FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Automated Proof Analysis by CERES

DISSERTATION

zur Erlangung des akademischen Grades

Doktorin der Technischen Wissenschaften

eingereicht von

Anela Lolić, M.Sc.
Matrikelnummer 1055207

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof.Dr.phil. Alexander Leitsch
Zweitbetreuung: Ao.Univ.Prof.Dr.phil Matthias Baaz

Diese Dissertation haben begutachtet:

Nicolas Peltier

Peter Schroeder-Heister

Wien, 23. Jänner 2020

Anela Lolić

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Automated Proof Analysis by CERES

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktorin der Technischen Wissenschaften

by

Anela Lolić, M.Sc.
Registration Number 1055207

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof.Dr.phil. Alexander Leitsch
Second advisor: Ao.Univ.Prof.Dr.phil Matthias Baaz

The dissertation has been reviewed by:

Nicolas Peltier

Peter Schroeder-Heister

Vienna, 23rd January, 2020

Anela Lolić

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Anela Lolić, M.Sc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 23. Jänner 2020

Anela Lolić

Danksagung

Ich habe großes Glück, mich die letzte PhD Studentin von Alexander Leitsch nennen zu dürfen (mal sehen, ob das auch so bleibt) und bin dankbar für all den Rat, die Geduld und Zeit, die er damit verbrachte mir Logik, wissenschaftliches Arbeiten und vor allem Genauigkeit beizubringen. Ich bin mir sicher, dass er einer von nur wenigen ist, die ihre kostbare Zeit nach der Pensionierung damit verbringen würden.

Ich möchte meinem Zweitbetreuer Matthias Baaz für die unzähligen Gespräche danken, welche mir dabei halfen, die Welt der Wissenschaft besser zu verstehen. Obwohl meine Betreuer in ähnlichen Gebieten der Beweistheorie arbeiten, kann ich mir kein Team vorstellen, von welchem ich mehr lernen hätte können.

Während meines Studiums wurde ich aus Projekten von Matthias Baaz und später durch ein DOC-Stipendium der Österreichischen Akademie der Wissenschaften finanziert, wofür ich sehr dankbar bin.

Ich möchte mich bei allen Mitgliedern der Theory and Logic Gruppe und der Computational Logic Gruppe an der TU Wien bedanken. Meine PhD Kollegen teilten viele meiner Zweifel und unsere Gespräche machten das Leben oft einfacher. Glücklicherweise habe ich auch mit Senior-Wissenschaftlern arbeiten können, welche mich nicht nur motivierten sondern auch als lebender Beweis dafür dienten, dass man eine Dissertation tatsächlich abschließen kann. Ich denke, dass sich die nette Atmosphäre, die wir während und nach der Arbeit teilen, positiv auf unsere wissenschaftliche Arbeit auswirkt.

Natürlich möchte ich auch meiner Familie und meinen Freunden danken. Bene Hartl, ein theoretischer Physiker, sagte mir einmal, dass Logik von vielen seiner Kollegen als das “richtig schwierige Ding” angesehen wird. Was auch immer das sein mag, so hat es mich bestimmt darin bekräftigt, Logikerin zu werden. Ilma Lolić hat mich immer motiviert, anders zu Denken und andere Wege zu gehen, auch in der Wissenschaft. Unsere Diskussionen beeinflussten immer schon mein wissenschaftliches Leben. Die größte Motivation bekomme ich von meinen Eltern, Mirsad und Rajsa Lolić. Sie zeigen mir täglich, dass man im Leben alles erreichen kann. Ich bin mir absolut sicher, dass ich ohne sie all dies nie erreicht hätte und bin dankbar für ihre Unterstützung.

Acknowledgements

I consider myself lucky for being the last PhD student of Alexander Leitsch (up until now - who knows what the future holds?). I am very thankful for all the guidance, patience and time he reserved for teaching me logic, how to do science and how to be precise. There are only few people that would find time for these things on such a regular basis in their retirement and Alexander Leitsch is one of them.

I am thankful to my co-advisor Matthias Baaz for the countless discussions that helped me better understand the world of science. Although my advisors work in similar areas of proof theory, I cannot imagine a team from whom I could have learned more.

During my PhD I was financed by projects of Matthias Baaz and later by a DOC-fellowship of the Austrian Academy of Sciences. I am very grateful for that.

I would also like to thank all members of the Theory and Logic group and the Computational Logic group at TU Wien. My fellow PhD colleagues shared all my anxieties and discussions with them made life feel easier again. Fortunately I worked also with senior researchers who not only motivated me but also served as living proof that PhD theses can be finished. I believe that the nice atmosphere we share during and after work is beneficial to all our research.

Of course, I would also like to thank my family and friends. Bene Hartl, a theoretical physicist, once told me that logic is frequently considered by his colleagues as the "really hard thing". Whatever that means, it surely motivated me in deciding to become a logician. Ilma Lolić inspired me to think differently and go different ways, also in science. Discussions with her have influenced my scientific life. The biggest motivation I always get from my parents, Mirsad and Rajsa Lolić. They are a steady reminder that anything can be achieved in life. I am absolutely sure that none of this would have happened if it weren't for them and am grateful for their support.

Kurzfassung

Aus mathematischen Beweisen lässt sich explizite Information gewinnen, welche nicht im Theorem sichtbar ist. Um diese Art von Information aus Beweisen gewinnen zu können, arbeitet man mit Beweisen in Normalform, also analytischen Beweisen ohne Schnittregeln. In dieser Arbeit interessieren wir uns insbesondere für Herbrand Sequenzen. Herbrand Sequenzen sind propositional gültige Sequenzen, welche aus Instanzen des entsprechenden Theorems erstellt werden und realisieren dadurch den Satz von Herbrand im Sequenzenkalkül. Bisher wurden Herbrand Sequenzen aus Beweisen in Normalform mit einer effizienten Schnitteliminationsmethode, der Methode CERES, extrahiert.

Wir zeigen, dass Beweise nicht unbedingt in Normalform sein müssen um Herbrand Sequenzen extrahieren zu können. Unsere Methode beruht auf speziellen Eigenschaften der CERES Methode für die Beweisanalyse. Da Herbrand Sequenzen nur aus Beweisen von pränexen Endsequenzen extrahiert werden können, verallgemeinern wir die Methode um Expansionsbeweise (eine Verallgemeinerung von Herbrand Sequenzen) zu gewinnen. Die entwickelten Methoden sind implementiert und an einigen Beispielbeweisen veranschaulicht. In einem letzten Schritt verallgemeinern wir unsere Methoden um auch mit Beweisen, welche Induktionsregeln beinhalten, arbeiten zu können. Wir wählen die Darstellung von Beweisen als Beweisschemata, welche die Erstellung von sogenannten Herbrand Systemen ermöglichen, und entwickeln eine schematische CERES Methode, welche mit komplizierteren Sätzen arbeiten kann als bestehende Methoden zur Beweisanalyse induktiver Beweise. Das Ziel dieser Forschung ist eine völlig automatisierte Analyse von interessanten mathematischen Beweisen, wie zum Beispiel von Fürstenbergs Beweis über die Unendlichkeit der Primzahlen. Obwohl eine völlig automatisierte Analyse dieses Beweises noch nicht erreicht werden konnte, stellt diese Arbeit einen wichtigen Schritt in der Entwicklung dar.

Abstract

Mathematical proofs can be used to extract explicit information about mathematical objects that is not visible in the theorem itself. In general, this kind of information can be extracted only from proofs in normal form, i.e. analytic proofs without cuts. In this work we are particularly interested in the extraction of Herbrand sequents from proofs. Herbrand sequents are propositionally valid sequents composed of instances of a corresponding theorem, thus they are a realization of Herbrand's theorem in sequent calculus. So far, Herbrand sequents are extracted from proofs in normal form using the method CERES, which is a method specifically designed for efficient cut-elimination.

In this work we show that for Herbrand sequent extraction proofs need not be in normal form. This novel method is based on specific features of the CERES method and simplifies and outperforms the current CERES method for proof analysis. Moreover, as the usual method for Herbrand sequent extraction only works for proofs of prenex end-sequents, we generalize our novel method to the extraction of expansion proofs (a generalization of Herbrand sequents in a non-prenex setting). The developed methods are implemented and we demonstrate some experiments with mathematical proofs. In a last step we lift our methods to the case with induction rules. We represent induction via schemata of proofs, as proof schemata allow the extraction of so-called Herbrand systems. We define a schematic CERES method that is capable of handling several induction parameters and thus allows to analyze more complicated proofs outside the range of previously defined methods. The overall goal is to develop a fully automated analysis of interesting mathematical proofs, as for instance Fürstenberg's proof of the infinitude of primes. Though a fully automated analysis of this proof is not yet within reach, this work can be seen as a major step in the development.

Contents

CHAPTER 1

Introduction

What is a proof? This question is a quite philosophical way of starting to describe this work. However, in this thesis we will answer this question in a rather technical way. Is a proof merely a verification that a statement is true? In proof theory, where proofs are considered as objects, and more particularly in proof analysis, where properties of these objects are investigated, mathematical proofs are more than just evidence that a statement is true. In fact, proofs can contain information about mathematical objects, for example explicit information on bounds or algorithms, which are not visible in the statement itself. Therefore, analyzing mathematical proofs and extracting explicit information is a central mathematical activity. This process is also known as *proof mining*.

Typical mathematical proofs are based on structuring the arguments by auxiliary statements, so-called lemmas. While lemmas are important in defining theories and making proofs comprehensible, the information in lemma-based proofs is only implicit; to make it *explicit* the proof has to be transformed into specific normal forms. Such a normal form for formal proofs can be obtained by an elimination method for lemmas which is called *cut-elimination* and was introduced by Gerhard Gentzen in his famous paper [Gen35]. The result of cut-elimination is a purely combinatorial proof, which can be used to extract explicit mathematical information, like bounds or algorithms. Indeed, cut-elimination plays a key role in the analysis of mathematical proofs, where [Gir87] is a prominent example: Girard's analysis of Fürstenberg and Weiss' topological proof, see [FW78], of van der Waerden's theorem [VdW27]. Cut-elimination, applied to the proof of Fürstenberg and Weiss, resulted in van der Waerden's original elementary proof. Girard's analysis was carried out within mathematical meta-language by hand. Actually, his analysis is an application of cut-elimination to the proof of Fürstenberg and Weiss with the intention to obtain van der Waerden's combinatorial proof. Indeed, as any other complex proof transformation carried out by humans also cut-elimination requires a goal oriented strategy. Though such a strategy is essential in proof analysis by hand, other interesting elementary proofs will most likely not be discovered by following a fixed goal. But how

can we obtain different elementary proofs than those we were trying to obtain and which kind of information can be hidden in these proofs? The idea of *automated proof analysis* originates from such questions and indeed, a fully automated proof analysis may yield new and unexpected results. In order to automate the process of proof analysis *formal proofs* are required. In fact, is is essential to work with *correct proofs* formalized in specific calculi rather than with mathematical proofs in meta-language. The reason is obvious: although proofs in meta-language can be understood by humans very easily, computers (and algorithms) in general cannot process natural language; they deal with formal languages. The discipline of *proof checking* is concerned with the process of using software for checking formal (or better formalized) proofs for correctness. The proof checkers typically use powerful proof assistants, as *Isabelle*[1] and *Coq*[2]. Proof checkers are capable of checking also very complicated mathematical proofs, e.g. the proof of the famous four color theorem has been formally checked in Coq [Gon08]. While proof checkers verify the correctness of proofs, automated proof analysis goes one step further: here not only the construction of formal proofs is required, but also their analysis and transformation and, in the end, an interpretation of the final results by humans. Therefore, automated proof analysis consists of formalizing mathematical proofs, applying algorithmic cut-elimination and, finally, interpreting the resulting formal proof.

The cut-elimination method CERES (Cut - Elimination by RESolution) [BL00, BL06] was specifically designed for automated proof analysis of mathematical proofs. CERES substantially differs from the traditional reductive cut-elimination methods à la Gentzen, where cuts are eliminated by a stepwise reduction of cut-complexity. The *reductive methods are local* in the sense that only a small part of the whole proof is analyzed, namely the derivation corresponding to the introduction of the outermost logical operator. As a consequence, many types of redundancy in proofs are left undetected, leading to an unfortunate computational behavior. In contrast, the method CERES is based on a *structural analysis of the whole proof*, analyzing all cut-derivations simultaneously. After an application of CERES a so-called CERES normal form (in general a proof with at most atomic cuts, ACNF) is obtained. In [BL06] it was shown that CERES outperforms reductive methods of cut-elimination à la Gentzen or Tait in computational complexity. Originally CERES was developed for classical first-order logic, but the method has been extended ever since: CERES$^{\omega}$ is a CERES method without proof skolemization specifically designed for higher-order logic, see [HLW11], and schematic CERES is the CERES method for schematic first-order logic, handling a representation of induction via schemata of proofs, see [DLRW13, LPW17]. Other extensions of CERES comprise an extension to Gödel logic [BCF08] and intuitionistic logic [CLRW17, Rei14].

