x A$. Therefore, since the formula $t\,\mathbf{mr}\,A$ is c.i. we can take in the definition of realizability $\to$ and $\forall$ instead of $\to^{nc}$ and $\forall^{nc}$.*

9.2 Decoration of Axioms

The uninstantiated forms of the axioms have fixed decorations, which are determined by the associated realizers. They represent one of the parameters in our decoration algorithm. In this section we give the decorations for the axioms which are used in the examples investigated in this thesis.

The Induction Axiom. As presented in Section 2.3, we work with the following induction schemes

$$\begin{array}{l}
\mathsf{Ind}_{n,A}\colon \forall_{m^{\mathtt{N}}}.\ A(0)\to\forall_n(A(n)\to A(\mathsf{S}n))\to A(m), \\
\mathsf{Ind}_{l,A}\colon \forall_{l^{\mathtt{L}(\rho)}}.\ A(\mathsf{nil})\to\forall_{x,l'}(A(l')\to A(x::l'))\to A(l), \\
\mathsf{Ind}_{b,A}\colon \forall_{b^{\mathtt{B}}}.\ A(\mathrm{tt})\to A(\mathrm{ff})\to A(b),
\end{array}$$

The decoration of the induction axiom is chosen in view of what its expected content is.

$\mathsf{Ind}_{n,A}$ The options are:

- The fully decorated version

$$\mathsf{Ind}^{c}_{n,A}\colon \forall^{c}_{m}.\ A(0) \to^{c} \forall^{c}_{n}(A(n) \to^{c} A(\mathsf{S}n)) \to^{c} A(m),$$

  which has as associated realizer the recursion operator $\mathcal{R}^{\sigma}_{\mathtt{N}}$, as known from Chapter 2 and from Section 3.4 in the A-Translation Chapter.

- In order to extract the cases operator $\mathcal{C}^{\sigma}_{\mathtt{N}}$, we need to consider the following decoration

$$\mathsf{Ind}^{nc}_{n,A}\colon \forall^{c}_{m}.\ A(0) \to^{c} \forall^{nc}_{n}(A(n) \to^{nc} A(\mathsf{S}n)) \to^{c} A(m),$$

  in which the hypothesis plays no computational contribution in the induction step.

  Note that $\mathsf{Ind}^{nc}_{n,A}$ expresses $\mathsf{Cases}_{n,A}$.

- In order to express iteration in terms of the recursion operator, we consider the following decoration of induction, in which we mark the recursion variable as not computational:

$$\mathsf{Ind}^{c,nc}_{n,A}\colon \forall^{c}_{m}.\ A(0) \to^{c} \forall^{nc}_{n}(A(n) \to^{c} A(\mathsf{S}n)) \to^{c} A(m).$$

  The conversion rules for its associated realizer $It_{\mathtt{N}} = [\![\mathsf{Ind}^{c,nc}_{n,A}]\!]$ are:

$$It_{\mathtt{N}}(0, f, g) = f, \qquad It_{\mathtt{N}}(Sn, f, g) = g(It_{\mathtt{N}}(n, f, g)).$$

  The recursion variable n is not used in the step, as it is the case for $\mathcal{R}_{\mathtt{N}}$ given in Section 2.1. $It_{\mathtt{N}}$ is in fact the iteration operator.

$\mathsf{Ind}_{l,A}$**.** The same reasoning is valid as for $\mathsf{Ind}_{n,A}$, leading to the decorated versions $\mathsf{Ind}^{c}_{l,A}$ and $\mathsf{Ind}^{c,nc}_{l,A}$.

Ind_{bA}**.** For the boolean induction only the fully decorated version makes sense.

While in each of these induction variants the decorations of the connectors are fixed, we are flexible regarding the comprehension terms which instantiate the predicate A. However, special care needs to be taken in order to unify the decorated comprehension terms. We give in what follows the criterion for this unification.

Definition 9.5. *Let A_1, A_2 be two decorations of some formula A. We take $\mathsf{lub}(A_1, A_2)$ to be the least upper bound on the common extensions of A_1 and A_2. Equivalently, $\mathsf{lub}(A_1, A_2)$ can be thought of as the least common decorated formula extending both A_1 and A_2.*

More precisely, if an operator occurs decorated in any *of A_1 and A_2, then in $\mathsf{lub}(A_1, A_2)$ we take its decorated form; otherwise we take the uniform version of the operator.*

Example 15. *For instance,*

- $\mathsf{lub}(A \to^{nc} B \to^{nc} C, A \to^{c} B \to^{nc} C) := A \to^{c} B \to^{nc} C$.
- $\mathsf{lub}(A \to^{c} B \to^{nc} C, A \to^{nc} B \to^{c} C) := A \to^{c} B \to^{c} C$.

Instantiation of $\mathsf{Ind}^c_{n,A}$. Given

$$\forall^c_m.\ A_1(0) \to^c \forall^c_n(A_2(n) \to^c A_3(\mathsf{S}n)) \to^c A_4(m),$$

with $A_i, 1 \leq i \leq 4$ all extending some substitution formula for A, a *valid instantiation* of the axiom is

$$\mathsf{Ind}^c_{n,A'} \colon \forall^c_m.\ A'(0) \to^c \forall^c_n(A'(n) \to^c A'(\mathsf{S}n)) \to^c A'(m),$$

with $A' = \mathsf{lub}(A_1, A_2, A_3, A_4)$ given by Definition 9.5.

Generalized Induction. The general induction scheme, as defined in Chapter 7 needs full decoration:

$$\mathsf{GInd}^c_{m,x,R} : \forall^c_{m,x}.\ \forall^c_x(\forall^c_y(m(y) < m(x) \to^c R(y)) \to^c R(x)) \to^c R(x).$$

The typing is:

$$\tau(\mathsf{GInd}^c_{m,x,R}) = (\alpha \to \mathtt{N}) \to \alpha \to (\alpha \to (\alpha \to \tau) \to \tau) \to \tau$$

and associated general recursion operator $\mathcal{F}$, with

$$\mathcal{F}(m, x, G) = Gx(\lambda y.\mathsf{if}\ (m(y) < m(x))\ \mathsf{then}\ \mathcal{F}(m, y, G)\ \mathsf{else}\ \varepsilon).$$

Again, if we have R_1, R_2, R_3 extending R, then the instantiation to consider is the extension of $\forall^c_{m,x}.\ \forall^c_x(\forall^c_y(m(y) < m(x) \to^c R_1(y)) \to^c R_2(x)) \to^c R_3(x)$ in which we take $R' := \mathsf{lub}(R_1, R_2, R_3)$. Thus, the correct instantiated formula extending $\mathsf{GInd}^c_{m,x,R}$ is

$$\forall^c_{m,x}.\ \forall^c_x(\forall^c_y(m(y) < m(x) \to^c R'(y)) \to^c R'(x)) \to^c R'(x).$$

Compatibility Axiom. This uses the predicate Eq for Leibniz Equality defined in Chapter 2. Its decorated version is

$$\forall^{nc}_{x_1,x_2}.\, Eq(x_1,x_2) \to^{nc} A(x_1) \to^{c} A(x_2) \qquad (Compat_A)$$

Axioms of Choice. The decorated Axiom of Choice is:

$$\mathrm{AC}\colon \forall^c_x \tilde{\exists}^c_y A(x,y) \to^c \tilde{\exists}^c_f \forall^c_x A(x, fx)$$

and the decorated Axiom of Dependent Choice:

$$\mathrm{DC}\colon \forall^c_{n,x} \exists^c_y A(n,x,y) \to^c \forall^c_x \tilde{\exists}^c_f.\, f(0) = x \wedge^c A(n, fn, f(\mathsf{S}n))$$

Again, when we are provided with (possibly decorated) instantiations A_1, A_2 for A we take

$$A' := \mathsf{lub}(A_1, A_2)$$

as an instance in place of A_1 and A_2.

9.3 Decoration Algorithm

Definition 9.6 (Sequent)**.** *Given the derivation $M : \Gamma \vdash A$ we call the* sequent *of M the set of assumption and goal formulas, i.e.,* $\mathrm{Seq}(M) := \{\Gamma, A\}$.

Remark 9.3. *When KL is an extension of MN then, clearly, K is an extension of M and L is an extension of N.*

In order to decorate correctly and optimal, we need to make sure that the conditions $(\circ)$ and $(\diamond)$ imposed on the inference rules (see Table 9.1 on page 184) are compatible with the extension of derivations. The following lemma guarantees that this is indeed the case.

Lemma 9.1. *Let K be a derivation, possibly decorated, and K_1 an extension of K. Then $FV([\![K]\!]) \subseteq FV([\![K_1]\!])$, up to extension of the assumption formulas.*

Proof. By induction on the derivations.

Case of an Axiom $FV([\![\mathsf{Ax}]\!]) = \emptyset$ for all axioms Ax, so this case is trivial.

Case $\overset{\smile}{\to}^+$ I.e.,

$$\dfrac{\begin{array}{c}[u\colon C]\\ \mid K\\ D\end{array}}{C \overset{\smile}{\to} D} \overset{\smile}{\to}^+ u \qquad (\lambda_u K)$$

Let $\lambda_u K_1$ be an extension of $\lambda_u K$. We make a case distinction on the last rule of $\lambda_u K$.

Case $\rightarrow^{nc,+}$ In this case

$$x_{u^C} \notin FV([\![K]\!]) \tag{9.1}$$

$$[\![\lambda_u K]\!]) \overset{Def 9.3}{=} [\![K]\!] \tag{9.2}$$

We distinguish on the last rule of $\lambda_u K$.

Case $\rightarrow^{nc,+}$ $FV([\![\lambda_u K_1]\!]) \overset{Def[\![.]\!]}{=} FV([\![K_1]\!]) \overset{(IH)}{\supset} FV([\![K]\!]) \overset{(9.2)}{=} FV(\lambda_u [\![K]\!])$.

Case $\rightarrow^{c,+}$ $FV([\![\lambda_u K_1]\!]) = FV(\lambda_{x_u} [\![K_1]\!]) = FV([\![K_1]\!]) \backslash \{x_u\} \overset{(IH)}{\supset} FV([\![K]\!]) \setminus \{x_u\} \overset{(9.1)}{=} FV([\![K]\!]) \overset{(9.2)}{=} FV(\lambda_u [\![K]\!])$.

Case $\rightarrow^{c,+}$ As an extension of $\lambda_u K_1$, $\lambda_u K$ must also end with $\rightarrow^{c,+}$. We have $FV([\![\lambda_u K]\!]) \overset{Def[\![.]\!]}{=} FV(\lambda_{x_u} [\![K]\!]) \overset{Def FV}{=} FV([\![K]\!]) \backslash \{x_u\} \overset{(IH)}{\subset} FV([\![K_1]\!]) \setminus \{x_u\} \overset{Def FV}{=} FV(\lambda_{x_u} [\![K_1]\!]) \overset{Def[\![.]\!]}{=} FV([\![\lambda_u K_1]\!])$.

Case $\overset{\smile}{\rightarrow}^{-}$ Assuming that we have applied the modus -ponens rule,

$$\frac{\begin{array}{cc} \mid K & \mid L \\ C \overset{\smile}{\rightarrow} D & C \end{array}}{D} \overset{\smile}{\rightarrow}^{-} \tag{KL}$$

and that $K_1 L_1$ would be an extension of KL.

We make a case distinction on the last rule of KL.

Case $\rightarrow^{nc,-}$ In this case,

$$FV([\![KL]\!]) \overset{Def}{=} FV([\![K]\!]) \overset{(IH)}{\subset} FV([\![K_1]\!]). \tag{9.3}$$

Case $\rightarrow^{nc,-}$ **in** $K_1 L_1$ Then, $FV([\![K_1 L_1]\!]) \overset{Def}{=} FV([\![K_1]\!]) \overset{(9.3)}{\supset} FV([\![KL]\!])$.

Case $\rightarrow^{c,-}$ **in** $K_1 L_1$ We have $FV([\![K_1 L_1]\!]) \overset{Def[\![.]\!]}{=} FV([\![K_1]\!][\![L_1]\!]) \supset \overset{Def FV}{\supset} FV([\![K_1]\!]) \cup FV([\![L_1]\!]) \supset FV([\![K_1]\!]) \overset{(9.3)}{\supset} FV([\![KL]\!])$.

Case $\rightarrow^{c,-}$ Then also in $K_1 L_1$ the last $\rightarrow^{-}$ needs to be decorated. Since from Remark 9.3, K_1 and L_1 extend K and L, respectively, we can apply the (IH) component-wise. We have $FV([\![KL]\!]) \overset{Def[\![.]\!]}{=} FV([\![K]\!][\![L]\!]) \overset{Def FV}{=} FV([\![K]\!]) \cup FV([\![L]\!]) \overset{(IH)}{\subset} FV([\![K_1]\!]) \cup FV([\![L_1]\!]) = \overset{Def[\![.]\!], FV}{=} FV([\![K_1 L_1]\!])$.

Case $\breve{\forall}^+$ Assume that the last rule was an introduction rule for $\breve{\forall}$

$$\frac{\begin{array}{c}\mid K\\ C\end{array}}{\breve{\forall}_x C}\,\breve{\forall}^+_x, \qquad (\lambda_x K)$$

subject of course to the *Eigen-variable condition*. Let $\lambda_x K_1$ be an extension of $\lambda_x K$.

We distinguish on whether the last $\forall$ introduced is uniform or decorated.

Case $\forall^{nc,+}$ By definition, in this case

$$x \notin FV([\![K]\!]) \tag{9.4}$$
$$[\![\lambda_x K]\!] = [\![K]\!] \tag{9.5}$$

We investigate the last rule of $\lambda_x K_1$.

Case $\forall^{nc,+}$ $FV([\![\lambda_x K_1]\!]) \overset{Def[\![.]\!]}{=} FV([\![K_1]\!]) \overset{(IH)}{\supset} FV([\![K]\!]) = FV([\![\lambda_x K]\!])$.

Case $\forall^{c,+}$ $FV([\![\lambda_x K_1]\!]) \overset{Def[\![.]\!]}{=} FV(\lambda_x [\![K_1]\!]) \overset{Def}{=} FV([\![K_1]\!])\backslash\{x\} \overset{(IH)}{\supset}$ $FV([\![K]\!]) \setminus \{x\} \overset{9.4}{=} FV([\![K]\!]) \overset{9.5}{=} FV([\![\lambda_x K]\!])$.

Case $\forall^{c,+}$ Since $\lambda_x K_1$ extends $\lambda_x K$, its last rule must also be $\forall^{c,+}$.

We have $FV([\![\lambda_x K_1]\!]) \overset{Def[\![.]\!]}{=} FV(\lambda_x [\![K_1]\!]) \overset{Def}{=} FV([\![K_1]\!])\backslash\{x\} \overset{(IH)}{\supset}$ $FV([\![K]\!]) \setminus \{x\} \overset{Def\,FV}{=} FV(\lambda_x [\![K]\!]) \overset{Def[\![.]\!]}{=} FV([\![\lambda_x K]\!])$.

Case $\breve{\forall}^-$ The elimination rule for the universal quantifier is

$$\frac{\begin{array}{cc}\mid K & \\ \breve{\forall}_x C(x) & r\end{array}}{C(r)}\,\breve{\forall}^-. \qquad (Kr)$$

Let $K_1 r$ be its extension.

Case $K^{\forall^{nc}_x C}$ By definition of $[\![.]\!]$ and the (IH)

$$FV([\![Kr]\!]) = FV([\![K]\!]) \subset FV([\![K_1]\!]) \tag{9.6}$$

Case $K_1^{\forall^{nc}_x C'}$ Then $FV([\![K_1 r]\!]) \overset{Def[\![.]\!]}{=} FV([\![K_1]\!]) \overset{(9.6)}{\supset} FV([\![Kr]\!]$.

Case $K_1^{\forall^{c}_x C'}$ We have in this case $FV([\![K_1 r]\!]) \overset{Def[\![.]\!]}{=} FV([\![K_1]\!] r) =$ $\overset{Def\,FV}{=} FV([\![K_1]\!]) \cup FV(r) \overset{(9.6)}{\supset} FV([\![Kr]\!]$.

Case $K^{\forall^c_x C}$ Since K_1 extends K, it must be also that $K_1^{\forall_x C'}$ and we have $FV([\![Kr]\!]) \overset{Def[\![.]\!]}{=} FV([\![K]\!]) \cup FV(r) \overset{(IH)}{\subset} FV([\![K_1]\!]) \cup FV(r) \overset{Def[\![.]\!]}{=} FV([\![K_1 r]\!])$.

□

Definition 9.7 (Optimality.)**.** *Consider $M : \Gamma \Rightarrow A$ to be the undecorated counterpart of a correct proof and let an extension $\{\Phi, C\}$ for* Seq(M) *be given.*

We call an extension K_∞ of M optimal *with respect to $\{\Phi, C\}$ when it fulfills simultaneously the following two requirements*

- *K_∞ is a correct proof of an appropriate extension of $\{\Phi, C\}$*
- *K_∞ is the minimal such correct extension.*

In other words, any other extension K of M deriving $\{\Phi, C\}$ or an extension of it must be an extension of K_∞ and, accordingly, Seq(K) *must extend* Seq(K_∞)*.*

The theorem that we will state and prove below, results in an algorithm which produces an optimal decoration[2], provided that it receives the correct arguments, as enumerated in what follows.

- A correct proof, which may contain decorated quantifiers. This proof will be in the first step turned into its undecorated counterpart, i.e., all its formulas will be made uniform.
- Correct extensions of the open assumptions. This means that the operators of the uninstantiated[3] open assumptions are fixed and carry the appropriate markings (see Section 9.2 on axioms for this). Should the decoration procedure enforce markings on the fixed uniform operators, we signal a contradiction and output an error.
- The goal. It is up to the user to decide which components in the goal are computationally relevant and to mark them as such. Should one try to pass on a decorated goal which is not an extension of the initial goal, the algorithm exits with an error message. The default form of the goal is its proper uniform variant, i.e., all logical operators are marked by *nc*.

[2] "Optimal" in the sense of Definition 9.7

[3] The assumptions may contain uninstantiated predicate variables.

Theorem 9.1 (Decorations of Proofs)**.** *Let $M^U : \Gamma \vdash A$ be the undecorated counterpart (or proof pattern) of a correct proof M and $\{\Phi, C\}$ be an extension of* $\mathrm{Seq}(M)$*. If a correct extension of M for the given $\{\Phi, C\}$ exists, then*

(i) we can construct this derivation, K_∞, and extensions $\{\Phi_\infty, C_\infty\}$ of $\{\Phi, C\}$ such that $K_\infty : \Phi_\infty \vdash C_\infty$.

(ii) K_∞ obtained in (i) is the optimal extension of M.

Proof. By structural induction on the derivations. We treat the two parts of the theorem simultaneously.

Axioms/Auxiliary theorems. We assume that all uninstantiated axioms and auxiliary lemmas/theorems have been a priori decorated. The valid decorations for the axioms used in this thesis are as presented in Section 9.2. The correct instantiations of the predicate variables is by the $\mathsf{lub}()$ of the corresponding comprehension terms, as introduced by Definition 9.5. Should there be more decorated versions for one axiom/theorem, we pick the least decorated one which allows to obtain a correct proof.

Let us consider for exemplification the induction axiom for natural numbers $\mathsf{Ind}_{n,A}$. As explained in Section 9.2, the options are $\mathsf{Ind}^{c}_{n,A}$, $\mathsf{Ind}^{nc}_{n,A}$ and $\mathsf{Ind}^{c,nc}_{n,A}$. Depending on which extension is provided for the axiom, say $\mathsf{Ind}^{\smile}_{n,A}$, we need to choose among these possible decorations the one minimally extending $\mathsf{Ind}^{\smile}_{n,A}$. In case they all fit, we choose $\mathsf{Ind}^{nc}_{n,A}$, as being the version with the least number of decorated operators. Further, assuming that we are provided with the substitutions

$$Q_1(0), Q_2(n), Q_3(\mathsf{S}n) \text{ and } Q_4(m) \tag{9.7}$$

for the corresponding instances of A in Ind, where Q_1, Q_2, Q_3, Q_4 all extend A, we take

$$Q_\infty = \mathsf{lub}(Q_1, Q_2, Q_3, Q_4).$$

By the induction hypothesis, Q_∞ is an extension of each Q_i, so all subproofs deriving Q_i are correctly and optimally decorated.

Similarly, in case auxiliary lemmas are used, we have by (IH) optimal decorations for them, so we only need to adjust the instantiations of their predicate variables by taking their $\mathsf{lub}()$.

Case $u : A$ Suppose that we start from

$$\Gamma,\, u\colon A$$

and are given C extending A and Φ an extension of Γ. We need to prove the extensions for the given assumptions. No uniform operator can be decorated in the process, unless it appears in a comprehension term substituting a predicate. Thus, in case C or Φ attempt to change the fixed uniformities, we reach a contradiction and exit with an error message.

Case $\rightarrow^{nc+}$

$$\dfrac{\begin{array}{c}\Gamma,[u\colon A]\\ \quad\mid M^U\\ B\end{array}}{A \rightarrow^{nc} B}\ \rightarrow^{nc+} u \qquad (\lambda_u M^U)$$

Suppose that we are given the extensions $C \overset{\smile}{\rightarrow} D$ of $A \rightarrow^{nc} B$ and Φ of Γ.

(i) We apply the (IH) on M^U and the extensions $\{\Phi, C, D\}$ and obtain $K_\infty : \Phi_\infty, C_\infty \vdash D_\infty$, if this exists, with $\{\Phi_\infty, C_\infty, D_\infty\}$ possible extensions of the given $\{\Phi, C, D\}$.
If we were given $C \rightarrow^{nc} D$, then if $x_u \notin FV([\![K_\infty]\!])$ we keep the uniformity on the implication, thus deriving $C_\infty \rightarrow^{nc} D_\infty$ by $\rightarrow^{nc+}$.
If given $C \rightarrow^{nc} D$, but $x_u \in FV([\![K_\infty]\!])$ *or* if given $C \rightarrow^{c} D$, then we need a decorated implication and thus derive $C_\infty \rightarrow^{c} D_\infty$ by $\rightarrow^{c+}$.

(ii) *Optimality* Assume $\lambda_u K$ to be another extension of $\lambda_u M$, given $\{\Phi, C, D\}$. By the (IH) on M^U and $\{\Phi, C, D\}$, K_∞ is an optimal extension of M^U, so K must be an extension of K_∞. Thus, by Lemma 9.1, if $x_u \in FV([\![K_\infty]\!])$ then $x_u \in FV([\![K]\!])$, so when the last implication must be decorated for $\lambda_u K_\infty$, then it must be also for $\lambda_u K$. Thus, $\lambda_u K$ is an extension of $\lambda_u K_\infty$.

Case $\rightarrow^{nc,-}$ Assuming that we had an application of the modus -ponens rule

$$\dfrac{\begin{array}{cc}\Gamma, \Delta & \Delta, \Pi\\ \quad\mid M^U & \quad\mid N^U\\ A \rightarrow^{nc} B & A\end{array}}{B}\ \rightarrow^{nc-} \qquad ((M\,N)^U)$$

and that extensions $\{\Phi, \Psi, \Theta\}$ and D of $\mathrm{Seq}(M\,N)$ are given.

(i) *Step 1.* For M^U and the extensions Φ, Ψ and $A \to^{nc} D$, the (IH) provides us with an extension K_1, a valid proof of the (possibly) extended $\Phi_1, \Psi_1 \vdash C_1 \overset{\smile}{\to} D_1$

$$\dfrac{\begin{array}{cc} \boldsymbol{\Phi_1}, \boldsymbol{\Psi_1} & \Delta, \Pi \\ \quad \mid \mathbf{K_1} & \quad \mid N^U \\ \mathbf{C_1} \overset{\smile}{\to} \mathbf{D_1} & A \end{array}}{D_1} \overset{\smile}{\to}^{-}(!) \qquad \text{(Step 1)}$$

However, this is not a valid proof, unless the right premise of the rule agrees with the left and the assumptions are synchronized. If we adjust them, we need a further application of the induction hypothesis

Step 2. We are given $\{\Psi_1, \Theta\}$ and C_1 extending Δ, Π and A, respectively. By the (IH) on N^U and these extensions we obtain L_2 proving C_2, (possibly) extending C_1, from $\{\Psi_2, \Theta_2\}$

$$\dfrac{\begin{array}{cc} \Phi_1, \Psi_1 & \boldsymbol{\Psi_2}, \boldsymbol{\Theta_2} \\ \quad \mid K_1 & \quad \mid \mathbf{L_2} \\ C_1 \overset{\smile}{\to} D_1 & \mathbf{C_2} \end{array}}{D_1} \overset{\smile}{\to}^{-}(!) \qquad \text{(Step 2)}$$

Thus, we again need to adapt the left premise to $C_2 \overset{\smile}{\to} D_1$ and to synchronize the assumptions, for the implication elimination to be valid. Since K_1 is not necessarily a correct proof of $C_2 \overset{\smile}{\to} D_1$ from Φ_1, Ψ_2, we need to apply the (IH) again:

Step 3. (IH) on M^U (the undecorated counterpart of K_1) and the extensions $\{\Phi_1, \Psi_2, C_2 \overset{\smile}{\to} D_1\}$ of $\{\Gamma, \Delta, A \to^{nc} B\}$ gives an extension K_3 proving $C_3 \overset{\smile}{\to} D_3$

$$\dfrac{\begin{array}{cc} \boldsymbol{\Phi_3}, \boldsymbol{\Psi_3} & \Psi_2, \Theta_2 \\ \quad \mid \mathbf{K_3} & \quad \mid L_2 \\ \mathbf{C_3} \overset{\smile}{\to} \mathbf{D_3} & C_2 \end{array}}{D_3} \overset{\smile}{\to}^{-}(!) \qquad \text{(Step 3)}$$

This, however, brings us to the above step again, having to adapt the right premise and apply the (IH) to extend L_2.

Step 4. By the (IH) on N^U and the extensions $\{\Psi_3, \Theta_2\}$ and C_3 we obtain L_4 proving C_4 (possibly extending C_3) from $\{\Psi_4, \Theta_4\}$

$$\dfrac{\begin{array}{cc} \Phi_3, \Psi_2 & \boldsymbol{\Psi_4}, \boldsymbol{\Theta_4} \\ \quad \mid K_3 & \quad \mid \mathbf{L_4} \\ C_3 \overset{\smile}{\to} D_3 & \mathbf{C_4} \end{array}}{D_3} \overset{\smile}{\to}^{-}(!) \qquad \text{(Step 4)}$$

The process will loop until there is nothing more to change, i.e., when there is an agreement between the sequences of the two derivations. This *terminates* in the worst case when all operators are decorated.

In the case of the left premise, it is important to note that $\rightarrow^{nc}$ can turn at some point into $\rightarrow^{c}$, by the application of the (IH). In this case we need to accordingly adapt the $\rightarrow^{nc-}$ rule to $\rightarrow^{c-}$.

Thus, after a finite number of steps, we retrieve the extension $K_\infty L_\infty$ of the given MN, such that the two premises of $\overset{\smile}{\rightarrow}^{-}$ agree, i.e,

$$\dfrac{\begin{array}{cc}\Phi_\infty, \Psi_\infty & \Psi_\infty, \Theta_\infty \\ \mid K_\infty & \mid L_\infty \\ C_\infty \overset{\smile}{\rightarrow} D_\infty & C_\infty\end{array}}{D_\infty}\ \overset{\smile}{\rightarrow}^{-}$$

with D_∞ extending D and $\{\Phi_\infty, \Psi_\infty, \Theta_\infty\}$ extensions of $\{\Gamma, \Delta, \Pi\}$, respectively.

(ii) *Optimality.* Assume KL

$$\dfrac{\begin{array}{cc}\Phi_0, \Psi_0 & \Psi_0, \Theta_0 \\ \mid K & \mid L \\ C_0 \overset{\smile}{\rightarrow} D_0 & C_0\end{array}}{D_0}\ \overset{\smile}{\rightarrow}^{-}$$

extends MN for the given $\{D, \Psi, \Theta\}$. We want to show that KL and $\mathrm{Seq}(KL)$ are respective extensions of $K_\infty L_\infty$ and $\mathrm{Seq}(K_\infty L_\infty)$ constructed in (i).

We use the argument of (bounded) induction on the number of iterative steps. First, we show that at each iterative step of (i) we constructed an optimal extension. Let j be an odd[4] natural less then the number of iterations.

Base Case. We analyze the first couple of iterations in (i)

$$\dfrac{\begin{array}{cc}\Phi_1, \Psi_1 & \Psi_2, \Theta_2 \\ \mid K_1 & \mid L_2 \\ C_1 \overset{\smile}{\rightarrow} D_1 & C_2\end{array}}{D_2}\ \overset{\smile}{\rightarrow}^{-} \qquad \text{(Steps 1 and 2 from (i))}$$

[4]Each iteration on the left premise is followed by an iteration on the right premise, so by taking an odd we start the analysis from the left premise.

By the (IH) on M^U and the extensions Φ, Ψ and $A \to^{nc} D$, it follows that K must be an extension of K_1 (obtained in (Step 1)) and that $\{C_0, D_0, \Phi_0, \Psi_0\}$ extend $\{C_1, D_1, \Phi_1, \Psi_1\}$, respectively. Further, by (IH) on N^U and C_1, since L_2 constructed in (i).(Step 2) is the optimal extension of N, it follows that L must be an extension of L_2, C_0 of C_2 and Ψ_0, Θ_0 respective extensions of Ψ_2, Θ_2.
Thus, KL extends K_1L_2 and $\mathrm{Seq}(KL)$ extends $\mathrm{Seq}(K_1L_2)$.