The last step in automated proof analysis consists in the interpretation of the result by humans. In this interpretation it is crucial to obtain *compact and meaningful information* rather than a full and typically very long formal proof. Relevant information extracted from proofs can consist of bounds for variables that are used in the proof or even programs

[1] http://isabelle.in.tum.de/index.html
[2] https://coq.inria.fr/

representing its algorithmic content (Gödel's dialectica interpretation [Göd58], see [BBS02, BBLS06] for applications to mathematical proofs). Another structure representing explicit information are mid-sequents (also called *Herbrand sequents*), a realization of Herbrand's theorem in sequent calculus. Herbrand's theorem, see [Her30, Bus94], provides one of the most fundamental insights of logic and characterizes the validity of a formula in classical first-order logic by the existence of a propositional tautology composed of instances of that formula. Its realization in sequent calculus, a Herbrand sequent, is a propositionally valid sequent composed of instances of a corresponding theorem. The crucial information lies in the *instantiations of the quantified variables* of the theorem. Also in mathematical proof analysis it is frequently more important to extract these instances (thus Herbrand sequents) than full formal proofs, which may be too large to be interpreted. So far, the most direct approach to calculate Herbrand sequents is based on Hilbert's ε-formalism, which is also the oldest framework for proof theory [HB39]. However, there are also efficient algorithms for extracting Herbrand sequents from proofs in ACNF of prenex end-sequents [HLWP08]. Though every formula (and any sequent) can be transformed to prenex form such a transformation is unnatural and may have a dramatic impact on proof complexity [BL94]. Thus it is of practical importance to extend the methods to non-prenex formulas and sequents. To this aim D. Miller [Mil87] developed the structure of *expansion trees* generalizing the derivation of end-sequents from a mid-sequent to the non-prenex case. These trees record the substitutions for quantifiers in the original formula and the formulas resulting from instantiations. The so-called deep function of an expansion tree generalizes the mid-sequent itself.

Mathematical induction is one of the most important principles in real mathematics, thus any substantial and relevant approach to proof analysis has to take into account induction. But in systems with induction rules, essential proof theoretic concepts and transformations become problematic. In particular Gentzen's method of cut-elimination fails for general induction proofs. However, there are methods for performing cut-elimination in presence of induction, but the resulting proofs do not have the subformula property and Herbrand's theorem cannot be realized [BS11, MM00]. If induction is represented via schemata of proofs [DLRW13, LPW17] schematic cut-elimination methods can be defined which allow the extraction of so-called *Herbrand systems*. A (partially) automated proof analysis using schemata was performed on Fürstenberg's proof of the infinitude of primes using topological concepts [F55̈, BHL+08]. Fürstenberg's proof was formalized as a sequence of **LK**-proofs indexed by the number of primes assumed to exist and the method CERES was applied to the entire sequence. The analysis was performed in a semi-automated way; in fact, major parts of the analysis had to be performed by hand. Nevertheless, the analysis showed that from Fürstenberg's proof Euclid's elementary proof could be obtained by applying a *formal* cut-elimination procedure. Though a fully automated analysis of this proof is not yet within reach, this example reveals the need for the development of formal schematic proof systems for handling proofs with induction. Recent developments based on schematic CERES can be considered as a first step in this direction. However, existing schematic calculi with a variant of Herbrand's theorem are either defined for a weak induction principle [LPW17], or do not guarantee a fully automated transformation and

analysis of the given proof [DLRW13].

Outline

In this work we present general methods of proof analysis as well as extensions and improvements thereof by focusing particularly on the extraction of Herbrand sequents from formalized proofs in sequent calculus.

In Chapter 2 we will introduce the basic concepts and notations needed throughout this work.

Chapter 3 focuses on Herbrand's theorem in first-order logic. We introduce the method CERES and show how Herbrand sequents and expansion proofs can be extracted. The extraction is performed after a CERES normal form, a proof with quantifier-free cuts, is constructed. Then we introduce a novel method particularly suited for the extraction of instantiations of quantified variables. This method uses specific features of the method CERES and does not construct a normal form. Indeed, the main result of this chapter is that for the extraction of Herbrand sequents and expansion proofs, the construction of a normal form is obsolete. Our results can be applied to a logic with equality rules and we show that in this setting the new method asymptotically outperforms the old one (quadratic versus cubic).

To handle proofs with induction rules, in Chapter 4 we introduce the basic concepts of an inductive setting. We introduce the inductive language, inductive derivations (proof schemata) and inductive refutations (refutation schemata). Some concepts in this chapter are already defined in [DLRW13, LPW17], however we extend the language to obtain a more general notion of proof schema and in fact, the concept of refutation schemata is novel and completely differs from the ones introduced in [DLRW13, LPW17], as it is particularly designed for a straightforward extraction of Herbrand sequents. Moreover, we obtain a more flexible formalism, which is able to handle an arbitrary number of induction *parameters*, whereas previously defined methods were only capable of handling one single parameter.

All these concepts will be needed for the construction of a schematic CERES method, which will be introduced in Chapter 5. This method is novel and based on the first-order CERES method, where the construction of a normal form becomes obsolete for the extraction of a Herbrand sequent. As this method is based on the formalisms developed in Chapter 4, it improves and extends previously developed schematic CERES methods from [DLRW13, LPW17] by handling an arbitrary number of induction parameters. Furthermore, the construction of a schematic Herbrand sequent is simplified by using the concept of refutation schemata.

Some of the methods developed in this work have already been implemented. In Chapter 6 we describe the implementations and demonstrate some experiments with mathematical proofs.

Finally, in Chapter 7 we will give an outlook on future work.

CHAPTER 2

Basic Concepts

In this section we will introduce all basic concepts that will be used throughout this work. Most of the concepts will be extended later on.

2.1 Formulas, Sequents and Sequent Calculus

We denote predicate symbols by P, Q, R, function symbols by f, g, h, constant symbols by a, b, c. Furthermore, we distinguish a countably infinite set of free variables V_f and a countably infinite set of bound variables V_b. The distinction between free and bound variables is vital to proof-transformations like cut-elimination. We use α, β for free variables and x, y, z for bound ones. Terms are defined as usual with the restriction that they may contain bound variables.

Definition 2.1.1 (semi-term, term [BL11])**.** We define a set of semi-terms inductively:

- bound and free variables are semi-terms,
- constants are semi-terms,
- if $t_1, \ldots, t_n$ are semi-terms and f is a n-place function symbol then $f(t_1, \ldots, t_n)$ is a semi-term.

Terms are semi-terms which do not contain bound variables.

Definition 2.1.2 (substitution [BL11])**.** A substitution is a mapping from $V_f \cup V_b$ to the set of semi-terms s.t. $\sigma(v) \neq v$ for only finitely many $v \in V_f \cup V_b$. If σ is a substitution with $\sigma(x_i) = t_i$ for $x_i \neq t_i$ $(1 \leq i \leq n)$ and $\sigma(v) = v$ for $v \notin \{x_1, \ldots, x_n\}$, then we denote σ by $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$.

Substitutions can be extended to terms, atoms and formulas in a homomorphic way.

Definition 2.1.3. A substitution σ is called *more general* than a substitution θ ($\sigma \leq_s \theta$) if there exists a substitution μ s.t. $\theta = \sigma\mu$.

Definition 2.1.4 (unifier [BL11])**.** Let $\mathscr{A}$ be a nonempty set of atoms and σ be a substitution. σ is a unifier of $\mathscr{A}$ if the set $\mathscr{A}\sigma$ contains only one element. σ is a most general unifier (m.g.u.) of $\mathscr{A}$ if σ is a unifier of $\mathscr{A}$ and for all unifiers λ of $\mathscr{A}$ $\sigma \leq_s \lambda$.

Note that we can also unify several sets of atoms simultaneously.

Definition 2.1.5 (simultaneous unifier [BL11])**.** Let $W = (\mathscr{A}_1, \ldots, \mathscr{A}_n)$, where the $\mathscr{A}_i$ are nonempty sets of atoms for $i = 1, \ldots, n$. A substitution σ is called a simultaneous unifier of W if σ unifies all $\mathscr{A}_i$. θ is called a most general simultaneous unifier of W if θ is a simultaneous unifier of W and $\theta \leq_s \sigma$ for all simultaneous unifiers σ of W.

Definition 2.1.6 (semi-formula, formula [BL11])**.** $\top$ and $\bot$ are formulas. If $t_1, \ldots, t_n$ are terms and P is a n-place predicate symbols then $P(t_1, \ldots, t_n)$ is an atomic formula.

- If A is a formula then $\neg A$ is a formula.
- If A, B are formulas, then $A \wedge B$, $A \vee B$ and $A \to B$ are formulas.
- If $A\{x \leftarrow a\}$ is a formula, then $\forall x A$ and $\exists x A$ are formulas.

Semi-formulas differ from formulas in containing free variables in V_b.

Definition 2.1.7 (prenex formula)**.** A formula is prenex (or in prenex form) if it is of the form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n F$, where $n \geq 0$ and $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$ and F is quantifier-free.

By $V(t)$ we denote the the set of variables (freely) occurring in a term or formula t.

Definition 2.1.8 (size of a formula)**.** Let A be a formula, then the size of A ($\|A\|_f$) is inductively defined as follows

$$\begin{array}{lcl} \|A\|_f & = & 1 \text{ if } A \text{ is an atomic formula ,} \\ \|\neg A\|_f & = & 1 + \|A\|_f, \\ \|A_1 \circ A_2\|_f & = & 1 + \|A_1\|_f + \|A_2\|_f, \circ \in \{\wedge, \vee, \to\}, \\ \|Qx.A\|_f & = & 1 + \|A\|_f, Q \in \{\forall, \exists\}. \end{array}$$

To improve legibility we write $\|F\|$ instead of $\|F\|_f$ (with the exception of cases where the precise notation is essential). The measure $\|\ \|$ will be used later on also for sequents, proofs and clause sets.

Gentzen's calculus **LK** is based on so-called sequents.

Definition 2.1.9 (sequent)**.** Let Γ and Δ be two multi-sets of formulas and $\vdash$ be a symbol not belonging to the logical language. Then $\Gamma \vdash \Delta$ is called a sequent.

If $S_1 \colon \Gamma \vdash \Delta$ and $S_2 \colon \Pi \vdash \Lambda$ are sequents we define the *concatenation* of S_1 and S_2 (notation $S_1 \circ S_2$) as $\Gamma, \Pi \vdash \Delta, \Lambda$.

Definition 2.1.10 (semantics of a sequent)**.** Semantically a sequent

$$S : A_1, \ldots, A_n \vdash B_1, \ldots, B_m$$

stands for

$$F(S) : \bigwedge_{i=1}^{n} A_i \to \bigvee_{j=1}^{m} B_j.$$

In particular, we define $\mathscr{M}$ to be an interpretation of S if $\mathscr{M}$ is an interpretation of $F(S)$. If $n = 0$ we assign $\top$ to $\bigwedge_{i=1}^{n} A_i$, if $m = 0$ we assign $\bot$ to $\bigvee_{j=1}^{m} B_j$. The empty sequent is represented by $\top \to \bot$, which is equivalent to $\bot$ and represents falsum. S is true in $\mathscr{M}$ if $F(S)$ is true in $\mathscr{M}$. S is called valid if $F(S)$ is valid.

Definition 2.1.11. Substitutions can be extended to sequents in an obvious way: If $S : A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ is a sequent and σ is a substitution, then

$$S\sigma : A_1\sigma, \ldots, A_n\sigma \vdash B_1\sigma, \ldots, B_m\sigma.$$

Definition 2.1.12 (prenex sequent)**.** A sequent $S \colon A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ is prenex (or in prenex form) if the formulas A_i and B_j, where $1 \leq i \leq n$ and $1 \leq j \leq m$, are prenex formulas.

In some proof theoretic transformations in this work we will use *normalized sequents*.