Step Case. Assume we are at some intermediate couple of steps in the derivation from (i) and at the $(j+1)^{th}$ step we have

$$\dfrac{\begin{array}{cc} \Phi_j, \Psi_j & \Psi_{j+1}, \Theta_{j+1} \\ \mid K_j & \mid L_{j+1} \\ C_j \overset{\smile}{\to} D_j & C_{j+1} \end{array}}{D_{j+1}} \overset{\smile}{\to}^{-} \qquad \text{(Step j+1)}$$

We denote the induction hypothesis (IH_j) to distinguish it from the hypothesis of the meta-induction that we use throughout this proof. By the (IH_j)

$$\begin{array}{l} K : \Phi_0, \Psi_0 \vdash C_0 \overset{\smile}{\to} D_0 \text{ extends } K_{j-2} : \Phi_{j-2}, \Psi_{j-2} \vdash C_{j-2} \overset{\smile}{\to} D_{j-2}. \\ L : \Psi_0, \Theta_0 \vdash C_0 \qquad \text{extends } L_{j-1} : \Psi_{j-1}, \Theta_{j-1} \vdash C_{j-1}, \end{array}$$

where extension is meant component-wise.
At the j^{th} step, we thus assume that we are given the extensions Φ_{j-2}, Ψ_{j-1} and $C_{j-1} \overset{\smile}{\to} D_{j-2}$. We apply (IH) on M^U and these extensions, obtaining the extension $K_j : \Phi_j, \Psi_j \vdash C_j \overset{\smile}{\to} D_j$, which by the (IH) is *optimal*. This means that $K : \Phi_0, \Psi_0 \vdash C_0 \overset{\smile}{\to} D_0$ must be an extension of it.
Likewise, by the (IH) at the $(j+1)^{th}$ Step, given Ψ_j, Θ_{j-1} and C_j we obtain an extension $L_{j+1} : \Psi_{j+1}, \Theta_{j+1} \vdash C_{j+1}$ which is optimal, i.e., $L : \Psi_0, \Theta_0 \vdash C_0$ must be an extension of it.

Last iteration. We know from (i) that the iterations ends with some k, for which, after Step $k+1$: $\Psi_k = \Psi_{k+1}$ and $C_k = C_{k+1}$. Thus, putting together the results of induction above, it follows that K must be an extension of $K_k := K_\infty$, L must be an extension of $L_{k+1} := L_\infty$ and $\{C_0, D_0, \Phi_0, \Psi_0, \Theta_0\}$ extend respectively $\{C_k, D_k, \Phi_k, \Psi_k, \Theta_k\}$ (or, with the notation from (i), $\{C_\infty, D_\infty, \Phi_\infty, \Psi_\infty, \Theta_\infty\}$).

$\rightarrow^{nc-}$ changed to $\rightarrow^{c-}$. By Lemma 9.1, $FV([\![K_\infty]\!]) \subset FV([\![K]\!])$. Thus, since the uniformity was eliminated only when necessary and we assume that the derivation given in (ii) is a correct one, if the implication $C_j \rightarrow^{nc} D_j$ needs the decoration, then $\rightarrow$ cannot be uniform in $C_0 \rightarrow D_0$ either. This is a consequence of the above argument that $C_0 \breve{\rightarrow} D_0$ extends all of the intermediary $C_j \breve{\rightarrow} D_j$. We can therefore conclude that KL extends $K_\infty L_\infty$.

Case $(\forall^{nc})^+$ Assume that the last rule of the undecorated counterpart was

$$\dfrac{\begin{array}{c}\Gamma \\ \mid M^U \\ A\end{array}}{\forall^{nc}_x A}\,(\forall^{nc})^+ x \qquad (\lambda_x M^U)$$

This implies that the appropriate variable conditions are fulfilled, i.e.,

$$\begin{aligned} x &\notin FV(FA(M^A)) \\ x_u &\notin FV([\![M]\!]), \end{aligned}$$

the latter being only necessary when the last rule introduces a uniform (undecorated) universal quantifier in M.

Assume also that we are given an extension Φ of Γ and extension $\breve{\forall}_x C$ of $\forall^{nc}_x A$.

(i) To find $\lambda_x K_\infty$, we apply the (IH) on the derivation M^U and on the given extensions Φ, C. We first obtain $K_\infty : \Phi_\infty \vdash C_\infty$, which (possibly) extends $K : \Phi \vdash C$. With an appropriate introduction of the universal quantifier, we construct the extension of $\lambda_x M$, as follows

- if $\forall^{nc}_x C$ was given and $x \notin FV([\![K_\infty]\!])$ then, since we want a minimal extension, we apply $(\forall^{nc})^+$ and obtain $\lambda_x K_\infty : \Phi_\infty \vdash \forall^{nc}_x C_\infty$.
- if $\forall^{nc}_x C$ was given, but $x \in FV([\![K_\infty]\!])$ *or* if $\forall^{c}_x C$ was given, then the minimal possible extension is $\forall^{c}_x C_\infty$ and we need to apply $(\forall^{c})^+$, thus building the proof $\lambda_x K_\infty : \Phi_\infty \vdash \forall^{c}_x C_\infty$.

Since the extensions do not change the proof structurally, the Eigen-variable condition is met in all above cases. In addition, in the construction of $\lambda_x K_\infty$ we also ensured that $x_u \notin FV([\![M]\!])$, so by the (IH) and the definition of $\breve{\forall}^+$ this is a correct proof.

(ii) *Optimality.* We want to show that $\lambda_x K_\infty : \Phi_\infty \vdash \breve{\forall}_x C_\infty$ is the optimal extension of $\lambda_x M : \Gamma \vdash \forall^{nc}_x A$ obtained when provided with the extensions $\Phi, \breve{\forall}_x C$

Assume that $\lambda_x K : \Phi_2 \vdash \breve{\forall}_x C_2$ is another extension of the given $\{M, \Phi, \breve{\forall}_x C\}$.

By the (IH) on M^U and $\{\Phi, C\}$, K_∞ is optimal, i.e., K must be an extension of K_∞.

If $\lambda_x K$ ends with $(\forall^{nc}_x)^-$, then we are in the case when are given $\forall^{nc}_x C$ and when $x \notin FV([\![K]\!])$. By Lemma 9.1, since K_∞ extends K, $FV([\![K_\infty]\!]) \subset FV([\![K]\!])$. This means that $\lambda_x K_\infty$ must have also ended with $(\forall^{nc}_x)^+$ (we are in the first case of the above analysis), so $\lambda_x K$ must be an extension of it.

In the case when $\lambda_x K$ is obtained by $\forall^+ x$ then it clearly extends $\lambda_x K_\infty$, no matter whether the last rule in K_∞ is decorated or not.

Case $(\forall^{nc})^-$ The elimination rule for the $\forall^{nc}$ quantifier is

$$\dfrac{\begin{array}{c}\Gamma \\ \vdots \\ M^U \end{array} \qquad \begin{array}{c} \\ \\ \forall^{nc}_x A(x) \end{array} \qquad \begin{array}{c} \\ \\ r \end{array}}{A(r)}\,(\forall^{nc})^- \qquad (M^U\, r)$$

Let $\{\Phi, C(r)\}$ extending $\{\Gamma, A(r)\}$ be given. We need to find the correct derivation $K_\infty\, r$ of (the extension) $\Phi_\infty \vdash C_\infty(r)$.

(i) The (IH) on M^U, Φ and $\forall^{nc}_x C(x)$ gives $K_\infty : \Phi_\infty \vdash \breve{\forall}_x C_\infty(x)$. According to whether K_∞ proves $\forall^{nc}_x C_\infty(x)$ or $\forall^{c}_x C_\infty(x)$, we apply $(\forall^{nc})^-$ or $(\forall^{c})^-$, respectively, and obtain $K_\infty\, r$ deriving $C_\infty(r)$.

(ii) *Optimality.* Assume that $K\, r : \Phi_2 \vdash C_2(r)$ extends $M\, r$ and that $\mathrm{Seq}(K\, r)$ extends $\{\Phi, C(r)\}$. By the (IH), K_∞ is optimal, so K must extend it. By Lemma 9.1, this implies $FV([\![K_\infty]\!]) \subset FV([\![K]\!])$, so K derives $\forall^{nc}_x C_2(x)$ at most when K_∞ proves $\forall^{nc}_x C_\infty(x)$. This means that Kr extends $K_\infty r$ as well.

□

We point to the fact that the case when the decoration cannot be detected at the axiom or assumption level.

Theorem 9.2 (Termination)**.** *The decoration algorithm given in Theorem 9.1 always terminates.*

Proof. Since this property of the decoration algorithm is an immediate consequence of Theorem 9.1, we only informally sketch the proof, rather than giving a formal presentation. We need to distinguish among the axiom case and the inference steps.

In case we are at an axiom or auxiliary lemma/theorem, we might need to search in our database and choose among various decorations of the axiom, lemma or theorem that our derivation is using. Since we have limited ways of decorating a formula, there must be a finite number of decorated axioms and theorems in our database to choose from. For each of these possible decorations we have to verify whether it is suitable and by the induction hypothesis, this verification is a process which terminates.

In case we are already at the level of inference rules, the choice whether to decorate them or not depends on the situations presented in Theorem 9.1. By the (IH), decoration terminates. As we have seen in the proofs of Theorem 9.1, the only problematic case is that of $\rightarrow^{-}$-rule, but as discussed, its decoration is also a terminating process, having at most as many steps as are necessary to fully decorate the formulas above in the proof tree. By the induction hypothesis, the rest of the proof is decorated in a finite number of steps, so the decoration of the entire derivation always terminates. □

9.4 Examples

9.4.1 Decoration of Implication

We first illustrate the benefits of decoration on a simple example involving implications. Consider $A \rightarrow B \rightarrow A$ with the trivial proof $M := \lambda u_1^A, u_2^B.\, u_1$. Let U be the proof pattern corresponding to M. Clearly, the premise of the second implication (B) is not of computational relevance, so we apply the decoration algorithm and specify as extension of $\mathrm{Seq}(U)$ the formula $A \rightarrow^{nc} B \rightarrow^{nc} A$. The algorithm detects that the first implication needs to be decorated, since the abstracted assumption variable is computational. The second implication can be however left undecorated, while still having a correct proof. The derivation of $A \rightarrow B \rightarrow^{nc} A$ is constructed from M.

9.4.2 Program Transformation by Decorations

Proof Transformation for Tail Recursion

In order to optimize the programs extracted from proofs using induction, (Chiarabini, 2010) proposes a series of techniques, with the purpose of transforming the extracted programs into tail-recursive form and of using accu-

mulators. These techniques are applied at proof level and consist in using special lemma, as shown in what follows. In order to achieve the desired results (Chiarabini, 2010) marks some operators as non-computational, but this is done in a rather ad-hoc manner or by following the intuition. We show in this chapter that such a decoration can be detected in a systematic way by the algorithm that we have proposed in Theorem 9.1.

We begin by reviewing the methods suggested Chapter 7 "Tail Recursion" and Chapter 8 "Beyond Primitive Recursion" in (Chiarabini, 2010). Next, we show the effects of the decoration algorithm proposed in Theorem 9.1, when coupled with the techniques from (Chiarabini, 2010). For a better understanding of the benefits of such methods, we conclude by illustrating them on the example of the factorial algorithm.

Turning recursion into iteration. For this, one needs a form of the induction axiom in which the induction step does not use the quantified variable computationally:

$$\forall^c_{n^{\mathbb{N}}} A(0) \to^c \forall^{nc}_n (A(n) \to^c A(\mathsf{S}n)) \to^c A(n) \qquad (\mathsf{Ind}^{it})$$

In order to transform a recursion operator into its iterative variant, the proof by induction of $\forall_n A(n)$ is replaced with a proof by Ind^{it} of $\forall_n \tilde{\exists}_m (m = A(n))$

Continuations. These can be forced in the extracted term by turning the Induction axiom into a lemma, provable from

$$P_A := \forall^c_n \forall^{nc}_m (A(n) \to^c A(n+m)) \to^c A(n+m)$$

In (Chiarabini, 2010) it is shown by (Proposition 7.2.3.) using induction on n that:

$$\vdash A(0) \to^c \forall^c_n (A(n) \to^c A(\mathsf{S}n)) \to^c P_A \qquad (9.8)$$

and thus the induction becomes provable from (9.8). Based on this observation, the transformation method proposed (Chiarabini, 2010) consists in replacing the induction axiom by the lemma:

$$A(0) \to^c \forall^c_n (A(n) \to^c A(\mathsf{S}n)) \to^c \forall^c_m A(m), \qquad (\mathsf{IndLem})$$

As a result of applying this transformation, one extracts programs in a continuation-passing style from constructive proofs.

Accumulator-based tail recursion (Chiarabini, 2010) proposes to replace the formula $\forall_n A(n)$ proven by induction by the formula

$$\forall^c_{n,m}.\ A(m) \to^c A(n+m).$$

For this, we have by Proposition 7.2.8 in (Chiarabini, 2010), shown by induction on n, that:

$$\vdash \forall^c_n(A(n) \to^c A(\mathsf{S}n)) \to^c \forall^c_{n,m}.\ A(m) \to^c A(n+m) \tag{9.9}$$

With this, induction follows easily (Proposition 7.2.9 in (Chiarabini, 2010)) from (9.9). Thus, one can replace the goal $\forall_n A(n)$ by

$$\forall^c_{n,m}.\ A(m) \to^c A(n+m),$$

which is provable from the original proof. With this "trick", the recursion associated with this new version of the proof is an accumulator-based recursion.

Automatic Decoration

The "iterative" variant of induction in the first transformation method above, as proposed by (Chiarabini, 2010), has already been presented as an option for the decoration of Ind to $\mathsf{Ind}^{c,nc}$ in Section 9.2.

In what follows, we apply the decoration procedure in order to algorithmically detect the *nc*-quantifier in the second transformation method presented above. We start with the fully decorated version - or proper extension, in the sense of Definition 9.1 - of the formula:

$$\forall^c_n \forall^c_m (A(n) \to^c A(n+m)) \to^c A(n+m). \tag{P^c_A}$$

The proof by induction of the decorated version of (9.8) is

$$A0 \to^c \forall^c_n (An \to^c A(\mathsf{S}n)) \to^c P^c_A \tag{AuxLem}$$

is $M_{\mathsf{AuxLem}} := \lambda t^{A0}_B, t_S^{\forall^c_n.\ An \to A(\mathsf{S}n)}, n.\ \mathsf{Ind}_{P^c_A}\ n\ M_{Base}\ M_{Step}$, where

$$\begin{aligned}
M_{Base} &:= \lambda m, u^{A0 \to^c A(0+m)}.\ u\, t_B \\
M_{Step} &:= \lambda n, w_1^{P^c_A\ n}, m, u^{A(\mathsf{S}n) \to^c A(\mathsf{S}n+m)}. \\
&\qquad (\lambda w_2^{An \to^c A(n+\mathsf{S}m)}.\ (w_1\ \mathsf{S}m\ w_2)^{A(n+\mathsf{S}m)})^{P^c_A\ n\ \mathsf{S}m} \\
&\qquad \lambda v^{An}.\ (u\ (t_S\ n\ v))^{A(\mathsf{S}n+m)}
\end{aligned}$$

With this lemma, induction can now be proven by

$$M_{\mathsf{IndLem}} := \lambda t_B^{A0}, t_S^{\forall_n^c(An \to A(\mathsf{S}n))}, n.\ \mathsf{AuxLem}\ t_B\ t_S\ n\ 0\ \lambda v^{An}.\ v.$$

Expanding the proof of AuxLem we have

$$\begin{aligned} M_{\mathsf{IndLem}} &:= \lambda u_0^{A0}, u_1, n.\ Ind\ n\ M_B\ M_S\ 0\ \lambda u_6^{An}.\ u_6, \text{ where} \\ M_B &:= \lambda m, u_2.\ u_2\ u_0 \\ M_S &:= \lambda n, u_3, m, u_4.\ (u_3 \mathsf{S} m)\ \lambda u_5.\ u_4\,((u_1 n) u_5), \end{aligned}$$

where the assumption variables are typed as

$$u_1^{\forall_n(An \to A(\mathsf{S}n))},\ u_2^{A0 \to A(0+m)},\ u_3^{P_A^c\ n}, u_4^{A(\mathsf{S}n) \to A(\mathsf{S}n+m)} \text{ and } u_5^{An}.$$

The term extracted from M_{IndLem}. We consider the typed auxiliary terms

$$x^{\alpha},\ k^{\alpha \Rightarrow \alpha},\ h^{\mathtt{N} \Rightarrow \alpha \Rightarrow \alpha},\ g^{\mathtt{N} \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha}$$

and have the following associated computational term

$$\begin{aligned} [\![M_{\mathsf{IndLem}}]\!] = \lambda x, h, n.\ \mathcal{R}_{\mathtt{N}}^{\mathtt{N} \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha}\ n \\ (\lambda m, k.\ k\ x) \\ (\lambda n, g, m, k.\ g\ \mathsf{S}\, m\ (\lambda x_1.\ k\,(h\ n\ x_1))), \end{aligned}$$

which is in tail recursive form.

Algorithmic decoration. Before we give the results obtained after applying the decoration algorithm and the corresponding extracted terms, we present the step-wise decoration, in order to illustrate how the mechanism works. For this we give the proof of M_{IndLem} and its decorated version M^c_{IndLem} in Gentzen-style. The arguments passed to the algorithm are the initial derivation, the fully decorated variant of the induction axiom, Ind^c_{n,P_A^c}, and the decorated lemma (IndLem) replacing the induction axiom.

In a first step the proof pattern of M_{IndLem} is constructed, by turning all operators into their undecorated versions, with the two aforementioned exceptions. These decorations impose the synchronization of the formulas derived by M_B and M_S, so that the operators are matched and the $\to^-$ -rule is applied correctly. For this the proof needs the following decorations:

- in order to adjust the proof pattern to Ind^c_{n,P^c_A}:

$$\frac{\mathsf{Ind}^c_{n,\mathsf{P}_A}\ n \qquad \begin{array}{l}\mid M_B\\ P^c_A\,0\end{array} \qquad \begin{array}{l}\mid M_S\\ \forall^c_n.\ P^c_A\ n \to^c P^c_A\ \mathsf{S}n\end{array}}{P^c_A\,n}\ \to^{c,}$$

- and further, for agreement with the decorations in (IndLem):

$$\frac{\dfrac{P^c_A\,n \qquad 0}{P^c_A\,n\,0}\ \to^{nc-} \qquad \dfrac{u_5 : An}{An \to An}\ \to^{nc+} u_5}{\dfrac{\dfrac{An}{\forall^c_n An}\ \forall^{c+}n}{\mathsf{IndLem}}\ \to^{c+} u^c_0, u^c_1}\ \to^{nc-}$$

,

where u_0 and u_1 have the types $A0$ and $\forall_n(An \to A(\mathsf{S}n))$, respectively.

Note that at this point, P^c_A can still be fully undecorated. However, since $A(n)$ is used computationally in (IndLem), we need to adjust the derivation, so that it complies with the requirement ($\circ$). For this we must change the implication introduction of u_5 from $\to^+$ to $\to^{c+}$. For the same reason, since this triggers the decoration of u_2 and u_5, these assumptions gain computational relevance, so they can only be introduced by the $\to^{c+}$-rule. Consequently, $P^c_A\ n$ needs the following decorations:

$$\forall^{nc}_m(A(n) \to^c A(n+m)) \to^c A(n+m) \qquad (P^d_A).$$

At this point, we need to backtrack to the above subproofs and change P^c_A to P^d_A accordingly. In addition, the rules in the second subproof need to be adjusted. We have

$$\frac{\dfrac{P^d_A\,n \qquad 0}{P^d_A\,n\,0}\ \to^{nc-} \qquad \dfrac{u_5 : An}{An \to^c An}\ \to^{c+} u_5}{\dfrac{\dfrac{An}{\forall^c_n An}\ \forall^{c+}n}{\mathsf{IndLem}}\ \to^{c+} u^c_0, u^c_1}\ \to^{c-}$$

Since the proof patterns of M_B and M_S need to be corrected to match their conclusions $P^d_A\,0$ and $\forall^c_n.\ P^d_A\ n \to^c P^d_A\ \mathsf{S}n$, in a next step we work bottom-up on both.

The proof of M_B needs to be decorated to

$$\dfrac{\dfrac{\dfrac{u_2^c : A0 \to^c A(0+m) \qquad u_0 : A0}{A(0+m)} \to^{c-}}{(A0 \to^c A(0+m)) \to^c A(0+m)} \to^{c+} u_2^c}{P_A^d\ 0} \forall^{nc+} m$$

The proof of M_S is decorated as follows

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{u_3^c : \mathsf{P}_\mathsf{A}^\mathsf{d}\ n \qquad \mathsf{S}m}{u_3^c\ \mathsf{S}m} \forall^{nc-} \qquad | \qquad An \to^c A(\mathsf{S}n+m)}{A(\mathsf{S}n+m)} \to^{nc-}}{P_A^c\ \mathsf{S}n\ m} \to^{c+} u_4^c}{P_A^c\ \mathsf{S}n} \forall^{nc+} m}{P_A^c\ n \to^c P_A^c\ \mathsf{S}n} \to^{c+} u_3}{\forall_n^c.\ P_A^c\ n \to^c P_A^c\ \mathsf{S}n} \forall^{c+} n$$

with

$$\dfrac{\dfrac{u_4 : A(\mathsf{S}n) \to^c A(\mathsf{S}n+m) \qquad \dfrac{\dfrac{u_1^c : \forall_n^c(An \to^c A(\mathsf{S}n)) \qquad n}{u_1\ n} \forall^{c-} \qquad u_5 : An}{A(\mathsf{S}n)} \to^{c-}}{A(\mathsf{S}n+m)} \to^{c-}}{An \to^c A(\mathsf{S}n+m)} \to^{c+} u_5$$

Tracing in parallel the rules in M_B and M_S and the induction subproof using both, we can see that no further restriction is imposed when introducing m, so we can safely conclude by applying $\forall^{nc+}$. The implementation in Minlog[5] of the decoration algorithm has provided exactly this result. As a result, the decoration algorithm detects the non computational character of m.

The program extracted from M_{IndLem}. After decoration we have

$$\begin{aligned}[\![M^c_{\mathsf{IndLem}}]\!] = \lambda x, h, n.\ \mathcal{R}_{\mathtt{N}}^{(\alpha\Rightarrow\alpha)\Rightarrow\alpha}\ n \\ (\lambda k.\ k\ x) \\ (\lambda n, g', k.\ g'\,(\lambda x.\ k\,(h\ n_1\ x_1))) \\ (\lambda x_2.\ x_2),\end{aligned}$$

[5] See (Minlog)

with x^{α}, $k^{\alpha\Rightarrow\alpha}$, $h^{\mathtt{N}\Rightarrow\alpha\Rightarrow\alpha}$, and $g'^{(\alpha\Rightarrow\alpha)\Rightarrow\alpha}$.

If we now compare $[\![M^{c}_{\mathsf{IndLem}}]\!]$ with $[\![M_{\mathsf{IndLem}}]\!]$, we see that the variable m is eliminated. The type of the $\mathcal{R}$-operator and of the step function g have been changed accordingly.

Application. In order to investigate the effects of this proof transformation method, combined with the decoration algorithm, we have analyzed the proof by induction of the lemma stating the existence of the factorial function:

$$(G\ 0\ 1) \to \forall_{n,k}(G\ n\ k \to G\ \mathsf{S}n\ \mathsf{S}n \cdot k) \to \forall_n \exists_k G\ n\ k,$$

with Gnk the graph of the factorial function, reading $n! = k$.

The proof is by induction on "n", using the first premise in the base case and observing that $\mathsf{S}n \cdot k$ is the wanted k, by the second premise.

The results are summarized in Table 9.2, where the auxiliary variables have the following typing: $g^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$, $f^{\mathtt{N}\Rightarrow\mathtt{N}}$ and $g'^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}$.

As can be seen in Table 9.2, the classical proof extracts into a tail-recursive program, without any further transformations other than the A-Translation. The same program can be obtained from the constructive variant if one uses IndLem and decorates the proof, in order to eliminate m from the step case. The decoration algorithm detects in this case that the variable m has no computational contribution and hence this is removed from the extracted term. Note that using IndLem alone does not extract into an optimal program, since the step function computes unnecessarily $\mathsf{S}m$ and for this the decoration algorithm needs to be applied.

9.4.3 List reversal

The list reversal example has already been treated in (Berger, 2005) to demonstrate how $\forall^{nc}$ is used. However, whereas in the paper the analysis has been carried out by hand, (Ratiu and Schwichtenberg, 2010) gives the program obtained in a mechanized way[6], by the decoration algorithm.

We give the informal weak existence proof for list reversal and refer the reader to (Ratiu and Schwichtenberg, 2010) for its formal presentation as λ-term. We use the same notation:

- vw for the list w appended to the list v
- vx for appending the one element list x: to the list v

[6]The tests have been performed with the (Minlog) system

Proof method	*Extracted program*
Classical, by the Ind-Axiom	$\lambda n_0.\ \mathcal{R}_{\mathtt{N}}^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}}\ n_0$ $\quad(\lambda f.\ f\ 1)$ $\quad(\lambda n_1, g', f.\ g'\ (\lambda n_2.\ f(n_1 \cdot n_2 + n_2)))$ $\ (\lambda n_1.\ n_1)$
Constructive, by Ind-Axiom	$\lambda n_0.\ \mathcal{R}_{\mathtt{N}}^{\mathtt{N}}\ n_0\ 1\ \lambda n_1, n_2.\ n_1 \cdot n_2 + n_2$
Constructive, by $M_{(\mathsf{IndLem})}$	$\lambda n_0.\ \mathcal{R}_{\mathtt{N}}^{\mathtt{N}\Rightarrow(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}} n_0$ $\quad(\lambda n_1, f.\ f\ 1)$ $\quad(\lambda n_1, g, m, f.\ g\ \mathsf{S}m\ (\lambda n_2.\ f\ (n_1 \cdot n_2 + n_2)))$ $\ 0\ (\lambda n_1.\ n_1)$
Constructive, by $M^c_{(\mathsf{IndLem})}$ i.e., after decoration	$\lambda n_0.\ \mathcal{R}_{\mathtt{N}}^{(\mathtt{N}\Rightarrow\mathtt{N})\Rightarrow\mathtt{N}} n_0$ $\quad(\lambda f.\ f\ 1)$ $\quad(\lambda n_1, g', f.\ g'\ (\lambda n_2.\ f\ (n_1 \cdot n_2 + n_2)))$ $\ (\lambda n_1.\ n_1)$

Table 9.2: Factorial, with variants of proof techniques

- xv for the list obtained by writing an element x in front of a list v

We show that every list has a reverse

$$\forall v \tilde{\exists}_w R(v, w) \tag{9.10}$$

starting from the assumptions

$$\text{InitRev}\colon R(\mathsf{nil}, \mathsf{nil}), \tag{9.11}$$

$$\text{GenRev}\colon \forall v, w, x(R(v, w) \to R(vx, xw)) \tag{9.12}$$

Proof. We fix an arbitrary v and assume $u\colon \forall w \neg R(v, w)$. We are left to show $\bot$. For this, we prove that all initial segments of v are non-revertible, which contradicts (9.11).

More precisely, from u and (9.12) we prove by induction on v_2 that

$$\forall v_2 \forall v_1 (v_1 v_2 = v \to \forall w \neg R v_1 w).$$

For $v_2 = \mathsf{nil}$ this follows from our initial assumption u.
For the step case, assume $v_1(xv_2) = v$, fix w and assume further $R(v_1, w)$. We must derive a contradiction. By (9.12) we conclude that $R(v_1x, xw)$.

From the properties of the append function we have that $(v_1x)v_2 = v$. The induction for v_1x gives $\forall w \neg R(v_1x, w)$, so by taking xw for w leads to the desired contradiction. □

We now have a classical proof M of $\forall_v \tilde{\exists}_w R(v, w)$ from the clauses InitRev and GenRev, to which we can apply the refined A-Translation.

Both assumptions are irrelevant goal formulas, so no substitution of the logical falsity by F is necessary. Likewise, the kernel of the goal formula is atomic, so no application of Lemma 3.6 is required. Using the refined A-Translation we replace $\bot$ throughout by $\exists_w R(v, w)$. The goal formula $\forall_v \tilde{\exists}_w R(v, w)$ is turned into $\forall_v(\forall w(R(v, w) \to \exists_w R(v, w)) \to \exists_w R(v, w))$. Since its premise is an instance of existence introduction we obtain a derivation $M^{\exists}$ of $\forall_v \exists_w Rvw$. Moreover, since neither of the assumptions, nor any of the axioms used involves $\bot$ in the uninstantiated formulas, the correctness of the proof is not affected by the substitution.