Definition 2.1.13 (normalized sequents)**.** A sequent $\Gamma \vdash \Delta$ is called *normalized* if the multiplicity of all formulas occurring in Γ (Δ) is one, more precisely: if $\Gamma = A_1, \ldots, A_n$ and $\Delta = B_1, \ldots, B_m$ then $A_i \neq A_j$ for $i \neq j$ $(i, j \in \{1, \ldots, n\})$ and $B_l \neq B_k$ for $l \neq k$ $(l, k \in \{1, \ldots, m\})$.

Remark. Note that every **LK**-proof φ of S can be easily transformed into a proof of a normalized sequent: just apply the contraction rules to S.

In normalized sequents the multisets become sets which allows us to define some set-based operations on sequents:

Definition 2.1.14. Let S_1, S_2 be two normalized sequents such that $S_1 = \Gamma_1 \vdash \Delta_1$, $S_2 = \Gamma_2 \vdash \Delta_2$. Let $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Delta = \Delta_1 \cap \Delta_2$. We define the following operations on S_1, S_2:

$$S_1 \cap S_2 = \Gamma \vdash \Delta,\ S_1 \setminus S_2 = (\Gamma_1 \setminus \Gamma) \vdash (\Delta_1 \setminus \Delta),\ S_2 \setminus S_1 = (\Gamma_2 \setminus \Gamma) \vdash (\Delta_2 \setminus \Delta).$$

The sequents S_1 and S_2 are called *disjoint* if $S_1 \cap S_2 = \vdash$. Then, obviously, $S_1 \cap S_2$, $S_1 \setminus S_2$ and $S_2 \setminus S_1$ are pairwise disjoint and

$$\begin{aligned} S_1 &= (S_1 \cap S_2) \circ (S_1 \setminus S_2), \\ S_2 &= (S_1 \cap S_2) \circ (S_2 \setminus S_1). \end{aligned}$$

Frequently, we will use the notion of strong and weak quantifiers. This notion is defined via the polarity of a formula occurrence in a sequent.

Definition 2.1.15 (polarity [BL11])**.** Let λ be an occurrence of a formula A in B.

- If $A \equiv B$ then λ is a positive occurrence in B.
- If $B \equiv \neg C$ and λ is a positive (negative) occurrence of A in C then the corresponding occurrence λ' of A in B is negative (positive).
- If $B \equiv C \wedge D$, $B \equiv C \vee D$, $B \equiv \forall x C$ or $B \equiv \exists x C$ and λ is a positive (negative) occurrence of A in C (or in D) then the corresponding occurrence λ' of A in B is positive (negative).
- If $B \equiv C \to D$ and λ is a positive (negative) occurrence of A in D then the corresponding occurrence λ' in B is positive (negative). If λ is a positive (negative) occurrence of A in C then the corresponding occurrence λ' of A in B is negative (positive).

If there exists a positive (negative) occurrence of a formula A in B we say that A is of positive (negative) polarity in B.

Definition 2.1.16 (strong and weak quantifiers [BL11])**.** If $\forall x$ occurs positively (negatively) in B then $\forall x$ is called a strong (weak) quantifier. If $\exists x$ occurs positively (negatively) in B then $\exists x$ is called a weak (strong) quantifier.

This definition can be extended to sequents: A sequent is called weakly (strongly) quantified if all quantifier occurrences in S are weak (strong).

Many proof-theoretic transformations in this work will be based on skolemized sequents.

Definition 2.1.17 (skolemized sequent [BL11])**.** Let sk be a function that maps closed formulas into closed ones. $sk(F)$ is the skolemization of F and defined as

$$sk(F) = F \text{ if } F \text{ does not contain strong quantifiers,}$$

and otherwise assume that Qy is the first strong quantifier in F (in a tree ordering) in the scope of the weak quantifiers $Q_1x_1Q_2x_2 \ldots Q_nx_n$. Let f be an n-ary function symbol not occurring in F, then $sk(F)$ is inductively defined as

$$sk(F) = sk(F_{Qy}\{y \leftarrow f(x_1, \ldots, x_n)\}),$$

where F_{Qy} is F after removal of Qy.

Now let $S\colon A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ be a sequent consisting of closed formulas only and

$$sk((A_1 \wedge \ldots \wedge A_n) \to (B_1 \vee \ldots \vee B_m)) = (A'_1 \wedge \ldots \wedge A'_n) \to (B'_1 \vee \ldots \vee B'_m).$$

Then the sequent $S'\colon A'_1, \ldots, A'_n \vdash B'_1, \ldots, B'_m$ is the skolemization of S.

Like most other calculi Gentzen's **LK** is based on axioms and rules.

Definition 2.1.18 (axiom set)**.** A (possibly infinite) set $\mathscr{A}$ of sequents is called an axiom set if it is closed under substitution, i.e. for all $S \in \mathscr{A}$ and for all substitutions θ we have $S\theta \in \mathscr{A}$. If $\mathscr{A}$ contains only atomic sequents we speak about an atomic axiom set.

Definition 2.1.19 (standard axiom set)**.** Let $\mathscr{A}_T$ be the smallest axiom set containing all sequents of the form $A \vdash A$ for arbitrary atomic formulas A. Then $\mathscr{A}_T$ is called the *standard axiom set*.

Definition 2.1.20 (sequent calculus **LK** [BL11])**.** We use a variant of Gentzen's version of **LK** [Gen35]. Since we consider multi-sets of formulas, we do not need exchange or permutation rules. There are two groups of rules, the logical and the structural ones. All rules except the cut have left and right versions, denoted by l and r, respectively. The binary rules are of multiplicative type, i.e. no auto-contraction of the context is applied. In the following, A and B denote formulas whereas $\Gamma, \Delta, \Pi, \Lambda$ denote multi-sets of formulas. In the rules there are introducing or auxiliary formulas in the premises and introduced or principal formulas in the conclusion. We indicate these formulas for all rules, auxiliary formula occurrences are marked by $+$ and principal formula occurrences are marked by $*$. We frequently say auxiliary (main) formula instead of auxiliary (main) formula occurrence.

The logical rules:

$\wedge$-introduction

$$\frac{A^+, B^+, \Gamma \vdash \Delta}{(A \wedge B)^*, \Gamma \vdash \Delta} \wedge_l \qquad \frac{\Gamma_1 \vdash \Delta_1, A^+ \qquad \Gamma_2 \vdash \Delta_2, B^+}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, (A \wedge B)^*} \wedge_r$$

$\vee$-introduction

$$\frac{\Gamma \vdash \Delta, A^+, B^+}{\Gamma \vdash \Delta, (A \vee B)^+} \vee_r \qquad \frac{A^+, \Gamma_1 \vdash \Delta_1 \qquad B^+, \Gamma_2 \vdash \Delta_2}{(A \vee B)^*, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \vee_l$$

$\to$-introduction

$$\frac{\Gamma_1 \vdash \Delta_1, A^+ \qquad B^+, \Gamma_2 \vdash \Delta_2}{(A \to B)^*, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \to_l \qquad \frac{A^+, \Gamma \vdash \Delta, B^+}{\Gamma \vdash \Delta, (A \to B)^*} \to_r$$

$\neg$-introduction

$$\frac{\Gamma \vdash \Delta, A^+}{\neg A^*, \Gamma \vdash \Delta}\,\neg_l \qquad\qquad \frac{A^+, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A^*}\,\neg_r$$

$\forall$-introduction

$$\frac{A\{x \leftarrow t\}^+, \Gamma \vdash \Delta}{\forall x A^*, \Gamma \vdash \Delta}\,\forall_l \qquad\qquad \frac{\Gamma \vdash \Delta, A\{x \leftarrow \alpha\}^+}{\Gamma \vdash \Delta, \forall x A^*}\,\forall_r$$

where t is an arbitrary term and α is a free variable which may not occur in Γ, Δ, A. α is called an eigenvariable.

$\exists$-introduction

$$\frac{A\{x \leftarrow \alpha\}^+, \Gamma \vdash \Delta}{\exists x A^*, \Gamma \vdash \Delta}\,\exists_l \qquad\qquad \frac{\Gamma \vdash \Delta, A\{x \leftarrow t\}^+}{\Gamma \vdash \Delta, \exists x A^*}\,\exists_r$$

where the variable conditions for $\exists_l$ are the same as those for $\forall_r$ and similarly for $\exists_r$ and $\forall_l$. The quantifier-rules $\forall_l, \exists_r$ are called *weak*, the rules $\exists_l, \forall_r$ *strong*.

The structural rules:

weakening

$$\frac{\Gamma \vdash \Delta}{A^*, \Gamma \vdash \Delta}\,w_l \qquad\qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A^*}\,w_r$$

contraction

$$\frac{A^+, A^+, \Gamma \vdash \Delta}{A^*, \Gamma \vdash \Delta}\,c_l \qquad\qquad \frac{\Gamma \vdash \Delta, A^+, A^+}{\Gamma \vdash \Delta, A^*}\,c_r$$

cut

$$\frac{\Gamma_1 \vdash \Delta_1, A^m \qquad A^n, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut(A, m, n)$$

where $m, n \geq 1$. The formulas A^m and A^n are the auxiliary formulas of $cut(A, m, n)$ (they are also called cut-formulas) and there are not principal ones. We use this form of the cut rule as it can be easily transformed into the resolution rule, which will be used in the method CERES.

An **LK**-proof from a set of axioms $\mathscr{A}$ is a tree formed according to the rules of **LK** such that all leaves are in $\mathscr{A}$. In general, our **LK**-proofs will be proofs from $\mathscr{A}_T$.

When it is clear from the context, we will omit marking auxiliary and principal formula occurrences with $+$ and $*$.

We denote occurrences of sequents in a derivation with $\nu_0, \nu_1, \ldots \nu_\alpha$. The sequent at node ν_α is denoted by $Seq(\nu_\alpha)$. A formula occurrence in a sequent will be denoted by

$\mu_0, \mu_1, \ldots, \mu_\beta$. Let μ be an occurrence of a formula A in a sequent S, then we define $\mathit{formula}(\mu) = A$ in S.

Definition 2.1.21 (regularity)**.** An **LK**-derivation φ is called regular if

- all eigenvariables of quantifier introductions $\forall_r$ and $\exists_l$ in φ are mutually different,
- an eigenvariable α occurs as an eigenvariable in a proof node ν then α occurs only above ν in the proof tree.

There is a straightforward transformation from **LK**-derivations into regular ones: rename the eigenvariables in different subderivations.

Definition 2.1.22 (prenex derivation)**.** A derivation is prenex (or in prenex form) if all its sequents are prenex.

Definition 2.1.23 (skolemized derivation)**.** A derivation is skolemized if its end-sequent is a skolemized sequent.

LK can be easily extended by adding rules. For instance, in Section 3.5 we will work with $\mathbf{LK}_=$, an extension of **LK** with equality rules.

Definition 2.1.24 (sequent calculus $\mathbf{LK}_=$ [LL19])**.** $\mathbf{LK}_=$ is **LK** extended with equality rules as in [BHL+06]:

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad A[s]_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} =_{l1} \quad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad A[s]_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} =_{l2}$$

for inference on the left and

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad \Gamma_2 \vdash \Delta_2, A[s]_\Lambda}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda} =_{r1} \quad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad \Gamma_2 \vdash \Delta_2, A[s]_\Lambda}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda} =_{r2}$$

on the right, where Λ denotes a position of a subterm where replacement of s by t has to be performed. We call $s = t$ the active equation of the rules.

Note that, on atomic sequents, the rules coincide with paramodulation – under previous application of the most general unifier.

Axioms:
Any set of atomic sequents which is closed under substitution and contains the sequent $\vdash x = x$ (and thus all sequents of the form $\vdash t = t$ for arbitrary terms t) is admitted as an axiom set. We define the axiom set $Ax = \mathscr{A}_T \cup \{\vdash t = t \mid t \text{ a term}\}$.

An $\mathbf{LK}_=$-proof from a set of axioms $\mathscr{A}$ is a tree formed according to the rules of $\mathbf{LK}_=$ such that all leaves are in $\mathscr{A}$.

2.2 Herbrand Sequents

Herbrand's theorem [Her30] is a fundamental result in proof theory, connecting first-order logic to propositional logic. In its simplest formulation it states that a formula $\exists x F(x)$ is provable only if there exist terms $t_1, \ldots, t_n$ such that the Herbrand disjunction $F(t_1) \vee \ldots \vee F(t_n)$ is provable. The crucial information that separates first-order from propositional logic is the information about the terms $t_1, \ldots, t_n$. Given the terms $t_1, \ldots, t_n$, first-order provability and provability in propositional logic coincide. A Herbrand sequent is a propositionally valid sequent which is computed as an instantiation of the end-sequent of a proof, where quantified formulas occurring in the end-sequent are replaced by their Herbrand instances. A direct approach to extract Herbrand sequents from proofs, and to the best of our knowledge the first correct proof of Herbrand's theorem, is based on Hilbert's ε-calculus [HB39, MZ06]. The extended first ε-theorem eliminates algorithmically so-called critical formulas, obtaining a Herbrand disjunction. Herbrand sequents can also be extracted directly from proofs of prenex end-sequents with at most quantifier-free cuts [BL11, HLWP08]. The method of Herbrand sequent extraction is based on collecting instances of quantified formulas directly from the proof.