Let $P_{RevList}$ denote the term extracted in Minlog from a formalization of the above proof. This is

$$\begin{aligned} P_{Revlist} = &\lambda v_0\ \mathcal{R}^{\mathtt{LN}\to\mathtt{LN}\to\mathtt{LN}}\ v_0 \\ &\quad (\lambda v_2, v_3.\ v_3) \\ &\quad (\lambda x, v_1, g, v_2, v_3.\ g\ (v_2 :+: x :)\ (x :: v_3)) \\ &\ (\mathsf{nil}\ \mathtt{N}) \\ &\ (\mathsf{nil}\ \mathtt{N}). \end{aligned}$$

with g a variable for binary functions on lists. To better see what the underlying algorithm looks like, let us unfold the recursion:

$$\begin{aligned} P_{RevList}\ \mathsf{nil} &= (\lambda v_2, v_3.\ v_3)\ (\mathsf{nil}\mathtt{N})\ (\mathsf{nil}\mathtt{N}) \\ P_{RevList}\ xv_1 &= (\lambda v_2, v_3.\ \mathcal{R}\ v_1\ v_2\, x\, x\, v_3)\ (\mathsf{nil}\mathtt{N})\ (\mathsf{nil}\mathtt{N}) \end{aligned}$$

In other words, the algorithm defines an auxiliary function h by

$$h(\mathsf{nil}, v_2, v_3) := v_3, \qquad h(xv_1, v_2, v_3) := h(v_1, v_2\, x, x\, v_3)$$

and gives the result by applying h to the original list and twice nil.

Notice that the parameter v_2 is not needed. However, its presence makes the algorithm quadratic rather than linear, because in each recursion step

$v_2\, x$ is computed, and the list append function is defined by recursion on its first argument. Therefore, it is desired to get rid of this superfluous parameter and in order to achieve this, we decorate the proof. It will turn out that in the proof (by induction on v_2) of the auxiliary formula $A(v_2) := \forall v_1(v_1 v_2 = v_3 \to \forall w \neg R v_1 w))$, the variable v_1 is not used computationally. Hence, in the decorated version of the proof, we can use $\forall^{nc}_{v_1}$.

Let us now apply the general method of decorating proofs. In order to better see the effects of the algorithm, we present our proof in a formal way by writing proof trees with formulas. The arguments to the decoration algorithm are the proof pattern of the correct initial derivation, the axioms and the sequent consisting of the context $R(\mathsf{nil}, \mathsf{nil})$ and $\forall^{nc}_{v,w,x}(R(v,w) \to^{nc} R(vx, xw))$ and the end formula $\forall^{nc}_v \exists^{\mathrm{l}} w R(v,w)$. $\exists^{\mathrm{l}}$ is a decorated variant of the strong existence quantifier[7], given by the rules

$$\exists^{\mathrm{l}+} : \forall^c_x(A \to^{nc} \exists^{\mathrm{l}}_x A) \qquad \exists^{\mathrm{l}-} : \exists^{\mathrm{l}}_x A \to^c \forall^c_x(A \to^{nc} B) \to^c B$$

In order to obtain a correct proof we need to extend the context, in a first step such that the rules involving the c.r. axioms match their decorated forms. Thus, we need to pay special attention how we apply the inference rules for the compatibility axiom in its decorated form ($Compat_A$) and the list induction in the variant $\mathsf{Ind}^c_{l,A}$ with

$$A(v_2) := \forall^{nc}_{v_1}(v_1 v_2 {=} v \to^{nc} \forall^c_w.\ R v_1 w \to^{nc} \exists^{\mathrm{l}}_w R(v,w)).$$

$M^{\exists}_{\mathrm{Base}}$ is the derivation[8]:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{Compat}_{\forall^c_w(Rvw \to \exists^{\mathrm{l}}_w R(v,w))} \quad v \quad v_1 \quad \begin{array}{c}[u_1 :\, v_1\, \mathsf{nil}{=}v] \\ \vdots\, N \\ v{=}v_1\end{array}}{\forall^c_w(Rvw \to \exists^{\mathrm{l}}_w R(v,w)) \to^c \forall^c_w R v_1 w \to \exists^{\mathrm{l}}_w R(v,w)} \qquad \exists^{\mathrm{l}+}(w, Rvw)}{\forall^c_w R v_1 w \to \exists^{\mathrm{l}}_w R(v,w)} \to^c -}{v_1\, \mathsf{nil} = v \to^{nc} \forall^c_w R v_1 w \to^{nc} \exists^{\mathrm{l}}_w R(v,w)} (\to^{nc})^+ u_1}{\forall^{nc}_{v_1}(v_1\, \mathsf{nil} = v \to \forall^c_w R v_1 w \to \exists^{\mathrm{l}}_w R(v,w)) \quad (= A(\mathsf{nil}))} \forall^{nc+} v_1$$

where N is a derivation involving the lemma $\forall_v(v = v\, \mathsf{nil})$ with a free assumption $u_1 :\, v_1\, \mathsf{nil}{=}v$. We have depicted the computational rule $(\to^c)^+$ and

[7] For the complete list of decorations of the strong quantifiers, see (Ratiu and Schwichtenberg, 2010).

[8] $\to$ abbreviates in this case the non-computational variant $\to^{nc}$.

pointed to the *nc*-rules which can be left as such in the proof pattern, while obtaining a valid proof. By this, we have inserted the *nc*-marking in $\forall_{v_1}$.

$M^{\exists}_{\mathrm{Step}}$ is the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[u_1\colon v_1(xv_2){=}v]}{\vert\ N_1}}{[u_0\colon A(v_2)]\qquad v_1x\qquad (v_1x)v_2{=}v}}{\forall^c_w.\ R(v_1x,w)\to\exists^{\mathrm{l}}_wR(v,w)}\qquad xw}(\forall^c)^-}{R(v_1x,xw)\to\exists^{\mathrm{l}}_{xw}R(v,xw)}\qquad [u_2\colon Rv_1w]\quad \vert\ N_2\quad R(v_1x,xw)}{\exists^{\mathrm{l}}_wR(v,w)}}{Rv_1w\to^{nc}\exists^{\mathrm{l}}_wR(v,w)}(\forall^c)^+w}{\forall^c_w(Rv_1w\to^{nc}\exists^{\mathrm{l}}_wR(v,w))}}{v_1(xv_2){=}v\to^{nc}\forall^c_w(Rv_1w\to^{nc}\exists^{\mathrm{l}}_wR(v,w))}}{\forall^{nc}_{v_1}(v_1(xv_2){=}v\to^{nc}\forall^c_w(Rv_1w\to^{nc}\exists^{\mathrm{l}}_wR(v,w)))}(\forall^{nc})^+v_1}{A(v_2)\to^cA(xv_2)}}{\forall^c_{x,v_2}(A(v_2)\to^cA(xv_2))}$$

where N_1 is a derivation involving the lemma $\forall_{v_1,x,v_2}((v_1x)v_2 = v_1(xv_2))$, with free assumption $u_1\colon v_1(xv_2){=}v$, and N_2 is one involving GenRev with the free assumption $u_2\colon Rv_1w$. We point again to the rules which need to be computational in order to match the decorated form of induction and the requirement that the goal formula must be universally quantified. Every other rule can remain non-computational, including the introduction of v_1, for which the variable condition and $(\diamond)$ are fulfilled.

The extracted term in the case of the decorated proof is

$$\begin{aligned}P^c_{RevList}\lambda v0.\ &\mathcal{R}^{\mathtt{LN}\to\mathtt{LN}}v_0\\ &(\lambda vv)\\ &(\lambda x,v,f,v_3f\,(x::v_3))\\ &(\mathsf{nil}\ \mathtt{N})\end{aligned}$$

with $\mathtt{f}$ a variable for unary functions on lists. Thus, this time, the underlying algorithm defines an auxiliary function g by

$$g(\mathsf{nil},w) := w, \qquad g(x::v,w) := g(v,x::w)$$

and gives the result by applying g to the original list and nil. In conclusion, we have obtained by machine extraction from an automated decoration of a weak existence proof the standard linear algorithm for list reversal, with its use of an accumulator.

9.5 Summary of the Chapter

In this chapter we have proposed a method by which the programs extracted from proofs are further optimized by eliminating some redundancies resulting from irrelevant terms. This work extends the concept of "non-computational universal quantifiers" introduced in (Berger, 1993) to the decoration of implication. In addition to precisely defining the notions of ^{nc} (or uniform) and ^{c} (or decorated) logical operators, we propose an algorithm, by which the these are marked in a *systematic* way according to their computational relevance. The extension to the strong operators is also possible and has been given in (Ratiu and Schwichtenberg, 2010), but in NA^{ω} these are not used, so we have limited the decoration to $\rightarrow$ and $\forall$.

In order to correctly use the decorations, we have extended the rules of natural deduction and clearly specified the conditions that need to be fulfilled, in order to mark operators as uniform (or non-computational). We have also revised the notions of computational type, proof terms and extracted terms. The modified realizability has been adapted to take into consideration such markings and associate the ε null-term to these non-computational operators. More precisely, the term-extraction is defined such that in the case of $\rightarrow^{nc}$ the empty realizer is associated to the premise and in the case of $\forall^{nc}$ the quantified variable is eliminated from the extracted term and by the ϵ-reduction the terms are simplified.

Special care is payed to the axioms - their uninstantiated versions are a-priori decorated. The formulas substituting the predicate variables in the formulations of the axioms can receive further decorations, but they need to be unified by taking the least upped bound of their decorated versions.

The algorithm that we have proposed in Section 9.3 has been proven to be correct, optimal and terminating. Its results have been illustrated on a series of examples in Section 9.4. We have pointed out to the fact that the decoration method is independent of the refined A-Translation, although it maps well to its use, as the example of the reverse list shows. However, it is possible to apply the decoration algorithm also to the constructive proofs.

Chapter 10

Concluding Remarks

10.1 Overview of this Thesis

In this thesis we have focused on program extraction from classical proofs transformed by A-Translation. The main focus was on improving the extracted terms by eliminating redundancies and for this we have proposed two methods. The first consists in using a refinement of the A-Translation and in order to use it on a larger class of formulas, we propose to insert double negations in a restrictive manner. The other technique consists in marking universal quantifiers and implications as either computationally relevant or irrelevant by a Decoration Algorithm.

Traditionally, the A-Translation method combines Friedman's trick with Gödel-Gentzen's double negation translation. This introduces however many redundancies in the extracted programs, due to the role played by the logical negation. Represented as a predicate variable, $\bot$ constitutes a place holder which is substituted in the process of A-translation by the strong existence goal, thus gaining computational relevance. In order to minimize the insertion of double negations, (Berger et al., 2002) proposes to work in a system from which $\mathsf{efq}^{\bot}$ and $\mathsf{Stab}^{\bot}$ are eliminated and this makes the double negation superfluous. We call this system NA^{ω} and introduce it in Chapter 2. Further, by restricting the classes of formulas to which the translation method is applied, it is possible to substitute selected occurrences of $\bot$ by F, where the latter is regarded as a constant in the system and thus has no computational content. Such classes of formulas, called definite/goal formulas, are presented in Chapter 3 in a fashion similar to Ishihara (2000). Since the classes introduced in the work of Ishihara are not identical to our definite/goal formulas, we have compared these concepts in Section 3.5, with

the purpose of understanding better the constraints imposed on the latter. We have concluded that all restrictions are necessary for the application of the refined A-Translation.

In order to enlarge the applicability domain of the refined A-Translation, we identify in this thesis the formulas which can be included in the definite/goal classes, by the insertion of double negations. Since we want to keep the occurrences of $\bot$ to a minimum necessary, we have singled out in Chapter 8 the so called *L-critical atoms*[1], as the only ones which need to be double negated. We have proposed to use this refined Double Negation in order to obtain NA^ω-validity and have found a class of formulas for which the double negation of the L-critical atoms enables us to eliminate $\mathsf{efq}^\bot$ and $\mathsf{Stab}^\bot$. Such formulas are NA^ω-valid, which is a requirement that needs to be fulfilled in order to apply the refined A-Translation. We propose in Chapter 8 an even further refinement: a mixed use of $\bot$ and F, in the case when we have relations R for which it is easy to find a complement $\overline{R} := R \to F$.

In Chapter 4 we have compared the specifics of the programs extracted from A-translated proofs, with respect to the programming techniques presented in (Griffin, 1990) and (Chiarabini, 2010). The former addresses the interpretation by control operators and the latter presents proof transformation techniques in order to obtain tail recursive programs. The NA^ω- proofs that we regard involve relevant goals, so they can be seen as proofs of $\bot$. We have argued that on the one hand, since the refined A-Translation preserves the structure of the proofs, the relevant formulas shown by induction extract in general in tail recursive programs. On the other hand, due to the special role played by $\bot$ in the process of A-translating the proofs, the associated programs adhere to the continuation passing style.

All these results concerning the refined A-Translation were based on the expertise gained by analyzing some relevant case studies, presented Chapters 5 - 7. The selected case studies are significant problems in their field. On one hand, by analyzing the extracted programs, we were able to validate the intuitive corelation made in Chapter 4 with respect to tail recursion and the continuation passing style. On the other hand, they have given us insight with respect to the application of the refined A-Translation and have provided also useful information needed in order to develop the aforementioned strategy of the refined double negation translation.

In the field of combinatorics, the *Infinite Pigeonhole Principle* has been studied for its various applications, for instance in connection with Ramsey's

[1]The sets L are fixed and collect the quantifier-free formulas in Π^0_2-statements that we want to show

Theorem, which generalizes it. The Infinite Pigeonhole Principle states that any infinite sequence that is colored with finitely many colors has an infinite monochromatic subsequence. We have treated in Chapter 5 a simplified version for a bicolor sequence and then extended this to the Infinite Pigeonhole Principle. We have analyzed the programs extracted from the corollaries of these properties, to which we have applied of the refined A-translation.

Another interesting result in combinatorics is the *Erdös-Szekeres Theorem*, which follows as a consequence of the Finite Pigeonhole Principle. The theorem goes further than Ramsey's Theorem, showing the existence of monotonically increasing/decreasing subsequences of a given sequence. We have transformed both the principle and the theorem by the A-Translation and have analyzed in Chapter 6 the programs extracted after applying the modified realizability. This study was of particular interest due to the situations in which the insertion of the double negation was necessary, in order to prove the translated theorems in minimal logic.

Dickson's Lemma, investigated in Chapter 7, proves that each monomial ideal has a finite basis. This is a necessary property in the Gröbner Basis Theory, since its consequence, Hilbert's Basis Theorem, is needed in order to ensure the termination of Buchberger's Algorithm. We have pointed out the correlation of the specification of interest to us with related formulations from the literature. This was of particular interest since the lemma is not a Π^0_2-property, so in order to extract a computable functional, we need to regard a corollary. The proof requires the application of the minimum principle, provable by general induction, and it was very interesting to analyze the outcome of A-translation on the proof using this classical principle.