Definition 2.2.1 (thread)**.** Let π be an **LK**-proof. A *thread* in π is a path of sequent occurrences in the proof tree beginning at the root and ending in an initial sequent.

Remark. Let $\vartheta\colon \nu_0, \ldots, \nu_\alpha$ be a thread in π then every ν_{i+1} (for $i < \alpha$) is a premise of ν_i and an immediate inference ancestor of ν_i.

To every thread there exist corresponding sequences of formula occurrences in the corresponding sequents. Such sequence of occurrences starting in an occurrence of the end-sequent is called a *trace*.

Definition 2.2.2 (trace)**.** Let π be an **LK**-proof of an end-sequent occurring at the root node ν_0 and $\vartheta\colon \nu_0, \ldots, \nu_\alpha$ be a thread in π. Let μ_0 be a formula occurrence in $Seq(\nu_0)$. For all $i \in \{0, \ldots, \alpha - 1\}$ let μ_{i+1} be an occurrence in $Seq(\nu_{i+1})$ which is an ancestor occurrence of μ_i in $Seq(\nu_i)$. Then the sequence $(\nu_0, \mu_0), \ldots, (\nu_\alpha, \mu_\alpha)$ is called a *trace of* μ_0 *in* ϑ. A sequence $\mu_0, \ldots, \mu_\alpha$ is called a *trace of* μ_0 *in* π if there exists a thread ϑ in π such that $(\nu_0, \mu_0), \ldots, (\nu_\alpha, \mu_\alpha)$ is a trace of μ_0 in ϑ.

Remark. In Definition 2.2.2 we have assumed that every μ_i has an ancestor occurrence μ_{i+1}. If μ_i is the principal formula of a weakening the occurrence μ_{i+1} does not exist. In this case we cannot define a trace of μ_0 corresponding to a thread ϑ. So traces, like threads, start in the root and end in an initial sequent. Moreover, as ancestor occurrences are not unique, a thread may define several traces.

Every trace defines a sequence of formulas, the yield.

Definition 2.2.3 (yield)**.** Let $\tau\colon \mu_0, \ldots, \mu_\alpha$ be a trace in an **LK**-proof π then the sequence $formula(\mu_0), \ldots, formula(\mu_\alpha)$ is called the *yield* of τ.

Example 2.2.1. Consider the **LK**-proof $\varphi =$

$$\dfrac{\dfrac{(\varphi_1)}{\nu_2 : B, B \vdash \mathbf{C}, C, Q(a)} \quad \dfrac{\dfrac{\dfrac{\dfrac{Q(a) \vdash Q(a)}{P(a), Q(a) \vdash Q(a)}\, w:l}{Q(a) \vdash P(a) \to Q(a)}\, \to\!: r}{Q(v) \vdash \exists y P(a) \to Q(y)}\, \exists : r}{Q(v), B \vdash C}\, w:l}{\nu_1 : B, B, B \vdash \mathbf{C}, C, C}\, cut}{\nu_0 : \forall x(P(x) \to Q(x)) \vdash \exists \mathbf{y}(\mathbf{P(a)} \to \mathbf{Q(y)})}\, c^*$$

where $\varphi_1 =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\nu_7 : \mathbf{P(a)} \vdash P(a)}{\nu_6 : \mathbf{P(a)} \vdash P(a), Q(a)}\, w:r}{\nu_5 : \vdash \mathbf{P(a)} \to \mathbf{Q(a)}, P(a)}\, \to\!: r}{\nu_4 : \vdash \exists \mathbf{y}(\mathbf{P(a)} \to \mathbf{Q(y)}), P(a)}\, \exists : r}{\nu_3 : B \vdash \mathbf{C}, P(a)}\, w:l \qquad \dfrac{\dfrac{\dfrac{P(a) \vdash P(a) \quad Q(a) \vdash Q(a)}{P(a), P(a) \to Q(a) \vdash Q(a)}\, \to\!: l}{P(a), \forall x(P(x) \to Q(x)) \vdash Q(a)}\, \forall : l}{P(a), B \vdash C, Q(a)}\, w:r}{\nu_2 : B, B \vdash \mathbf{C}, C, Q(a)}\, cut$$

and $B = \forall x(P(x) \to Q(x))$ and $C = \exists y(P(a) \to Q(y))$.

Then $\vartheta = \nu_0, \nu_1, \nu_2, \nu_3, \nu_4, \nu_5, \nu_6, \nu_7$ is a thread in φ. Let μ_0 be the formula occurrence of $\exists y(P(a) \to Q(y))$ in $\mathrm{Seq}(\nu_0)$. Then the sequence

$$\tau \quad = \quad (\nu_0, \mu_0), (\nu_1, \mu_0), (\nu_2, \mu_0), (\nu_3, \mu_0), (\nu_4, \mu_0), (\nu_5, \mu_1), (\nu_6, \mu_2), (\nu_7, \mu_2)$$

is a trace of μ_0 in ϑ, where μ_1 is the occurrence of $P(a) \to Q(a)$ in $\mathrm{Seq}(\nu_5)$ and μ_2 the occurrence of $P(a)$ in $\mathrm{Seq}(\nu_6)$ and $\mathrm{Seq}(\nu_7)$. This trace is indicated in bold letters in the derivation φ. The sequence $\exists y(P(a) \to Q(y))$, $\exists y(P(a) \to Q(y))$, $\exists y(P(a) \to Q(y))$, $\exists y(P(a) \to Q(y))$, $\exists y(P(a) \to Q(y))$, $P(a) \to Q(a)$, $P(a)$, $P(a)$ is the yield of τ.

Proposition 2.2.1. *Let π be a cut-free* **LK***-proof of a skolemized prenex end-sequent S and let μ_0 be an occurrence of a formula $F\colon Qx_1 \ldots Qx_\beta.F'(x_1, \ldots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and F' is quantifier-free. Let τ be a trace of μ_0 in π. Then there exist a substitution σ such that $F'\sigma$ occurs in the yield of τ.*

Proof. By induction on the number of quantifiers $Qx_1 \ldots Qx_\beta$ in F. Base case: the formula F is QxF'. We follow the trace τ of the occurrence μ_0 of F in the proof-tree; by the subformula property and as axioms are atomic the quantified variable x in F has to be replaced by a term t in the proof-tree and hence, $\{x \leftarrow t\}$ is the desired substitution σ such that $F'\sigma$ occurs in the yield of τ.

Assume there are n quantifiers in the formula $F\colon Qx_1 \ldots Qx_n.F'(x_1, \ldots, x_n)$, then there exist a substitution σ such that $F'\sigma$ occurs in the yield of τ (IH).

Now let us consider $n+1$ quantifiers $Qx_1 \dots Qx_{n+1}$ in $F\colon Qx_1 \dots Qx_n, Qx_{n+1}.F'(x_1, \dots, x_{n+1})$. We follow the trace τ of the occurrence μ_0 of F in the proof-tree; by the subformula property and the fact that axioms are atomic the first quantifier that will be eliminated is Qx_1. Therefore, x_1 will be replaced by a term t_1 and we obtain a substitution $\sigma_1 = \{x_1 \leftarrow t_1\}$. The formula F hence changes to a formula $F''\colon Qx_2 \dots Qx_{n+1}.F'\sigma_1$. In F'' there are only n quantifiers and hence we can apply the IH: there exists a substitution σ such that the formula $F'\sigma_1\sigma$ occurs in the yield of τ. $\sigma_1\sigma$ is the desired substitution. □

Following Proposition 2.2.1 we can define a Herbrand substitution.

Definition 2.2.4 (Herbrand substitution)**.** Let π be a cut-free **LK**-proof of a skolemized prenex end-sequent S and μ_0 an occurrence of a formula $F\colon Qx_1 \dots Qx_\beta.F'(x_1, \dots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and F' is quantifier-free. Let τ be a trace of μ_0 in π such that $F'(t_1, \dots, t_\beta)$ occurs in the yield of τ. Then $S(\pi, \tau) = \{x_1 \leftarrow t_1, \dots, x_\beta \leftarrow t_\beta\}$ is a Herbrand substitution for F in π .

Remark. Note that the formula $F'(t_1, \dots, t_\beta)$ and the substitution $\{x_1 \leftarrow t_1, \dots, x_\beta \leftarrow t_\beta\}$ in the propositions above are uniquely defined by the trace τ.

Definition 2.2.5. Let π be a cut-free **LK**-derivation of a skolemized prenex end-sequent S and let μ_0 be an occurrence of a formula $F\colon Qx_1 \dots Qx_\beta.F'(x_1, \dots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and F' is quantifier-free. We define as $T(\mu_0, \pi)$ the set of all traces of μ_0 in π.

Definition 2.2.6 (Herbrand instances)**.** Let π be a cut-free **LK**-derivation of a skolemized prenex end-sequent S and let μ_0 be an occurrence of a formula $F\colon Qx_1 \dots Qx_\beta.\ F'(x_1, \dots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and F' is quantifier-free. Let $\tau\colon \mu_0, \dots, \mu_\alpha$ be a trace in $T(\mu_0, \pi)$. The set of all Herbrand instances for μ_0 is then defined as

$$S(\pi, \mu_0) = \bigcup\{S(\pi, \tau) \mid \tau \in T(\mu_0, \pi)\}.$$

Similarly, let $\mu_0, \dots, \mu_n$ be all occurrences of quantified formulas in S. Then

$$S(\pi) = \{S(\pi, \mu_i) \mid i = 0, \dots, n\}$$

is the set of Herbrand instances of π.

Definition 2.2.7 (Herbrand expansion)**.** Let π be a cut-free **LK**-derivation of a skolemized prenex end-sequent S and let μ_0 be an occurrence of a formula $F\colon Qx_1 \dots Qx_\beta.$ $F'(x_1, \dots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and F' is quantifier-free. Let

$$S(\pi, \mu_0) = \{\{x_1 \leftarrow t_1^1, \dots, x_\beta \leftarrow t_\beta^1\}, \dots, \{x_1 \leftarrow t_1^\alpha, \dots, x_\beta \leftarrow t_\beta^\alpha\}\}$$

be the set of Herbrand instances for μ_0. Then the Herbrand expansion for μ_0 is defined as

$$H(\pi, \mu_0)\colon F'(t_1^1, \dots, t_\beta^1), \dots, F'(t_1^\alpha, \dots, t_\beta^\alpha).$$

Example 2.2.2. Consider the **LK**-proof φ from Example 2.2.1. Let μ_0 be the occurrence of the formula $\forall x(P(x) \to Q(x))$ in the end-sequent. The set of all Herbrand instances for μ_0 is $S(\varphi, \mu_0) = \{x \leftarrow a\}$. The Herbrand expansion for μ_0 is $H(\varphi, \mu_0) = P(a) \to Q(a)$.

Let μ_0' be the occurrence of the formula $\exists x(P(a) \to Q(y))$ in the end-sequent. The set of all Herbrand instances for μ_0' is $S(\varphi, \mu_0') = \{x \leftarrow a, x \leftarrow a\}$. The Herbrand expansion for μ_0' is $H(\varphi, \mu_0') = P(a) \to Q(a)$.

By Herbrand's theorem every valid weakly quantified sequent has a Herbrand sequent. Completeness proofs in automated deduction are based on a semantic proof of this theorem, which is based on König's lemma and thus a weak form of the axiom of choice. However, Herbrand sequents S' from **LK**-proofs φ of S can be obtained constructively, provided the cut-formulas in φ are quantifier-free [BL94,HLWP08]. The Herbrand sequent of a prenex **LK**-proof with quantifier-free cuts can be constructed by replacing quantified formulas occurring in an end-sequent by their Herbrand expansions.

Definition 2.2.8 (Herbrand sequent)**.** Let π be an **LK**-proof with quantifier-free cuts of a skolemized prenex end-sequent

$$S\colon \Gamma, F_0, \ldots, F_n \vdash G_0, \ldots, G_m, \Delta$$

such that Γ, Δ are multisets of quantifier-free formulas and $F_1, \ldots, F_n, G_1, \ldots, G_m$ are formulas containing quantifiers. Let $\mu_0, \ldots, \mu_n$ be the occurrences of $F_0, \ldots, F_n$ in S and let $\nu_0, \ldots, \nu_m$ the occurrences of $G_0, \ldots, G_m$ in S. Then the Herbrand sequent of π is defined as

$$H(\pi)\colon \Gamma, H(\pi, \mu_0), \ldots, H(\pi, \mu_n) \vdash H(\pi, \nu_0), \ldots H(\pi, \nu_m), \Delta.$$

Theorem 2.2.2 (soundness)**.** *Let π be an* **LK***-proof with quantifier-free cuts of a skolemized prenex end-sequent S, then*

1. *the formulas of $H(\pi)$ are substitution instances of the formulas of S without their quantifiers.*
2. *$H(\pi)$ is valid.*

Proof. Item 1 follows directly by the construction of $H(\pi)$, as quantified formulas are replaced by their substitution instances. Item 2 follows because every Herbrand expansion of a quantified formula in S, i.e. every substitution instance of a quantified formula in S, occurs in the corresponding yield. Therefore, every substitution instance is provable. But then, the Herbrand sequent is provable as well. □

Example 2.2.3. Let φ be the **LK**-proof in Example 2.2.1. Its Herbrand sequent is

$$P(a) \to Q(a) \vdash P(a) \to Q(a).$$

2.3 Expansion Trees and Expansion Proofs

As Herbrand sequents can be obtained only from prenex end-sequents, a more general notion is necessary to describe the instantiations of quantified variables occurring nested in an end-sequent. To this aim we use so-called *expansion trees*. Expansion trees, first introduced in [Mil87], are natural structures representing the instantiated variables for quantified formulas. These structures record the substitutions for quantifiers in the original formula and the formulas resulting from instantiations. Expansion trees may contain logical connectives as well as the new connective $+^t$, where t is a term. Informally, an expression of the kind $QxA(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n$ is an expansion tree, where $Q \in \{\forall, \exists\}$ and $t_1, ..., t_n$ are terms such that this expansion tree represents the result when instantiating the quantified expression $QxA(x)$ with the terms $t_1, ..., t_n$ to get the structures E_i. E_i is again an expansion tree representing $A(t_i)$ for $i = 1, ..., n$. In the method for the extraction of expansion proofs we consider skolemized proofs. Moreover, the cuts are quantifier-free. Therefore, our definition is a modified one as we do not have quantifiers with eigenvariable conditions. The definitions in this section are based on [Mil87, LL19]. The definition below takes care that only trees with weak quantifiers are constructed.

Definition 2.3.1. Expansion trees, dual expansion trees and a function Sh (shallow) which maps expansion trees to formulas are defined inductively as follows:

1. If A is a quantifier-free formula then A is an expansion tree (and a dual expansion tree) for A and $Sh(A) = A$.
2. If E is an expansion tree then $\neg E$ is a dual expansion tree and $Sh(\neg E) = \neg Sh(E)$.
3. If E is a dual expansion tree then $\neg E$ is an expansion tree and $Sh(\neg E) = \neg Sh(E)$.
4. If E_1 and E_2 are (dual) expansion trees, then $E_1 \wedge E_2$, $E_1 \vee E_2$ are (dual) expansion trees and $Sh(E_1 \wedge E_2) = Sh(E_1) \wedge Sh(E_2)$, the same for $\vee$.
5. If E_1 is a dual expansion tree and E_2 is an expansion tree then $E_1 \to E_2$ is an expansion tree and $Sh(E_1 \to E_2) = Sh(E_1) \to Sh(E_2)$.
6. If E_1 is an expansion tree and E_2 is a dual expansion tree then $E_1 \to E_2$ is a dual expansion tree and $Sh(E_1 \to E_2) = Sh(E_1) \to Sh(E_2)$.
7. Let $A(x)$ be a formula and $t_1, ..., t_n$ $(n \geq 1)$ be a list of terms. Let $E_1, ..., E_n$ be expansion trees with $Sh(E_i) = A(t_i)$ for $i = 1, ..., n$; then $\exists xA(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n$ is an expansion tree with $Sh(\exists xA(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n) = \exists xA(x)$.
8. Let $A(x)$ be a formula and $t_1, ..., t_n$ $(n \geq 1)$ be a list of terms. Let $E_1, ..., E_n$ be dual expansion trees with $Sh(E_i) = A(t_i)$ for $i = 1, ..., n$; then $\forall xA(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n$ is a dual expansion tree with $Sh(\forall xA(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n) = \forall xA(x)$.

Substitutions can be extended to expansion trees.

Definition 2.3.2. Let σ be a substitution and E an expansion tree (or a dual expansion tree). Then $E\sigma$ is inductively defined as:

1. If $E = A$ for a quantifier-free formula A then $E\sigma = A\sigma$.
2. If $E = E_1 \circ E_2$, where E_1 and E_2 are (dual) expansion trees and $\circ \in \{\wedge, \vee, \rightarrow\}$, then $E\sigma = E_1\sigma \circ E_2\sigma$.
3. If $E = \exists x A(x) +^{t_1} E_1 +^{t_2} \ldots +^{t_n} E_n$, where $E_1, \ldots, E_n$ are expansion trees and $t_1, \ldots, t_n$ $(n \geq 1)$ a list of terms, then $E\sigma = \exists x A(x) +^{t_1\sigma} E_1\sigma +^{t_2\sigma} \ldots +^{t_n\sigma} E_n\sigma$.
4. If $E = \forall x A(x) +^{t_1} E_1 +^{t_2} \ldots +^{t_n} E_n$, where $E_1, \ldots, E_n$ are dual expansion trees and $t_1, \ldots, t_n$ $(n \geq 1)$ a list of terms, then $E\sigma = \forall x A(x) +^{t_1\sigma} E_1\sigma +^{t_2\sigma} \ldots +^{t_n\sigma} E_n\sigma$.

Example 2.3.1. Let $P(x)$ be an atom. Then $P(a)$ is a dual expansion tree and $\forall x.P(x) +^a P(a)$ is a dual expansion tree. $\exists x.P(x) +^a P(a)$ is an expansion tree. So

$$\forall x.P(x) +^a P(a) \rightarrow \exists x.P(x) +^a P(a)$$

is an expansion tree. But note that

$$\forall x.P(x) +^a P(a) \rightarrow \forall x.P(x) +^a P(a)$$

is not an expansion tree according to Definition 2.3.1 as $\forall x.P(x) +^a P(a)$ is a dual expansion tree but not an expansion tree. Indeed, having strong quantifiers with the type of expansion defined above would be unsound.

The function Dp (deep) maps expansion trees (and dual expansion trees) to quantifier-free formulas, their *full expansion*.

Definition 2.3.3. Dp maps a (dual) expansion tree to a formula as follows:

$$\begin{array}{rcl} Dp(E) & = & E \text{ for an atomic expansion tree E,} \\ Dp(\neg E) & = & \neg Dp(E), \\ Dp(E_1 \circ E_2) & = & Dp(E_1) \circ Dp(E_2) \text{ for } \circ \in \{\wedge, \vee, \rightarrow\}, \\ Dp(\exists x A +^{t_1} E_1 +^{t_2} \ldots +^{t_n} E_n) & = & Dp(E_1) \vee \ldots \vee Dp(E_n), \\ Dp(\forall x A +^{t_1} E_1 +^{t_2} \ldots +^{t_n} E_n) & = & Dp(E_1) \wedge \ldots \wedge Dp(E_n). \end{array}$$

In [Mil87] a notion of expansion proof was defined from expansion trees using the conditions acyclicity and tautology. Acyclicity ensures that there are no cycles between the strong quantifier nodes in the expansion tree. Since our formulas are skolemized and hence do not contain strong quantifiers, we do not need this condition.

Definition 2.3.4 (expansion proof)**.** Let ET be an expansion tree of a formula A without strong quantifiers. Then ET is called an expansion proof of A from a set of axioms $\mathscr{A}$ if $Sh(ET) = A$ and $\mathscr{A} \models Dp(ET)$.

Expansion proofs encode a proof of validity of the formula they represent. They can be directly translated into sequent calculus, see [Mil87], and the transformation is based on so-called q-sequents, which we refer to as s-expansion trees (sequent of expansion trees) in this work.

Definition 2.3.5 (s-expansion tree)**.** The structure $S\colon\ \Gamma \vdash \Delta$ where $\Delta\colon Q_1, \dots Q_s$ is a multiset of expansion trees, $\Gamma\colon P_1, \dots, P_r$ is a multiset of dual expansion trees is called an *s-expansion tree*. If $\neg\Gamma \vee \Delta$ (which stands for $\neg P_1 \vee \dots \vee \neg P_r \vee Q_1 \vee \dots \vee Q_s$) is an expansion proof then S is called an *s-expansion proof*. This expansion proof is the expansion proof associated with S; the sequent

$$Seq(S)\colon\ \ Sh(P_1), ..., Sh(P_r) \to Sh(Q_1), ..., Sh(Q_s)$$

is the sequent associated with S.

Substitutions on expansion trees can be extended to s-expansion trees. Let σ be a substitution and $S\colon P_1, \dots, P_r \vdash Q_1, \dots Q_s$ an s-expansion tree, then $S\sigma\colon P_1\sigma, \dots, P_r\sigma \vdash Q_1\sigma, \dots Q_s\sigma$.

Definition 2.3.6 (\-operator on s-expansion trees)**.** Let $S\colon\ \Gamma \vdash \Delta, F$ be an s-expansion tree, where $\Delta\colon Q_1, \dots, Q_s$ is a multiset of expansion trees, $\Gamma\colon P_1, \dots, P_r$ is a multiset of dual expansion trees and F a quantifier-free formula not occurring in Δ, Γ. Then we define $S \setminus\ \vdash F = \Gamma \vdash \Delta$.

Note that the \-operator is only used on quantifier-free formulas in an s-expansion tree. Therefore, we do not eliminate expansion trees but only formulas that do not expand further and hence, the operator does not have to be defined inductively over the whole tree.

It is also possible to read off expansion proofs from sequent calculus proofs. Note that the expansion proof of a proof φ is a sequent of expansion trees, which are defined to be the expansion trees of all formulas in the end-sequent of φ. An algorithm for the extraction of expansion proofs from sequent calculus proofs is presented in [Mil87] and modified algorithms (dealing with cuts and equality) are presented in [HLRR13] and [HW13]. There exist also algorithms for a transformation of resolution-trees into expansion-trees, see [Pfe84].

We will use an algorithm that is briefly described in [HLRR13]. In order to show how an expansion proof is extracted from a proof in **LK**, we first need to define an operation on expansion trees. The Merge operator on expansion trees is defined in [Mil87]. Intuitively, two expansion trees T_1 and T_2 can be merged, if $Sh(T_1) = Sh(T_2)$. We give a definition adapted to our concept of expansion tree.

Definition 2.3.7 (merge)**.** Let E_1, E_2 be (dual) expansion trees such that $Sh(E_1) = Sh(E_2)$. We define the merge inductively on the complexity of E_1.

- If E_1 is an atom then E_2 is an atom too and $E_1 = E_2$; we define $\mathrm{Merge}_t(E_1, E_2) = E_1$.
- If $E_1 = \neg E_1'$. Then $E_2 = \neg E_2'$ for some E_2'. Let $\mathrm{Merge}_t(E_1', E_2') = E_3'$, then $\mathrm{Merge}_t(E_1, E_2) = \neg E_3'$.
- Let $E_1 = E_{11} \circ E_{12}$ for $\circ \in \{\wedge, \vee, \rightarrow\}$. Then $E_2 = E_{21} \circ E_{22}$ for some E_{21}, E_{22}. Let $E_1' = \mathrm{Merge}_t(E_{11}, E_{21})$ and $E_2' = \mathrm{Merge}_t(E_{21}, E_{22})$. Then $\mathrm{Merge}_t(E_1, E_2) = E_1' \circ E_2'$.
- Let $E_1 = Qx.A(x) +^{t_1} E_{11} + \cdots +^{t_n} E_{1n}$. Then E_2 is of the form $Qx.A(x) +^{s_1} E_{21} + \cdots +^{s_m} E_{2m}$. Then

$$\mathrm{Merge}_t(E_1, E_2) = Qx.A(x) +^{t_1} E_{11} + \cdots +^{t_n} E_{1n} +^{s_1} E_{21} + \cdots +^{s_m} E_{2m}.$$

Example 2.3.2. Let $T_1 = \forall x Px +^a Pa$ and $T_2 = \forall x Px +^b Pb$ be two dual expansion trees then

$$\mathrm{Merge}_t(T_1, T_2) = \forall x Px +^a Pa +^b Pb.$$

The definition can be easily extended to more than two expansion trees. Let $T_1, \ldots, T_n$ (for $n \geq 2$) be (dual) expansion trees such that $Sh(T_i) = Sh(T_j)$ for all $i, j \in \{1, \ldots, n\}$. Then we define

$$\begin{aligned} \mathrm{merge}_t(T_1, T_2) &= \mathrm{Merge}_t(T_1, T_2), \\ \mathrm{merge}_t(T_1, \ldots, T_n) &= \mathrm{Merge}_t(\mathrm{merge}_t(T_1, \ldots, T_{n-1}), T_n) \text{ for } n > 2. \end{aligned}$$

It is also possible to merge s-expansion trees. As an s-expansion tree is defined via multisets of expansion trees, some expansion trees might occur more than once either on the left or on the right. In such cases merging s-expansion trees might become ambiguous. To avoid this ambiguity we restrict the merge of s-expansion trees to normalized ones, where the shallow forms occur only once.