The conclusions drawn from these examples concerning the role played by negations and the possible extension of the applicability of the refined A-Translation are presented in Chapter 8. We offer a systematic view of the "tricks" deployed in each of the chapters in order to adapt the proofs to our system NA^ω. For this, we use the concept of L-critical atoms, which are the only ones requiring double negation. We prove that for a certain class of formulas, the refined double negation of formulas is NA^ω-provable and the some formulas can be transformed in this way into definite/goal ones.

Due to the interpretation given to the logical falsity and its role for the refined A-Translation, we can view it as pin-pointing the computational content of the proof. The proof terms in which $\bot$ does not occur do not contribute to the extracted programs. If they do need to contribute, then we need to insert double negations "manually" and by this unravel the construction hidden in the classical proof. Such atoms are detected by the mechanism identifying the L-critical predicate symbols, described in Chap-

ter 8. This justifies also the elimination of $\mathsf{efq}^{\perp}$ and $\mathsf{Stab}^{\perp}$ - they do not provide explicit information on how the witness for the goal formula is constructed. By inserting double negations in order to eliminate these axioms from the constructive proofs, we shift the use of $\perp$ to the L-critical atoms, which will contribute in this way to the extracted programs.

In Chapter 9 we go further and propose an improvement on top of the refined A-Translation and classical proofs. The method consists in identifying universal quantifiers and implications as either computationally relevant or irrelevant and marking them accordingly. We call such a procedure "decoration" and propose an algorithm by which the computationally irrelevant operators are automatically detected. We have shown that this algorithm is correct, optimal and terminating. The modified realizability has been adapted accordingly to deal with the markings and eliminate from the extracted programs the terms associated to the non-computational operators. We conclude Chapter 9 by presenting the effects of this Decoration Algorithm on some case studies.

10.2 Contributions

The contributions of this dissertations, in order of their importance, are:

- an enhancement of the applicability domain of A-Translation by a restrictive and controlled insertion of double negations. We have described in Chapter 8 a method by which we detect the atoms that need to be double negated. By this refined double negation we can obtain further definite/goal formulas. For some classes of intuitionistically valid formulas it is also possible to obtain their minimal logic-valid variants by the refined double negation. By this we avoid the additional negations inserted by the Gödel-Gentzen translation. Examples for this refined negative translation are presented in Chapters 5 - 7

- the elimination of part of unnecessary computations in the extracted programs, arising from associating realizers to variables which do not contribute computationally to the final result. For this, we have proposed to mark universal quantifiers/implications according to the computational relevance of their variables/premises by the Decoration Algorithm proposed in Chapter 9

- a comparison in Chapter 3 with related work identifying restricted classes of formulas for which it is possible to construct intuitionistic

proofs from classical ones. The purpose of this analysis was to understand the role of the restrictions imposed on the definite/goal formulas

- exemplifications of the application of the refined A-Translation to proofs, by considering relevant case studies: the Infinite Pigeonhole Principle (Chapter 5), the Erdös-Szekeres Theorem (Chapter 6) and Dickson's Lemma (Chapter 7)

- observations regarding the correlation with continuation passing style and tail recursion. We have explained intuitively in Chapter 4 why the refined A-translated proofs offer such functionalities in the extracted programs without the need for further transformations and have illustrated this on the analyzed examples.

10.3 Future Lines of Research

Abstracting over repeating subterms. In Chapters 8 and 9 we have focused on methods designed to improve the extracted programs. With A-translation this is achieved by manipulating $\bot$. On top of it we proposed a Decoration Algorithm, in order to identify and mark accordingly the computationally irrelevant operators and consequently clean the extracted terms from unnecessary computations. (Trifonov, 2010) proposes another method for improving the terms extracted using Gödel's Dialectica Interpretation. Trifonov uses the "let" construct in order to avoid subterm repetition in the programs associated to classical existence proofs. The claim from (Trifonov, 2010) is that by abstracting over repeating terms using the *let*-construct, an "almost linear bound" on the size of extracted programs is achieved, since re-evaluation of equal subterms is avoided. We believe that the programs extracted using the A-translation can be also improved by some similar technique. The way in which the programs extracted from the A-translated proofs can benefit from this concept requires however further investigation.

Tail recursion; connection with general induction. As presented in Chapter 4, the A-translated induction proofs extract in general to tail recursive programs. We have illustrated this claim through the examples studied in this thesis, but have also seen a counter-example when this is not the case. Hence, a more rigorous classification of the situations when the inductive proofs result in tail recursive programs is desirable. It would be interesting to investigate whether the proof transformation method as

proposed by (Chiarabini, 2010) has effect in the cases when the proofs do not extract directly in tail recursive programs.

Further, in the case of Dickson's Lemma the minimum principle is used and this is shown by general recursion, for which no connection with tail recursion has been yet established. Therefore, it would be very interesting to investigate whether general recursion extracts also into some form of tail recursion. In order to validate the general observation, one could already benefit from having an example at hand in the case of DL.

Dickson's Lemma and related results. We have already discussed the special role played by Dickson's Lemma in the Gröbner Basis Theory, but have pointed out that the A-Translation is not powerful enough to exploit the full version of the lemma. This is due to the fact that as it is, DL is not a Π^0_2-property, in neither of the versions investigated in Chapter 7. However, even if we cannot expect a computable functional to be associated with Dickson's Lemma, one could search for Π^0_2-applications, to which the refined A-Translation can be applied. For this, we have overviewed the theorems for which it is necessary that DL holds and left it as future work to find suitable weakenings of these theorems.

Another way to exploit DL is to associate it a realizer by a different method and then use the refined A-Translation on Π^0_2-corollaries of the lemma. (Seisenberger, 2003) gives an example (Stolzenberg's Principle) in which A-translation is coupled with bar recursion, in order to provide realizers for non-definite assumptions.

It is of major interest to us to investigate the connections that can be established between Dickson's Lemma and other classical and constructive principles, as we have briefly mentioned at the end of Chapter 7. Determining the strength of the full Dickson's Lemma can help towards identifying its computational potential. The results from (Berardi, 2006) suggest that one can make a correlation with other principles, such as Weak König's Lemma or the simplified version of the Infinite Pigeonhole Principle.

The theory translated in sequent calculus. We have worked in Gentzen's Natural Deduction, as presented in Chapter 2, but it is worth to investigate whether the results from our work apply to sequent calculus. It would be interesting to analyze in this context the role of the definite/goal formulas.

Extending the definite/goal classes of formulas. As we have seen in Section 3.5 in which we compared the definite/goal classes with a very similar

approach of Ishihara (2000), it is not easy to extend the defined classes, if at all meaningful. However, (Schwichtenberg and Wainer, 2011) presents a counterexample for a formula which is not definite (see the paragraph 3.5 in Section 3.5), yet fulfills the property from Lemma 3.6. The question of finding the maximal class conforming with the restrictions imposed on the definite/goal remains therefore open.

Appendix A

Technical Details

A.1 The Erdös-Szekeres Theorem

Auxiliary Functions needed for FPH

- $\mathsf{ChangeAt}(ll^{\mathtt{L(LN)}}, i, n^{\mathtt{N}})$, which modifies at each call the list ll, by appending n to the sublist ll_i, is given by the following computational rules:

$$\begin{aligned}
\mathsf{ChangeAt}((\mathsf{nil}\ \mathtt{L}), 0, n) &= (n :) : \\
\mathsf{ChangeAt}((\mathsf{nil}\ \mathtt{L}), (\mathsf{S}i), n) &= (\mathsf{nil}\ \mathtt{N}) :: \mathsf{ChangeAt}((\mathsf{nil}\ \mathtt{L}), i, n) \\
\mathsf{ChangeAt}((l :: ll), 0, n) &= (n :: l) :: ll \\
\mathsf{ChangeAt}((l :: ll), \mathsf{S}i, n) &= l :: \mathsf{ChangeAt}(ll, i, n)
\end{aligned}$$

- $\mathsf{GroupbyCols}(c, n)$ constructs the list ll, such that each sublist ll_i contains all objects of color i, as follows:

$$\begin{aligned}
\mathsf{GroupbyCols}(c, 0) &= \mathsf{ChangeAt}((\mathsf{nil}\ \mathtt{L}), c(0), 0) \\
\mathsf{GroupbyCols}(c, \mathsf{S}i)) &= \mathsf{ChangeAt}(\mathsf{GroupbyCols}(c, i), c(\mathsf{S}i), \mathsf{S}i)
\end{aligned}$$

- The flattening of a lists of lists to a list of natural numbers is done by

$$\begin{aligned}
\mathsf{flatten}(\mathsf{nil}\ \mathtt{L}) &= (\mathsf{nil}\ \mathtt{N}) \\
\mathsf{flatten}(l :: ll) &= l : + : \mathsf{flatten}(ll)
\end{aligned}$$

Unfolding of Extracted Programs The program extracted from the Finite Pigeonhole Principle FPH is:

$$P_{\mathrm{FPH}}(\mathsf{GC}) := \lambda k, n, r, c.\, \mathcal{R}_1 \mathsf{GC}\, F_1\, G_1\, k\, (\lambda i.\, \mathsf{GC}_i),$$

with

$$\begin{aligned}
F_1 &:= \lambda k', f.\, (\mathsf{nil}^{\mathbb{N}}) \\
G_1 &:= \lambda l, ll, g, k', f.\, \mathcal{R}_2\, k'\, F_2\, G_2\, M \\
F_2 &:= \lambda i, n_0, n_1, l_1, l_2.\, [\mathsf{if}\ i\ l_2\ (\lambda n_2.\, l_1)] \\
G_2 &:= \lambda i, h, i', n_0, n_1, l_1, l_2.\, [\mathsf{if}\ i'\, l_2\, (\lambda n_2.\ \ h\ n_2\ n_0\ n_1\ l_1\ l_2)] \\
M &:= |l|\ (k' \cdot |ll|)\ |flatten(ll)|\ (f0)\ (g\, k'\, (\lambda i.\, f\, \mathsf{S}i))
\end{aligned}$$

According to the values collected in GC, we have for $\mathcal{R}_1 \mathsf{GC}\, F_1\, G_1\, k\, (\lambda i.\, \mathsf{GC}_i)$ the following situations

- $\mathsf{GC} = \mathsf{nil}$

$$\begin{aligned}
P_{\mathrm{FPH}}\, k\, n\, r\, c &= \mathcal{R}_1 \mathsf{nil}\, F_1\, G_1\, k\, (\lambda i.\, \mathsf{GC}_i) \\
&= F_1\, k\, (\lambda i.\, \mathsf{GC}_i) \\
&= \lambda k', f.\, (\mathsf{nil}^{\mathbb{N}})\, k\, (\lambda i.\, \mathsf{GC}_i) = (\mathsf{nil}^{\mathbb{N}})
\end{aligned}$$

- $\mathsf{GC} = (l :: ll)$

$$\begin{aligned}
P_{\mathrm{FPH}}\, k\, n\, r\, c &= G_1\, l\, ll\, (\mathcal{R}_1\, ll\, F_1\, G_1)\, k\, (\lambda i.\, \mathsf{GC}_i) \\
&= (\lambda l, ll, g, k', f.\, \mathcal{R}_2\, k'\, F_2\, G_2\, M)\, l\, ll\, (\mathcal{R}_1\, ll\, F_1\, G_1)\, k\, (\lambda i.\, \mathsf{GC}_i) \\
&= \mathcal{R}_2\, k\, F_2\, G_2\, M\, \mathsf{S}
\end{aligned}$$

where S is the substitution

$$[l := l, ll := ll, g := \mathcal{R}_1\, ll\, F_1\, G_1, k' := k, f := \lambda i.\, \mathsf{GC}_i.]$$

and we have the following two cases,

$$\begin{aligned}
P_{\mathrm{FPH}(\mathsf{GC})}\, 0\, n\, r\, c &= \mathcal{R}_2 0\, F_2\, G_2\, M\, \mathsf{S} \\
&= F_2\, M\, \mathsf{S} \\
&= \lambda i, n_0, n_1, l_1, l_2.\, [\mathsf{if}\ i\ l_2\ (\lambda n_2.\, l_1)]\, M\, \mathsf{S} \\
&= [\mathsf{if}\ |l|\ (g\, k'\, (\lambda i.\, f\, \mathsf{S}i))\ (\lambda n_2.\, (f\, 0))]\, \mathsf{S} \\
&= [\mathsf{if}\ |l|\ ((\mathcal{R}_1\, ll\, F_1\, G_1)\, 0\, (\lambda i.\, (\lambda i.\, \mathsf{GC}_i)\, \mathsf{S}i))\ (\lambda n_2.\, (\lambda i.\, \mathsf{GC}_i)\, 0)] \\
&= [\mathsf{if}\ |l|\ ((\mathcal{R}_1\, ll\, F_1\, G_1)\, 0\, (\lambda i.\, \mathsf{GC}_{\mathsf{S}i}))\ (\lambda n_2.\, \mathsf{GC}_0)] \\
&= [\mathsf{if}\ |l|\ (P_{\mathrm{FPH}}(\mathsf{cdrGC})\, 0\, n\, r\, c)\ (\lambda n_2.\, \mathsf{GC}_0)]
\end{aligned}$$

$$
\begin{aligned}
P_{\mathrm{FPH}}\,\mathsf{S}k\,n\,r\,c &= \mathcal{R}_2\,\mathsf{S}k\,F_2\,G_2\,M\,\mathsf{S} \\
&= G_2\,k\,(\mathcal{R}_2\,k\,F_2\,G_2)\,M\,\mathsf{S} \\
&= (\lambda i, h, i', n_0, n_1, l_1, l_2.\ [\mathsf{if}\ i'\,l_2\,(\lambda n_2.\ \ h\ n_2\ n_0\ n_1\ l_1\ l_2)]) \\
&\quad\ k\,(\mathcal{R}_2\,k\,F_2\,G_2)\,M\,\mathsf{S} \\
&= (\lambda i', n_0, n_1, l_1, l_2.\ [\mathsf{if}\ i'\,l_2\,(\lambda n_2.\ \ h\ n_2\ n_0\ n_1\ l_1\ l_2)]\,M)\,\mathsf{S}' \\
&\text{where } \mathsf{S}' := \mathsf{S} \cup [i := k, h := \mathcal{R}_2\,k\,F_2\,G_2],
\end{aligned}
$$

so further

$$
\begin{aligned}
P_{\mathrm{FPH}}\,\mathsf{S}k\,n\,r\,c &= [\mathsf{if}\ |l| \\
&\qquad (g\,k'\,(\lambda i.\ f\,\mathsf{S}i)) \\
&\qquad (\lambda n_2.\ \ h\ n_2\ k' \cdot |ll|\ |flatten(ll)|\ (f\,0)\ (g\,k'\,(\lambda i.\ \mathsf{GC}_{\mathsf{S}i})))]\ \mathsf{S}' \\
&= [\mathsf{if}\ |l| \\
&\qquad ((\mathcal{R}_1\,ll\,F_1\,G_1)\,k\,(\lambda i.\ \mathsf{GC}_{\mathsf{S}i})) \\
&\qquad (\lambda n_2.\ \ (\mathcal{R}_2\,k\,F_2\,G_2)\ n_2\ k \cdot |ll|\ |flatten(ll)|\ \mathsf{GC}_0 \\
&\qquad\quad ((\mathcal{R}_1\,ll\,F_1\,G_1)\ k\ (\lambda i.\ \mathsf{GC}_{\mathsf{S}i})))] \\
&= [\mathsf{if}\ |l| \\
&\qquad P_{\mathrm{FPH}}(\mathsf{cdr}(\mathsf{GC}))\,k\,n\,r\,c \\
&\qquad (\lambda n_2.\ \ (\mathcal{R}_2\,k\,F_2\,G_2)\ n_2\ k \cdot |ll|\ |flatten(ll)|\ \mathsf{GC}_0 \\
&\qquad\quad ((\mathcal{R}_1\,ll\,F_1\,G_1)\ k\ (\lambda i.\ \mathsf{GC}_{\mathsf{S}i})))] \\
&= [\mathsf{if}\ |l| \\
&\qquad P_{\mathrm{FPH}}(\mathsf{cdr}(\mathsf{GC}))\,k\,n\,r\,c \\
&\qquad (\lambda n_2.\ \ G_1\,\mathsf{nil}^{\mathbb{N}}\,(\mathsf{cdr}\mathsf{GC})\,g\,k\,[|ll| := n_2])]
\end{aligned}
$$