Definition 2.3.8 (normalized s-expansion trees)**.** Let S be an s-expansion tree then S is called normalized if $Seq(S)$ is normalized.

In our applications we will merge s-expansion trees only if they correspond to end-sequents of proofs. Restricting the merge to normalized s-expansion proofs does not affect the generality of our approach, as s-expansion trees can be easily transformed into normalized ones.

Now we are ready to define the merging of normalized s-expansion trees.

Definition 2.3.9 (merge of s-expansion trees)**.** Let S_1 and S_2 be two normalized s-expansion trees and $S_1^* = Seq(S_1)$, $S_2^* = Seq(S_2)$. Then, by definition, S_1^*, S_2^* are

normalized sequents. We define $\Gamma^* \vdash \Delta^* = S_1^* \cap S_2^*$, $\Pi_1^* \vdash \Lambda_1^* = S_1^* \setminus S_2^*$, $\Pi_2^* \vdash \Lambda_2^* = S_2^* \setminus S_1^*$. Then

$$\begin{aligned} S_1^* &= (\Gamma^* \vdash \Delta^*) \circ (\Pi_1^* \vdash \Lambda_1^*), \\ S_2^* &= (\Gamma^* \vdash \Delta^*) \circ (\Pi_2^* \vdash \Lambda_2^*). \end{aligned}$$

Then there exist s-expansion trees $\Gamma \vdash \Delta$, $\Gamma' \vdash \Delta'$, $\Pi_1 \vdash \Lambda_1$ and $\Pi_2 \vdash \Lambda_2$ such that

$$\begin{aligned} S_1 &= (\Gamma \vdash \Delta) \circ (\Pi_1 \vdash \Lambda_1), \\ S_2 &= (\Gamma' \vdash \Delta') \circ (\Pi_2 \vdash \Lambda_2), \end{aligned}$$

where $Seq(\Gamma \vdash \Delta) = Seq(\Gamma' \vdash \Delta') = \Gamma^* \vdash \Delta^*$, $Seq(\Pi_1 \vdash \Lambda_1) = \Pi_1^* \vdash \Lambda_1^*$ and $Seq(\Pi_2 \vdash \Lambda_2) = \Pi_2^* \vdash \Lambda_2^*$. Note that the concatenation $\circ$ of sequents can be directly extended to s-expansion trees. Then there exist bijective mappings $\pi_l \colon \Gamma \to \Gamma'$ and $\pi_r \colon \Delta \to \Delta'$ with $\pi_l(T) = T'$ iff $Sh(T) = Sh(T')$ (the same for π_r). So assume

$$\begin{aligned} \Gamma \vdash \Delta &= T_1, \ldots, T_n \vdash T_{n+1}, \ldots, T_{n+m} \text{ and therefore} \\ \Gamma' \vdash \Delta' &= \pi_l(T_1), \ldots, \pi_l(T_n) \vdash \pi_r(T_{n+1}), \ldots, \pi_r(T_{n+m}). \end{aligned}$$

Now let $T_i^* = \mathrm{merge}_t(T_i, \pi_l(T_i))$ for $i = 1, \ldots, n$ and $T_i^* = \mathrm{merge}_t(T_i, \pi_r(T_i))$ for $i = n+1, \ldots, n+m$. Then we define

$$\mathrm{Merge}_s(S_1, S_2) = (T_1^*, \ldots, T_n^* \vdash T_{n+1}^*, \ldots, T_{n+m}^*) \circ (\Pi_1 \vdash \Lambda_1) \circ (\Pi_2 \vdash \Lambda_2).$$

Note that, by construction, $\mathrm{Merge}_s(S_1, S_2)$ is a normalized s-expansion tree.

We extend the merging of s-sequents to more than two as follows. Let $n \geq 2$ and $S_1, \ldots, S_n$ be normalized s-expansion trees. Then

$$\begin{aligned} \mathrm{merge}_s(S_1, S_2) &= \mathrm{Merge}_s(S_1, S_2) \text{ for } n = 2, \\ \mathrm{merge}_s(S_1, \ldots, S_n) &= \mathrm{Merge}_s(\mathrm{merge}_s(S_1, \ldots, S_{n-1}), S_n) \text{ for } n > 2. \end{aligned}$$

The s-expansion tree $\mathrm{merge}_s(S_1, \ldots, S_n)$ is also normal which can be verified by an obvious inductive argument.

Frequently we will write $\mathrm{merge}_s\{S_i \mid i = 1, \ldots, n\}$ for $\mathrm{merge}_s(S_1, \ldots, S_n)$. If no confusion arises we will frequently write merge instead of merge_t and merge_s.

Example 2.3.3. Let $S_1 = \forall x Px +^a Pa \vdash$, $S_2 = \forall x Px +^b Pb \vdash Qa$ and $S_3 = \vdash \exists y Qy$ be s-expansion trees. Then

$$\mathrm{merge}_s(S_1, S_2, S_3) = \forall x Px +^a Pa +^b Pb \vdash Qa, \exists y Qy.$$

The extraction of expansion proofs from **LK**-proofs requires, as the extraction of Herbrand sequents, quantifier-free cuts.

Definition 2.3.10 (extraction of s-expansion trees from **LK**-proofs)**.** We define a transformation ET which maps skolemized proofs with quantifier-free cuts in **LK** to s-expansion trees. We define the transformation inductively (on the number of inferences in the proof) but the rules for $\neg_l, \neg_r, \vee_l, \vee_{r_1}, \vee_{r_2}$ are omitted, the transformation of the these rules being obvious.

Base case: φ is an axiom. Then φ is of the form $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ for atoms A_i, B_j and so $\mathrm{ET}(\varphi) = \varphi$.

If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ A, B, \Gamma \vdash \Delta\end{matrix}}{A \wedge B, \Gamma \vdash \Delta} \wedge_l$$

and $\mathrm{ET}(\pi) = A^*, B^*, \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}(\varphi) = A^* \wedge B^*, \Gamma^* \vdash \Delta^*$.

If $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A & \Gamma_2 \vdash \Delta_2, B\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B} \wedge_r$$

and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\mathrm{ET}(\pi_2) = \Gamma_2^* \vdash \Delta_2^*, B^*$, then $\mathrm{ET}(\varphi) = \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A^* \wedge B^*$.

If $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_1)\\ \Gamma_1 \vdash \Delta_1, A & B, \Gamma_2 \vdash \Delta_2\end{matrix}}{A \to B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \to_l$$

and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\mathrm{ET}(\pi_2) = B^*, \Gamma_2^* \vdash \Delta_2^*$, then $\mathrm{ET}(\varphi) = A^* \to B^*, \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*$.

If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ A, \Gamma \vdash \Delta, B\end{matrix}}{\Gamma \vdash \Delta, A \to B} \to_r$$

and $\mathrm{ET}(\pi) = A^*, \Gamma^* \vdash \Delta^*, B^*$, then $\mathrm{ET}(\varphi) = \Gamma^* \vdash \Delta^*, A^* \to B^*$.

If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ A\{x \leftarrow t\}, \Gamma \vdash \Delta\end{matrix}}{\forall x A, \Gamma \vdash \Delta}\, \forall_l$$

and $\mathrm{ET}(\pi) = A\{x \leftarrow t\}^*, \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}(\varphi) = \forall x A +^t A\{x \leftarrow t\}^*, \Gamma^* \vdash \Delta^*$.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ \Gamma \vdash \Delta, A\{x \leftarrow t\}\end{matrix}}{\Gamma \vdash \Delta, \exists x A}\, \exists_r$$

and $\mathrm{ET}(\pi) = \Gamma^* \vdash \Delta^*, A\{x \leftarrow t\}^*$, then $\mathrm{ET}(\varphi) = \Gamma^* \vdash \Delta^*, \exists x A +^t A\{x \leftarrow t\}^*$.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ \Gamma \vdash \Delta\end{matrix}}{A, \Gamma \vdash \Delta}\, w_l$$

and $\mathrm{ET}(\pi) = \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}(\varphi) = A, \Gamma^* \vdash \Delta^*$. Similarly for w_r.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ A, A, \Gamma \vdash \Delta\end{matrix}}{A, \Gamma \vdash \Delta}\, c_l$$

and $\mathrm{ET}(\pi) = A_1^*, A_2^*, \Gamma^* \vdash \Delta^*$ then $\mathrm{ET}(\varphi) = \mathrm{merge}(A_1^*, A_2^*), \Gamma^* \vdash \Delta^*$. Similarly for c_r.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A^m & A^n, \Gamma_2 \vdash \Delta_2\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut(A, m, n)$$

where A^m and A^n are arbitrary quantifier-free formulas and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^m$ and $\mathrm{ET}(\pi_1) = A^n, \Gamma_2^* \vdash \Delta_2^*$; then $\mathrm{ET}(\varphi) = \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*$.

Proposition 2.3.1. *The transformation* ET *is sound: if φ is a skolemized proof with quantifier-free cuts in* **LK** *then* $\mathrm{ET}(\varphi)$ *is an s-expansion proof.*

Proof. We proceed by induction on the number of inferences in φ. We consider the cases of axioms (represented by an axiom set $\mathscr{A}$), $\wedge_r$ and *cut*; the other cases are analogous.

- (axiom) Let S be an axiom sequent in $\mathscr{A}$. Then $S = A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ for atoms A_i, B_j. Therefore for $F_S \colon \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots B_m$ we have $Sh(F_S) = Dp(F_S) = F_S$ and $\mathscr{A} \models Dp(F_S)$.

- $(\wedge_r)$ $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A & \Gamma_2 \vdash \Delta_2, B\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B}\wedge_r$$

  and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\mathrm{ET}(\pi_2) = \Gamma_2^* \vdash \Delta_2^*, B^*$ are s-expansion-proofs. Therefore, $\neg\Gamma_1^* \vee \Delta_1^* \vee A^*$ and $\neg\Gamma_2^* \vee \Delta_2^* \vee B^*$ are expansion proofs and $\mathscr{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee A^*)$ and $\mathscr{A} \models Dp(\neg\Gamma_2^* \vee \Delta_2^* \vee B^*)$. But then $\mathscr{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee (A^* \wedge B^*))$ and $\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee (A^* \wedge B^*)$ is an expansion proof. Therefore $\Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A^* \wedge B^*$ $(= \mathrm{ET}(\varphi))$ is an s-expansion-proof.

- (cut) $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A^m & A^n, \Gamma_2 \vdash \Delta_2\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut(A, m, n)$$

  where $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^m$ and $\mathrm{ET}(\pi_1) = A^n, \Gamma_2^* \vdash \Delta_2^*$ are s-expansion-proofs. Therefore, $\neg\Gamma_1^* \vee \Delta_1^* \vee A^m$ and $\neg A^n \vee \neg\Gamma_2^* \vee \Delta_2^*$ are expansion proofs and $\mathscr{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee A^m)$ and $\mathscr{A} \models Dp(\neg A^n \vee \neg\Gamma_2^* \vee \Delta_2^*)$. But then $\mathscr{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^*)$ and $\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^*$ is an expansion proof. Therefore $\Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*$ $(= \mathrm{ET}(\varphi))$ is an s-expansion-proof.