For the FPH with modified $(Lem_{nPlusLt})$, we have:

$$P'_{\mathrm{FPH}}(\mathsf{GC}) := \lambda k, n, r, c.\ \mathcal{R}_1\mathsf{GC}\,F_1\,G'_1\,k\,(\lambda i.\ \mathsf{GC}_i)$$

which unfolds to

- Case $\mathsf{GC} = \mathsf{nil}$

$$P'_{\mathrm{FPH}(\mathsf{GC})}\,k\,n\,r\,c = \mathcal{R}_1\mathsf{nil}\,F_1\,G'_1\,k\,(\lambda i.\ \mathsf{GC}_i) = (\mathsf{nil}^{\mathbb{N}})$$

- Case $\mathsf{GC} = (l :: ll)$

$$
\begin{aligned}
P'_{\mathrm{FPH}(\mathsf{GC})}\,k\,n\,r\,c &= G'_1\,l\,ll\,(\mathcal{R}_1\,ll\,F_1\,G'_1)\,k\,(\lambda i.\ \mathsf{GC}_i) \\
&= (\lambda l, ll, g, k', f.\ \mathcal{R}'_2\,k' \cdot |ll|\,F'_2\,G'_2\,M')\,l\,ll\,(\mathcal{R}_1\,ll\,F_1\,G'_1)\,k\,(\lambda i.\ \mathsf{GC}_i) \\
&= \mathcal{R}'_2\,k' \cdot |ll|\,F'_2\,G'_2\,M'\,\mathsf{S}
\end{aligned}
$$

where S is the substitution

$$[l := l, ll := ll, g := \mathcal{R}_1\, ll\, F_1\, G_1', k' := k, f := \lambda i.\ \mathsf{GC}_i.]$$

Further,

$$\begin{aligned}
&\text{If } k \cdot |ll| = 0\\
P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c &= \mathcal{R}_2' k \cdot |ll|\, F_2'\, G_2'\, M'\, \mathsf{S}\\
&= F_2'\, M'\, \mathsf{S}\\
&= \lambda i, l_1, l_2.\ [\mathsf{if}\ i\ l_1\ (\lambda n_2.\ l_2)]\, M'\, \mathsf{S}\\
&= [\mathsf{if}\ |flatten\, ll|\ f0\ (\lambda n_2.\ (g\, k'\, (\lambda i.\ f\mathsf{S}i)))]\, \mathsf{S}\\
&= [\mathsf{if}\ |flatten\, ll|\ \mathsf{GC}_0\ (\lambda n_2.\ ((\mathcal{R}_1\, ll\, F_1\, G_1')\, k\, (\lambda i.\ \mathsf{GC}_{\mathsf{S}i})))]\\
&= [\mathsf{if}\ |flatten\, ll|\ \mathsf{GC}_0\ (\lambda n_2.\ P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c)]\\
&\text{If } k \cdot |ll| \neq 0\\
P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c &= \mathcal{R}_2'\, \mathsf{S}(k' \cdot |ll|)\, F_2'\, G_2'\, M'\, \mathsf{S}\\
&= G_2\, k' \cdot |ll|\, \mathcal{R}_2'\, k' \cdot |ll|\, F_2'\, G_2'\, M'\, \mathsf{S}\\
&= (\lambda i, h', i', l_1, l_2.\ [\mathsf{if}\ i'\ (h'\ 0\ l_1\ l_2)\ (\lambda n_2.\ h'\ n_2\ l_1\ l_2)])\\
&\qquad k' \cdot |ll|\, \mathcal{R}_2'\, k' |ll|\, F_2'\, G_2'\, M'\, \mathsf{S}\\
&= [\mathsf{if}\ (|flatten\, ll|)\ ((\mathcal{R}_2'\, k' \cdot |ll|\, F_2'\, G_2')\ 0\ l_1\ l_2)\\
&\qquad\qquad (\lambda n_2.\ (\mathcal{R}_2'\, k' \cdot |ll|\, F_2'\, G_2')\ n_2\ l_1\ l_2)]\, \mathsf{S}'
\end{aligned}$$

where $\mathsf{S}' := [l_1 := f0,\ l_2 = g\, k'\, (\lambda i.\ f\mathsf{S}i)] \cup \mathsf{S}$.
We let $T_2 := \mathcal{R}_2'\, k \cdot |ll|\, F_2'\, G_2'$ and we have further

$$\begin{aligned}
P'_{\mathrm{FPH}(\mathsf{GC})}\, k\, n\, r\, c &= [\mathsf{if}\ (|flatten\, ll|)\ (T_2\ 0\ l_1\ l_2)(\lambda n_2.\ T_2\ n_2\ l_1\ l_2)]\, \mathsf{S}'\\
&= [\mathsf{if}\ (|flatten\, ll|)\ (T_2\ 0\ \mathsf{GC}_0\ g\, k\, (\lambda i.\ \mathsf{GC}_{\mathsf{S}i}))\\
&\qquad\qquad (\lambda n_2.\ T_2\ n_2\ \mathsf{GC}_0\ g\, k\, (\lambda i.\ \mathsf{GC}_{\mathsf{S}i}))]\, \mathsf{S}\\
&= [\mathsf{if}\ (|flatten\, ll|)\ Case(0)(\lambda n_2.\ Case(n_2))],
\end{aligned}$$

$$\begin{aligned}
\text{where } Case(x) &:= T_2\ x\ \mathsf{GC}_0\ (\mathcal{R}_1\, ll\, F_1\, G_1')\, k\, (\lambda i.\ \mathsf{GC}_{\mathsf{S}i})\\
&= T_2\ x\ \mathsf{GC}_0\ P'_{\mathrm{FPH}(\mathsf{cdr}(\mathsf{GC}))}\, k\, n\, r\, c.
\end{aligned}$$

dnFPH The program associated to the double negated form of the Finite Pigeonhole Principle (FPH_{dn}) is:

$$\begin{aligned}
P_{\mathrm{FPH}_{dn}} := \lambda k, n, r, c.\ &\mathcal{R}_3\ n\ F_3\ G_3\ (\lambda m, l.\ [\mathsf{if}\ (c(m) < r)\ l\ (\mathsf{nil}\ \mathtt{N})])\\
&(\mathcal{R}_1\ \mathsf{GC}\ F_1'\ G_1'\ k\ (\lambda i.\ \mathsf{GC}_i)),
\end{aligned}$$

where

$$
\begin{aligned}
F_1' &:= \lambda k', f.\ (\mathsf{nil}^{\mathtt{N}}) \\
G_1' &:= \lambda l, ll, g, k', f.\ \mathcal{R}_2\ k'\ F_2'\ G_2'\ M' \\
F_2' &:= \lambda i, n_0, n_1, l_1, l_2.\ [\mathsf{if}\ i\ l_2\ (\lambda n_2.\ l_1)] \\
G_2' &:= \lambda i, h, i', n_0, n_1, l_1, l_2.\ [\mathsf{if}\ i'\ l_2\ (\lambda n_2.\ h\ n_2\ n_0\ n_1\ l_1\ l_2)] \\
M' &:= |l|\ (k' \cdot |ll|)\ |\mathsf{flatten}(ll)|\ (f\,0)\ (g\ k'\ \lambda i.\ f\ (\mathsf{S}i)) \\
F_3 &:= \lambda p.\ p\,0 \\
G_3 &:= \lambda m, C, p, l.\ C\,p\,(p\,\mathsf{S}m\,l)
\end{aligned}
$$

In order to unfold it, for $\mathcal{R}_3$, we need to distinguish on the values of n:

- $n = 0$. Then $\mathcal{R}_3\ 0\ F_3\ G_3 = F_3 = \lambda p.\ p\,0$, thus

$$
\begin{aligned}
P_{\mathrm{FPH}_{dn}}\,k\,0\,r\,c &:= (\lambda p.\ p\,0)(\lambda m, l.\ [\mathsf{if}\ (c\,m < r)\ l\ (\mathsf{nil}^{\mathtt{N}})])\,(P_{\mathrm{FPH}}\,k\,n\,r\,c) \\
&:= (\lambda m, l.\ [\mathsf{if}\ (c\,m < r)\ l\ (\mathsf{nil}^{\mathtt{N}})])\,0\,(P_{\mathrm{FPH}}\,k\,0\,r\,c) \\
&:= \mathsf{if}\ (c\,0 < r)\ \mathsf{then}\ (P_{\mathrm{FPH}}\,k\,0\,r\,c)\ \mathsf{else}\ (\mathsf{nil}^{\mathtt{N}})
\end{aligned}
$$

- $n := \mathsf{S}n$. In this case, we have $\mathcal{R}_3\ \mathsf{S}n\ F_3\ G_3 = G_3\ n\ (\mathcal{R}_3\ n\ F_3\ G_3) = (\lambda m, C, p, l.\ C\,p\,(p\,\mathsf{S}m\,l)) = \lambda p, l.\ (\mathcal{R}_3\ n\ F_3\ G_3)\,p\,(p\,\mathsf{S}n\,l)$ and therefore

$$
\begin{aligned}
P_{\mathrm{FPH}_{dn}}\,k\,\mathsf{S}n\,r\,c &:= (\lambda p, l.\ (\mathcal{R}_3\ n\ F_3\ G_3)\,p\,(p\,\mathsf{S}n\,l)) \\
&\qquad (\lambda m, l.\ [\mathsf{if}\ (c\,m < r)\ l\ (\mathsf{nil}^{\mathtt{N}})])(P_{\mathrm{FPH}}\,k\,\mathsf{S}n\,r\,c) \\
&:= (\mathcal{R}_3\ n\ F_3\ G_3)\,(\lambda m, l.\ [\mathsf{if}\ (c\,m < r)\ l\ (\mathsf{nil}^{\mathtt{N}})]) \\
&\qquad ((\lambda m, l.\ [\mathsf{if}\ (c\,m < r)\ l\ (\mathsf{nil}\ \mathtt{N})])\,\mathsf{S}n\,(P_{\mathrm{FPH}}\,k\,\mathsf{S}n\,r\,c)) \\
&:= (\mathcal{R}_3\ n\ F_3\ G_3)\,(\lambda m, l.\ [\mathsf{if}\ (c(m) < r)\ l\ (\mathsf{nil}^{\mathtt{N}})]) \\
&\qquad (\mathsf{if}\ (c\,(\mathsf{S}n) < r)\ \mathsf{then}\ (P_{\mathrm{FPH}}\,k\,\mathsf{S}n\,r\,c)\ \mathsf{else}\ (\mathsf{nil}^{\mathtt{N}}))
\end{aligned}
$$

A.2 Dickson's Lemma

Unfolding of Extracted Programs

$$
\begin{aligned}
P_{\mathrm{DL}} &:= \lambda f, g.\ \mathcal{F}_1(f, k, G_1) \\
&= \lambda f, g.\ G_1 k (\lambda y.\mathsf{if}\ (f(y) < f(k))\ \mathsf{then}\ \mathcal{F}_1(f, y, G_1)\ \mathsf{else}\ \varepsilon) \\
&= \lambda f, g.\ G_1 k\, T_1(k) \\
&= \lambda f, g.\ (\lambda n.\ \mathcal{F}_2(g, n, G_2))\, k\, T_1(k) \\
&= \lambda f, g.\ \mathcal{F}_2(g, k, G_2)\, T_1(k) \\
&= \lambda f, g.\ (G_2 k (\lambda y.\mathsf{if}\ (g(y) < g(k))\ \mathsf{then}\ \mathcal{F}_2(g, y, G_2)\ \mathsf{else}\ \varepsilon))\, T_1(k) \\
&= \lambda f, g.\ (G_2 k\, T_2(k))\, T_1(k) \\
&= \lambda f, g.\ (\lambda n_0, H, F_1.\ \mathcal{F}_3(f, \mathsf{S}n_0, G_3)) k\, T_2(k))\, T_1(k) \\
&= \lambda f, g.\ \big((\lambda n_0, H, F_1.\ G_3\,(\mathsf{S}n_0)\,(\lambda y.\mathsf{if}\ (f(y) < f(\mathsf{S}n_0))\ \mathsf{then}\ \mathcal{F}_3(f, y, G_3)\ \mathsf{else}\ \varepsilon)\big) \\
&\qquad k\ T_2(k)\big)\ T_1(k) \\
&\overset{(\circ)}{=} \lambda f, g.\ ((\lambda H, F_1.\ G_3^k\,(\mathsf{S}k)\,(\lambda y.\mathsf{if}\ (f(y) < f(\mathsf{S}k))\ \mathsf{then}\ \mathcal{F}_3(f, y, G_3^k)\ \mathsf{else}\ \varepsilon))\, T_2(k))\, T_1(k) \\
&\overset{(\circ)}{=} \lambda f, g.\ ((\lambda H, F_1.\ G_3^k\,(\mathsf{S}k)\, T_3(\mathsf{S}k))\, T_2(k))\, T_1(k) \\
&= \lambda f, g.\ \big((\lambda H, F_1.\ (\lambda n_1, F_0.\ [\mathsf{if}\ (f n_1 < f k)\ (F_1\, n_1)\ [\mathsf{if}\ (g n_1 < g k)\ (H n_1 F_0)\ (k, n_1)]]) \\
&\qquad\qquad \mathsf{S}k\, T_3(\mathsf{S}k)\big)\, T_2(k)\big)\, T_1(k) \\
&\overset{(\star)}{=} \lambda f, g.\ \big((\lambda H, F_1.\ (\lambda F_0.\ [\mathsf{if}\ (f\mathsf{S}k < f k)\ (F_1\, \mathsf{S}k)\ [\mathsf{if}\ (g\mathsf{S}k < g k)\ (H\, \mathsf{S}k\, F_0)\ (k, \mathsf{S}k)]]) \\
&\qquad\qquad T_3(\mathsf{S}k)\big)\, T_2(k)\big)\, T_1(k) \\
&= \lambda f, g.\ \big((\lambda H, F_1.\ [\mathsf{if}\ (f\mathsf{S}k < f k)\ (F_1\, \mathsf{S}k)\ [\mathsf{if}\ (g\mathsf{S}k < g k)\ (H\, \mathsf{S}k\, T_3(\mathsf{S}k))\ (k, \mathsf{S}k)]]\big) \\
&\qquad \mathrm{T}_2(k)\big)\, T_1(k) \\
&= \lambda f, g.\ (\lambda F_1.\ [\mathsf{if}\ (f\mathsf{S}k < f k)\ (F_1\, \mathsf{S}k)\ [\mathsf{if}\ (g\mathsf{S}k < g k)\ (T_2(k)\, \mathsf{S}k\, T_3(\mathsf{S}k))\ (k, \mathsf{S}k)]])\, T_1(k) \\
&= \lambda f, g.\ [\mathsf{if}\ (f\mathsf{S}k < f k)\ (T_1(k)\, \mathsf{S}k)\ [\mathsf{if}\ (g\mathsf{S}k < g k)\ (T_2(k)\, \mathsf{S}k\, T_3(\mathsf{S}k))\ (k, \mathsf{S}k)]] \\
&= \lambda f, g.\ \mathsf{if}\ (f\mathsf{S}k < f k)\ \mathsf{then}\ T_1(k)\, \mathsf{S}k \\
&\qquad\qquad \mathsf{else\ if}\ (g\mathsf{S}k < g k)\ \mathsf{then}\ T_2(k)\, \mathsf{S}k\, T_3(\mathsf{S}k).
\end{aligned}
$$