□

Example 2.3.4. Consider the **LK**-proof φ from Example 2.2.1. Its s-expansion proof, after contraction on the right side of the sequent, is

$$\forall x(P(x) \to Q(x)) +^a (P(a) \to Q(a)) \vdash \exists y(P(a) \to Q(y)) +^a (P(a) \to Q(a)).$$

Note that in case of a prenex end-sequent S an expansion proof corresponds to the derivation of S from the mid-sequent (Herbrand sequent). In fact, the function Dp of an expansion proof corresponds to a Herbrand sequent. Moreover, every expansion proof defines a set of Herbrand instances. In analogy to Definition 2.2.6 we define the set of Herbrand instances from an expansion proof.

Proposition 2.3.2 (Herbrand instances from expansion proofs)**.** *Let π be a cut-free* **LK***-derivation of a skolemized end-sequent S and let* $\mathrm{ET}(\pi)$ *be its expansion proof. Let $F \colon \forall x_1 \ldots \forall x_\beta . F'(x_1, \ldots, x_\beta)$ for a quantifier-free F' be a quantified formula in S and*

F^* its corresponding expansion tree in $\mathrm{ET}(\pi)$*. Then the full expansion* $Dp(F^*)$ *of* F^* *is a quantifier-free formula* $F'(t_1^1, \ldots, t_\beta^1) \wedge \ldots \wedge F'(t_1^\alpha, \ldots, t_\beta^\alpha)$*. We obtain a set of Herbrand instances for* F

$$S(\pi, F) = \{\{x_1 \leftarrow t_1^i, \ldots, x_\beta \leftarrow t_\beta^i\} \mid 1 \leq i \leq \alpha\}.$$

Similarly, let $F_1, \ldots, F_n$ *be all quantified formulas in* S*. Then*

$$S(\pi) = \{S(\pi, F_i) \mid i = 1, \ldots, n\}$$

is the set of Herbrand instances of π*.*

CHAPTER 3

Herbrand's Theorem in First-Order Logic

Herbrand sequents can be constructed from formalized mathematical proofs using several methods and one interesting approach is based on *Hilbert's ε-calculus*, which is also the oldest framework for proof theory [HB39]. The *extended first ε-theorem* [HB39, MZ06] eliminates algorithmically the so-called critical formulas from an ε-proof which yields a Herbrand disjunction. While this approach has several advantages, the ε-calculus has never become popular in computational proof theory of first-order logic. The main reasons are the untractability of almost all nonclassical logics by any adaptation of the ε-formalism and the clumsiness of the ε-formalism itself. In [BLL18] the second problem was addressed and a translation of **LK**-proofs with cuts to the ε-calculus was developed, obtaining a sequent-calculus based formulation of the extended first ε-theorem. The translation leads to a simplified notation and a new elimination order of the critical formulas. It is shown that a non-elementary speed-up of the length of the obtained Herbrand sequents w.r.t. the length of the Herbrand sequents obtained by the original elimination order can be achieved. While this investigation lead to an interesting approach to the extraction of Herbrand sequents, it can only be used for a specific class of formalized proofs and the extension to proofs with induction rules is not possible. For this reason we will not go into the details of this work here and refer the reader to [BLL18].

A more widespread approach to construct Herbrand sequents is based on *Gentzen's midsequent theorem* [Gen35]. Gentzen proved that from a cut-free proof a so-called midsequent, separating the proof in a propositional part and a part containing quantifiers, can be constructed. The mid-sequent itself is a propositionally valid sequent and corresponds to the Herbrand sequent. Gentzen's midsequent theorem requires cut-free proofs, however as demonstrated in Chapter 2 it is also possible to extract Herbrand sequents directly from proofs of prenex end-sequents that contain cuts. The only restriction is that the cut-formulas need to be quantifier-free. Hence, what we need is a

method to eliminate quantified cuts in order to extract Herbrand sequents. To this aim we will use the method CERES, a cut-elimination method which was specifically designed for automated proof analysis of mathematical proofs. CERES substantially differs from the traditional reductive cut-elimination methods à la Gentzen, where cuts are eliminated by stepwise reduction of cut-complexity. The reductive methods are local in the sense that only a small part of the whole proof is analyzed, namely the derivation corresponding to the introduction of the outermost logical operator. As a consequence, many types of redundancy in proofs are left undetected, leading to an unfortunate computational behavior. In contrast, the method CERES is based on a structural analysis of the whole proof, analyzing all cut-derivations simultaneously. After an application of CERES a so-called CERES *normal form*, i.e. a proof with at most atomic cuts (ACNF for atomic cut normal form), is obtained. Herbrand sequents can be extracted directly from this normal form.

In this chapter we will first introduce the CERES method and then demonstrate that not only proofs with quantifier-free cuts (i.e., proofs in normal form) suffice for Herbrand sequent extraction, but that in fact *no normal form* needs to be constructed in order to obtain this explicit information from proofs. This observation considerably simplifies the method CERES for proof analysis and results in a more efficient way of Herbrand sequent extraction. Moreover, as CERES can be extended to a formalization of induction, the developed method paves the way for the analysis of proofs with induction inferences.

3.1 The Method CERES

In this section we will present the method for proof analysis with the cut-elimination method CERES for first-order logic. The developed method will serve as a foundation for the schematic CERES method in Section 5.1.

A major motivation for CERES was the development of an *efficient* cut-elimination method for proof analysis. Indeed, in [BL06] it was shown that CERES outperforms reductive methods of cut-elimination (à la Gentzen or Tait) in computational complexity: there are infinite sequences of proofs where the computing time of CERES is nonelementarily faster than that of the reductive methods; on the other hand a nonelementary speed-up of CERES via reductive methods is shown impossible.

The CERES method we use in this work differs from the original CERES method developed in [BL00] (which we will refer to as clausal CERES) by constructing a *characteristic formula* and a single *proof projection*, both introduced in [LPW17]. In [BL00] a *characteristic clause set* and proof projections to the clauses from this set were constructed. We chose the novel method from [LPW17] for practical reasons. In particular when moving from first-order logic to a schematic language (as will be done in Section 5.1), handling of clause sets and hence of schematic sets of clauses becomes problematic. It should also be noted that in the setting of first-order logic, both CERES methods can be used. In fact, we can simulate the clausal CERES method from [BL00] with the method developed

in [LPW17]. We will introduce the basic concepts from clausal CERES, which differ from those in our CERES method, in Section 3.5.1 and demonstrate the simulation properties.

The method CERES in first-order logic roughly works as follows: The structure of a proof φ of a skolemized end-sequent S containing cuts is encoded in an unsatisfiable formula F, which is referred to as the characteristic formula of φ. A refutation of F by resolution then can serve as a skeleton for a CERES *normal form*, a new proof of S which contains quantifier-free cuts. The corresponding proof theoretic transformation uses a so-called proof projection φ^*, which is a simple cut-free proof extracted from φ proving $S\circ \vdash F$. From now on we will only consider proofs of closed end-sequents, i.e. sequents of formulas that do not contain free variables.

At the heart of the CERES method lies the characteristic formula, which describes the cut-derivations in a proof. More precisely, the characteristic formula is a quantifier-free formula corresponding to the subproofs of an **LK**-proof ending in a cut. Intuitively, the construction of a characteristic formula consists in collecting all atomic ancestors of the cuts which occur in the axioms of the proof. The formula is formed depending on how these atoms are related via binary inferences in the proof. The definition of the characteristic formula is based on the cut-status of the formula occurrences in a proof, i.e. whether a given formula occurrence is a cut-ancestor or not. In the definition below we will use a set Ω describing the set of all cut-ancestors in a given proof φ. Moreover, from φ a cut-free **LK**-proof of a sequent composed of the original end-sequent of the proof and the characteristic formula can be constructed. This proof is referred to as the *proof projection*.

Definition 3.1.1 (characteristic formula and proof projection, [LPW17])**.** Let φ be a skolemized **LK**-proof and let Ω be the set of all occurrences of cut formulas in φ. We define a formula F_ν and proof $\varphi^*(\nu)$ inductively for all nodes ν in φ. With $S(\nu, \Omega)$ we denote the intersection of $Seq(\nu)$ and Ω.

- Let ν be a leaf node in φ. Then $Seq(\nu) = A \vdash A$ for some atom A. We distinguish 3 cases:

  1. $S(\nu, \Omega) = \vdash$.
     Then $F_\nu = \bot$ and $\varphi^*(\nu) = \nu\colon A \vdash A$.
  2. $S(\nu, \Omega) = \vdash A$.
     Then $F_\nu = A$ and $\varphi^*(\nu) = \nu\colon A \vdash A$.
  3. $S(\nu, \Omega) = A \vdash$.
     Then $F_\nu = \neg A$ and $\varphi^*(\nu) =$.

$$\frac{A \vdash A}{\vdash A, \neg A}\ \neg\colon r$$

4. $S(\nu, \Omega) = A \vdash A$.
   Then $F_\nu = A \vee \neg A$ and $\varphi^*(\nu) =$.

$$\dfrac{\dfrac{A \vdash A}{\vdash A, \neg A}\,\neg : r}{\vdash A \vee \neg A}\,\vee : r$$

- Now let us assume that ξ is a unary inference in π with premise ν' and conclusion ν in φ. Inductively we assume that $F_{\nu'}$ and $\varphi^*(\nu')$ have been defined and $\varphi^*(\nu')$ is a proof of $\Gamma \vdash \Delta, F_{\nu'}$ where $\Gamma \vdash \Delta = S(\nu', \Omega)$.

  We define $F_\nu = F_{\nu'}$. For defining $\varphi^*(\nu)$ we distinguish two cases:

  1. the principal formula of ξ is an ancestor of Ω then $\varphi^*(\nu) = \varphi^*(\nu')$.
  2. The principal formula of ξ is an ancestor of the end-sequent. Then $\varphi^*(\nu) =$

$$\dfrac{\begin{matrix}(\varphi^*(\nu'))\\ \Gamma \vdash \Delta, F_{\nu'}\end{matrix}}{\Gamma' \vdash \Delta', F_{\nu'}}\,\xi$$

- Assume that ξ is a binary inference with premises ν_1, ν_2 and conclusion ν. Assume further that F_{ν_1}, F_{ν_2} are defined and $\varphi^*(\nu_i)$ are derivations of $\Gamma_i \vdash \Delta_i, F_{\nu_i}$ (for $i = 1, 2$) such that $\Gamma_i \vdash \Delta_i = S(\nu_i, \Omega)$. We distinguish two cases:

  1. the auxiliary formulas in ξ are ancestors of Ω. Then we define $\varphi^*(\nu) =$

$$\dfrac{\begin{matrix}(\varphi^*(\nu_i)) & (\varphi^*(\nu_2))\\ \Gamma_1 \vdash \Delta_1, F_{\nu_1} & \Gamma_2 \vdash \Delta_2, F_{\nu_2}\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, F_{\nu_1} \wedge F_{\nu_2}}\,\wedge : r$$

     and $F_\nu = F_{\nu_1} \wedge F_{\nu_2}$.
  2. the auxiliary formulas in ξ are ancestors of the end-sequent. Then we define $\varphi^*(\nu) =$

$$\dfrac{\dfrac{\begin{matrix}(\varphi^*(\nu_i)) & (\varphi^*(\nu_2))\\ \Gamma_1 \vdash \Delta_1, F_{\nu_1} & \Gamma_2 \vdash \Delta_2, F_{\nu_2}\end{matrix}}{\Gamma'_1, \Gamma'_2 \vdash \Delta'_1, \Delta'_2, F_{\nu_1}, F_{\nu_2}}\,\xi}{\Gamma'_1, \Gamma'_2 \vdash \Delta'_1, \Delta'_2, F_{\nu_1} \vee F_{\nu_2}}\,\vee : r$$

     and $F_\nu = F_{\nu_1} \vee F_{\nu_2}$.

Finally, the characteristic formula of φ is defined as $F = F_{\nu_0}$ and the proof projection as $\varphi^* = \varphi^*(\nu_0)$ for the root node (end-sequent) ν_0 of φ.