where we have used the abbreviations G_3^k for $G_3[n_0 := k]$ in $(\circ)$ and G_3 for $G_3[n_0 := n]$ in $(*)$:

$$
\begin{aligned}
T_1(k) &:= \lambda y.\mathsf{if}\ (f(y) < f(k))\ \mathsf{then}\ \mathcal{F}_1(f, y, G_1)\ \mathsf{else}\ \varepsilon && \text{(A.1)} \\
T_2(k) &:= \lambda y.\mathsf{if}\ (g(y) < g(k))\ \mathsf{then}\ \mathcal{F}_2(g, y, G_2)\ \mathsf{else}\ \varepsilon && \text{(A.2)} \\
T_3(k) &:= \lambda y.\mathsf{if}\ (f(y) < f(k))\ \mathsf{then}\ \mathcal{F}_3(f, y, G_3)\ \mathsf{else}\ \varepsilon. && \text{(A.3)}
\end{aligned}
$$

In the unfolding of P_{DL}, we have the substitution

$$F_0 := T_3(\mathsf{S}k),\ F_1 := T_2(k) \text{ and } H := T_1(k).$$

T_3 gets evaluated if $\neg(f\mathsf{S}k < fk)$ and $g\mathsf{S}k < gk$, i.e., if k is suitable for f, but not for g. Should this be the case, we have:

$$\begin{aligned}
T_P &:= \mathcal{F}_2(g, \mathsf{S}k, G_2)\, T_3(\mathsf{S}k)\\
&= G_2(\mathsf{S}k)(\lambda y.\ \mathsf{if}\ (gy < g\mathsf{S}k)\ \mathsf{then}\ \mathcal{F}_2(g, y, G_2)\ \mathsf{else}\ \varepsilon)\, T_3(\mathsf{S}k)\\
&= G_2(\mathsf{S}k)\, T_2(\mathsf{S}k)\, T_3(\mathsf{S}k)\\
&= (\lambda n_0, H, F_1.\ \mathcal{F}_3(f, \mathsf{S}n_0, G_3))\, (\mathsf{S}k)\, T_2(\mathsf{S}k)\, T_3(\mathsf{S}k)\\
&\overset{(\bullet)}{=} (\lambda H, F_1.\ \mathcal{F}_3(f, \mathsf{SS}k, G_3^{\mathsf{S}k}))\, T_2(\mathsf{S}k)\, T_3(\mathsf{S}k)\\
&\overset{(\bullet)}{=} (\lambda H, F_1.\ G_3^{\mathsf{S}k}\, \mathsf{SS}k\, (\lambda y.\ \mathsf{if}\ (fy < f(\mathsf{SS}k))\ \mathsf{then}\ \mathcal{F}_3(f, y, G_3)\ \mathsf{else}\ \varepsilon))\, T_2(\mathsf{S}k)\, T_3(\mathsf{S}k)\\
&= (\lambda H, F_1.\ G_3\, \mathsf{SS}k\, T_3(\mathsf{SS}k))\, T_2(\mathsf{S}k)\, T_3(\mathsf{S}k)\\
&= (\lambda H, F_1.\ G_3\, \mathsf{SS}k\, T_3(\mathsf{SS}k))\, T_2(\mathsf{S}k)\, T_3(\mathsf{S}k)\\
&\overset{(*)}{=} (\lambda H, F_1.\ (\lambda n_1, F_0.\ \mathsf{if}\ (fn_1 < f\mathsf{S}k)\, (F_1\, n_1)\, \mathsf{if}\ (gn_1 < g\mathsf{S}k)\, (Hn_1F_0)\, (\mathsf{S}k, n_1))\\
&\qquad\qquad \mathsf{SS}k\ T_3(\mathsf{SS}k))\, T_2(\mathsf{S}k)\, T_3(\mathsf{S}k)\\
&= \mathsf{if}\ (f\mathsf{SS}k < f\mathsf{S}k)\, (T_3(\mathsf{S}k)\, \mathsf{SS}k)\\
&\qquad\qquad \mathsf{if}\ (g\mathsf{SS}k < g\mathsf{S}k)\, (T_2(\mathsf{S}k)\ \mathsf{SS}k\, T_3(\mathsf{SS}k))\, (\mathsf{S}k, \mathsf{SS}k)
\end{aligned}$$

where in $((\bullet))$ $G_3^{\mathsf{S}k}$ abbreviates $G_3[n_0 := \mathsf{S}k]$ and in $(*)$ we have the substitution

$$F_0 := T_3(\mathsf{SS}k),\ F_1 := T_2(\mathsf{S}k) \text{ and } H := T_3(\mathsf{S}k)$$

is used, where the terms T_i's are given by (A.1) - (A.3).

Bibliography

T. Becker and V. Weispfenning. *Gröbner Bases. A Computational Approach to Commutative Algebra*. Number 141 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1993. 4

S. Berardi. Some intuitionistic equivalents of classical principles for degree 2 formulas. *Annals of Pure and Applied Logic*, 139:185–200, 2006. 7.6, 10.3

U. Berger. Uniform Heyting arithmetic. *Annals Pure Applied Logic*, 133 (1-3):125–148, 2005. 1.3, 9, 9.4.3

U. Berger. Program extraction from normalization proofs. In M. Bezem and J. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 91–106, 1993. 9.5

U. Berger and P. Oliva. Modified bar recursion. *Mathematical Structures in Computer Science*, 16(2):163–183, 2006. 7.6

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. 7.3, 7.3, 7.4.1

U. Berger, W. Buchholz, and H. Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002. (document), 1, 1.1, 1.3, 3, 3.3, 3.3.1, 3.3.1, 3.4, 3.6, 7.3, 1, 7.3, 7.4.1, 7.4.2, 7.4.2, 8.2, 8.2, 8.6, 10.1

B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)*. PhD thesis, English translation in Journal of Symbolic Computation, 2004, Mathematical Institute, University of Innsbruck, Austria, 1965. 7.5.2

L. Chiarabini. *Program Development by Proof Transformation.* PhD thesis, Ludwig Maximilian University of Munich, `http://www.mathematik.uni-muenchen.de/~chiarabi/research/PhdThesis.pdf`, 2010. 1.3, 4, 4.1.2, 4.1.2, 4, 4.1.2, 4.3, 9.4.2, 9.4.2, 9.4.2, 9.4.2, 9.4.2, 9.4.2, 10.1, 10.3

T. Coquand. A semantics of evidence for classical arithemtic. *Journal of Symbolic Logic*, Volume 60(Issue 1):325–337, Mar. 1995. 5.1

T. Coquand and H. Persson. Gröbner bases in type theory. In T. Altenkirch, W. Naraschewski, and B. Reus, editors, *Types for Proofs and Programs*, volume 1657 of *LNCS*. Springer Verlag, Berlin, Heidelberg, New York, 1999. 7

D. Cox, J. Little, and D. O'Shea. *Ideal, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics. Springer Verlag, 1992. 1, 5, 7.3, 7.3, 7.5

L. E. Dickson. Finitness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35:413–422, 1913. 7.3

A. Dragalin. New kinds of realizability. In *Abstracts of the 6th International Congress of Logic, Methodology and Philosophy of Sciences*, pages 20–24, Hannover, Germany, 1979. 3.2.1

M. Felleisen and D. P. Friedman. Control operators, the secd-machine, and the lambda-calculus. *3rd Working Conference on the Formal Description of Programming Concepts*, 1986. 2

M. Felleisen, D. P. Friedman, E. Kohlbecker, and B. Duba. Reasoning with continuations. In *Proceedings of 1st Annual IEEE Symposium on Logic in Computer Science, LICS'86*, pages 131–141, Washington, DC, 1986. IEEE Computer Society Press. 2

M. Felleisen, D. P. Friedman, E. Kohlbecker, and B. Duba. A syntactic theory of sequential control. *Theoretical Computer Science*, 52(3):205–237, 1987. 2

H. Friedman. Classically and intuitionistically provably recursive functions. In D. S. Scott and G. H. Müller, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–28. Springer Verlag, Berlin, Heidelberg, New York, 1978. 1, 3.2.1, 3.2, 3.2.2, 3.3

G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934. 2.4

G. Gentzen. Die widerspruchfreiheit der reinen zahlentheorie. *Mathematische Annalen*, 112:493565, 1936. 1, 3.1

K. Gödel. Zur intuitionistischen arithmetik und zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums*, 4:34–38, 1933. 1, 3.1

K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958. 1, 2

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. ACM Press, 1990. 1.3, 4, 4.2, 4.2.1, 4.2.1, 4.2.2, 4.3, 10.1

A. Heyting. *Intuitionism: An introduction*. North-Holland Publishing Co., Amsterdam, second revised edition edition, 1966. 2

H. Ishihara. A note on the Goedel-Gentzen translation. *Matematical Logic Quarterly*, 46:135–137, 2000. 1.3, 3, 3.3.1, 3.5, 3.12, 3.2, 3.5, 3.5, 3.6, 10.1, 10.3

F. Martin-Mateos, J. Alonso, M. Hidalgo, and J. Ruiz-Reina. A formal proof of Dickson's lemma in acl2. In *10th International Conference on Logic for Programming Artificial Intelligence and Reasoning*, 2003. 7, 2, 7.3

Minlog. The minlog proof assistant. `http://www.minlog-system.de/`. 1.2, 6, 6.1, 6.2.4, 5, 6

C. Murthy. Extracting constructive content from classical proofs. Technical Report 90–1151, Cornell University, Ithaca, NY, USA, 1990. PhD thesis. 1.1

P. Oliva. Understanding and using spector's bar recursive interpretation of classical analysis. *Proceedings of CiE'2006*, 3988:423–434, 2006. 5.2.5

P. Oliva and M. Escardo. Selection functions, bar recursion and backward induction. *Mathematical Structures in Computer Science*, 20(2):127–168, 2010. 5.2.5

C. Raffalli. Getting results from programs extracted from classical proofs. *Theoretical Computer Science*, 323:49–70, 2004. 1

D. Ratiu and H. Schwichtenberg. Decorating proofs. In S. Feferman and W. Sieg, editors, *Proofs, Categories and Computations. Essays in honor of Grigori Mints*, pages 171–188. College Publications, 2010. 1.3, 9, 9.4.3, 7, 9.5

D. Ratiu and T. Trifonov. Exploring the computational content of the infinite pigeonhole principle. *Journal of Logic and Computation*, 0:11–22, 2010. 5.2, 5.2.4, 5.2.4, 5.2.5

C. Schwarzweller. Groebner bases - theory refinement in the Mizar system. In *Lecture Notes in Artificial Intelligence*, volume 3863, pages 299–314. Springer Verlag, 2005. 7

H. Schwichtenberg. Content in proofs of list reversal. *Proceedings of the Marktoberdorf Summer School*, 2007. 1

H. Schwichtenberg and S. S. Wainer. *Proofs and Computations*. Perspectives in Mathematical Logic. Cambridge University Press, 2011. 2, 3.1, 3.3.2, 3.4.1, 3.4.2, 3.5, 7.2, 9.1, 10.3

M. Seisenberger. *On the Constructive Content of Proofs*. PhD thesis, Mathematisches Institut der Universität München, http://edoc.ub.uni-muenchen.de/1619/, 2003. PhD thesis, Supervised by Helmut Schwichtenberg. 3.3, 5.1.4, 5.3, 7.6, 10.3

T. Trifonov. Quasi-linear dialectica extraction. In *Lecture Notes in Computer Science*, volume 6158, pages 417–426, 2010. 10.3

T. Trifonov. *Analysis of Methods for Extraction from Non-constructive Proofs*. PhD thesis, Ludwig-Maximilian Universität — München, 2011. In progress. 5.2.5

A. S. Troelstra. Mathematical investigation of intuitionistic arithmetic and analysis. In A. S. Troelstra, editor, *Lecture Notes in Mathematics*, volume 344. Springer-Verlag, Berlin, Heidelberg, New York, 1973. 2

A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, 2nd edition, 2000. 2.1, 3.1, 3.4.1

A. S. Troelstra and D. Van Dalen. *Constructivism in Mathematics: An Introduction*. Studies in Logic and the Foundations of Mathematics. Elsevier Science and Technology, 1988. 3.5

C. Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, `http://www4.in.tum.de/~urbanc/Publications/Phd-Urban.ps.gz`, October 2000. 5.1, 5.1.4, 5.3

C. Urban and D. Ratiu. Classical logic is better than intuitionistic logic: A conjecture about double-negation translations. In *Proceedings of the 1st International Workshop on Classical Logic and Computation, CL&C 2006, Venice, Italy*, page 20, 2006. 5.1.4

W. Veldman and M. Bezem. Ramsey's theorem and the pigeonhole principle in intuitionistic mathematics. *Journal of the London Mathematical Society*, s2-47(2):193–211, 1993. 5.1.1