Example 3.1.1. Let φ be the following **LK**-proof:

$$\dfrac{\begin{matrix}(\varphi_1) & (\varphi_2)\\ \forall x(Px \to Pfx) \vdash \forall x(Px \to Pf^2x) & Pa, \forall x(Px \to Pf^2x) \vdash \exists z Pf^4z\end{matrix}}{Pa, \forall x(Px \to Pfx) \vdash \exists z Pf^4z}\,cut$$

where $\varphi_1 =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P\alpha \vdash P\alpha} \quad \dfrac{}{Pf\alpha \vdash Pf\alpha} \quad \dfrac{}{Pf^2\alpha \vdash Pf^2\alpha}\,\to_l}{Pf\alpha, Pf\alpha \to Pf^2\alpha \vdash Pf^2\alpha}\,\to_l}{P\alpha, P\alpha \to Pf\alpha, Pf\alpha \to Pf^2\alpha \vdash Pf^2\alpha}\,\forall_l, \forall_l}{P\alpha, \forall x(Px \to Pfx), \forall x(Px \to Pfx) \vdash Pf^2\alpha}\,\to_r}{\forall x(Px \to Pfx), \forall x(Px \to Pfx) \vdash P\alpha \to Pf^2\alpha}\,c_l}{\forall x(Px \to Pfx) \vdash P\alpha \to Pf^2\alpha}\,\forall_r}{\forall x(Px \to Pfx) \vdash \forall x(Px \to Pf^2x)}$$

and $\varphi_2 =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pa \vdash Pa} \quad \dfrac{}{Pf^2a \vdash Pf^2a} \quad \dfrac{}{Pf^4a \vdash Pf^4a}\,\to_l}{Pf^2a, Pf^2a \to Pf^4a \vdash Pf^4a}\,\to_l}{Pa, Pa \to Pf^2a, Pf^2a \to Pf^4a \vdash Pf^4a}\,\forall_l, \forall_l}{Pa, \forall x(Px \to Pf^2x), \forall x(Px \to Pf^2x) \vdash Pf^4a}\,c_l}{Pa, \forall x(Px \to Pf^2x) \vdash Pf^4a}\,\exists_r}{Pa, \forall x(Px \to Pf^2x) \vdash \exists z Pf^4z}$$

Following Definition 3.1.1 we obtain the proof projection $\pi^* =$

$$\dfrac{(\pi_1^*) \qquad (\pi_2^*)}{Pa, \forall x(Px \to Pfx) \vdash \exists z Pf^4z, Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)}\,\wedge : r$$

where $\pi_1^* =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P\alpha \vdash P\alpha}}{\vdash P\alpha, \neg P\alpha}\,\neg : r \quad \dfrac{\dfrac{}{Pf\alpha \vdash Pf\alpha} \quad \dfrac{}{Pf^2\alpha \vdash Pf^2\alpha}\,\to : l}{Pf\alpha, Pf\alpha \to Pf^2\alpha \vdash Pf^2\alpha}\,\to : l}{P\alpha \to Pf\alpha, Pf\alpha \to Pf^2\alpha \vdash \neg P\alpha, Pf^2\alpha}\,\vee : r}{P\alpha \to Pf\alpha, Pf\alpha \to Pf^2\alpha \vdash \neg P\alpha \vee Pf^2\alpha}\,\forall_l, \forall_l}{\forall x(Px \to Pfx), \forall x(Px \to Pfx) \vdash \neg P\alpha \vee Pf^2\alpha}\,c_l}{\forall x(Px \to Pfx) \vdash \neg P\alpha \vee Pf^2\alpha}$$

and $\pi_2^* =$

$$\dfrac{\dfrac{\dfrac{\dfrac{Pf^2a \vdash Pf^2a}{\vdash Pf^2a, \neg Pf^2a}\neg : r}{\vdash Pf^2a \vee \neg Pf^2a}\vee : r \qquad \dfrac{Pf^4a \vdash Pf^4a}{\vdash Pf^4a, \neg Pf^4a}\neg : r}{\vdash Pf^4a, (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a}\wedge : r \qquad Pa \vdash Pa}{Pa \vdash Pf^4a, Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a}\wedge : r$$
$$\dfrac{Pa \vdash Pf^4a, Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a}{Pa \vdash \exists z Pf^4z, Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a}\exists_r$$

and the characteristic formula

$$F = Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha).$$

Let μ_0 be the occurrence of the formula $\forall x(Px \to Pfx)$ and μ_1 the occurrence of the formula $\exists z Pf^4z$ in the end-sequent of π^*. Then $S(\pi^*, \mu_0) = \{\{x \leftarrow \alpha\}, \{x \leftarrow f\alpha\}\}$, $S(\pi^*, \mu_1) = \{\{z \leftarrow a\}\}$ and $S(\pi^*) = \{\{x \leftarrow \alpha\}, \{x \leftarrow f\alpha\}, \{z \leftarrow a\}\}$. The Herbrand expansion of the formula $\forall x(Px \to Pfx)$ is $H(\pi^*, \mu_0) = P\alpha \to Pf\alpha, Pf\alpha \to Pf^2\alpha$ and the Herbrand expansion of the formula $\exists z Pf^4z$ is $H(\pi^*, \mu_1) = Pf^4a$. The Herbrand sequent of π^* is $H(\pi^*) = Pa, P\alpha \to Pf\alpha, Pf\alpha \to Pf^2\alpha \vdash Pf^4a, Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)$.

It can be shown that the universal closure of a characteristic formula F is always unsatisfiable. In fact, a refutation of the characteristic formula can serve as a skeleton for a proof in normal form, which can be used for a direct extraction of Herbrand sequents. However, in general the construction of a refutation of the characteristic formula can be a tedious task and is often, especially in a schematic setting for handling proofs with induction rules, the most problematic step in the CERES method. In Section 3.2 we will introduce a novel resolution calculus to overcome these difficulties, as the calculus can be easily extended to a schematic setting. To move on with the method CERES, for now we only need to know that the universal closure of the characteristic formula is unsatisfiable.

Proposition 3.1.1 ([LPW17]). *Let φ be a skolemized* **LK***-proof and F its characteristic formula. Then $\forall F$ is unsatisfiable.*

Proof. In [BL00] a set of clauses $\mathrm{CL}(\varphi)$ (the characteristic clause set) is defined and shown unsatisfiable. It is easy to check that $\mathrm{CL}(\varphi)$ can be obtained from F (the conjuncts of F define the clauses in $\mathrm{CL}(\varphi)$, see Section 3.5.1), hence $\forall F$ is unsatisfiable as well. □

As $\forall F$ is unsatisfiable there exists a cut-free proof (or a proof with quantifier-free cuts) χ of $\forall F \vdash$. By Herbrand's theorem there exist ground substitutions $\theta_1, \ldots, \theta_\alpha$ on the variables in F s.t. $S_\alpha\colon F\sigma_1, \ldots, F\sigma_\alpha \vdash$ is a propositionally valid sequent. Therefore there exists an **LK**-proof ψ deriving S_α. This proof can be used to construct a CERES normal form.

Definition 3.1.2 (CERES normal form, [LPW17])**.** Let φ be a skolemized proof of a sequent $S\colon \Gamma \vdash \Delta$, let F be its characteristic formula and π^* the proof projection. $\forall F$ is unsatisfiable, therefore there exists a cut-free proof (or a proof with quantifier-free cuts) χ of $\forall F \vdash$. By Herbrand's theorem there exist ground substitutions $\theta_1, \ldots, \theta_\alpha$ on the variables in F s.t. $S_\alpha\colon F\theta_1, \ldots, F\theta_\alpha \vdash$ is a propositionally valid sequent. Therefore there exists an **LK**-proof ψ deriving S_α. We abbreviate $F\theta_\beta$ by F_β and $\pi^*\theta_\beta$ by π^*_β. The CERES normal form is defined as:

$$\dfrac{\dfrac{(\pi_n^*)}{\Gamma \vdash \Delta, F_\alpha} \qquad \dfrac{\dfrac{(\pi_2^*)}{\Gamma \vdash \Delta, F_2} \qquad \dfrac{\dfrac{(\pi_1^*)}{\Gamma \vdash \Delta, F_1} \qquad \dfrac{(\psi)}{F_1, \ldots, F_\alpha \vdash}\, cut}{\Gamma, F_2, \ldots, F_\alpha \vdash \Delta}\, cut + c^*}{\Gamma, F_3, \ldots, F_\alpha \vdash \Delta} \\ \vdots}{\Gamma \vdash \Delta}\, cut + c^*$$

From a CERES normal form Herbrand sequents can be extracted by collecting instances of the quantified formulas occurring in the end-sequent [HLWP08].

As mentioned before, the introduced CERES method differs from previously defined CERES methods (the clausal CERES method). The clausal CERES method can be considered as a special case of the new CERES method and we will use the clausal CERES method in Section 3.5 for defining an efficient method for Herbrand sequent extraction from proofs with equality inferences. We would like to highlight once more that the main reason for choosing the CERES method over the clausal CERES method lies in the application to inductive structures.

3.2 The Resolution Calculus RPL_0

As the universal closure $\forall F$ of a characteristic formula F is always unsatisfiable, it can be refuted by any theorem proving method. For refuting characteristic formulas we chose the novel calculus RPL_0 for quantifier-free formulas, which combines dynamic normalization rules à la Andrews [And71] with the resolution rule. In contrast to [And71] we do not restrict the resolution rule to atomic formulas. The main motivation of the calculus RPL_0 is that it can be extended to a schematic setting in a straightforward way (see Section 4.3). Moreover, RPL_0 is particularly suited for the extraction of Herbrand sequents, which is the main aim of this work. In particular, a refutation in RPL_0 is a refutation from $\vdash F$, for a quantifier-free formula F. Having constructed a RPL_0 refutation of a characteristic formula F, a Herbrand sequent of F can be obtained by a simple proof transformation.

We denote as PL_0 the set of quantifier-free formulas in predicate logic. As characteristic formulas do not contain $\rightarrow$, for simplicity we omit $\rightarrow$, but can represent it by $\neg$ and $\vee$ in the usual way. In this setting, as sequents we consider objects of the form $\Gamma \vdash \Delta$ where Γ and Δ are multisets of formulas in PL_0.

Definition 3.2.1 (RPL_0)**.** The axioms of RPL_0 are sequents $\vdash F$ for $F \in \mathrm{PL}_0$.

The rules are elimination rules for the connectives and the resolution rule.

$$\frac{\Gamma \vdash \Delta, A \wedge B}{\Gamma \vdash \Delta, A} \wedge\colon r_1 \quad \frac{\Gamma \vdash \Delta, A \wedge B}{\Gamma \vdash \Delta, B} \wedge\colon r_2 \quad \frac{A \wedge B, \Gamma \vdash \Delta}{A, B, \Gamma \vdash \Delta} \wedge\colon l$$

$$\frac{\Gamma \vdash \Delta, A \vee B}{\Gamma \vdash \Delta, A, B} \vee\colon r \quad \frac{A \vee B, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} \vee\colon l_1 \quad \frac{A \vee B, \Gamma \vdash \Delta}{B, \Gamma \vdash \Delta} \vee\colon l_2$$

$$\frac{\Gamma \vdash \Delta, \neg A}{A, \Gamma \vdash \Delta} \neg\colon r \quad \frac{\neg A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, A} \neg\colon l$$

The resolution rule where ϑ is an m.g.u. of $\{A_1, \ldots, A_k, B_1, \ldots, B_l\}$ and $V(\{A_1, \ldots, A_k\}) \cap V(\{B_1, \ldots, B_l\}) = \emptyset$ is

$$\frac{\Gamma \vdash \Delta, A_1, \ldots, A_k \quad B_1, \ldots, B_m, \Pi \vdash \Lambda}{\Gamma\vartheta, \Pi\vartheta \vdash \Delta\vartheta, \Lambda\vartheta}\ res$$

A RPL_0-derivation is a tree formed from axioms $\vdash F$ for $F \in \mathrm{PL}_0$ by application of the rules above.

Note that in a RPL_0-derivation several resolution rules may occur and hence several most general unifiers ϑ_i need to be applied. A *total unifier* (or total m.g.u.) can be obtained by considering the most general simultaneous unifier of the unification problems given by the atoms in the premises of all resolution rules, after regularizing the derivation.

Proposition 3.2.1. RPL_0 *is sound and refutationally complete, i.e.*

1. *all rules in* RPL_0 *are sound and*
2. *for any unsatisfiable formula* $\forall F$ *and* $F \in \mathrm{PL}_0$ *there exists a* RPL_0*-derivation of* $\vdash$ *from axioms of the form* $\vdash F\vartheta$ *where* ϑ *is a renaming of* $V(F)$*.*

Proof. 1. is trivial: if $\mathscr{M}$ is a model of the premise(s) of a rule then $\mathscr{M}$ is also a model of the conclusion.

For proving 2. we first derive the standard clause set $\mathscr{C}$ of F. Therefore, we apply the rules of RPL_0 to $\vdash F$, decomposing F into its subformulas, until we cannot apply any rule other than the resolution rule *res*. The last subformula obtained in this way is atomic and hence a clause. The standard clause set $\mathscr{C}$ of F is comprised of the clauses obtained in this way. As $\forall F$ is unsatisfiable, its standard clause set is refutable by resolution. Thus, we apply *res* to the clauses and obtain $\vdash$. The whole derivation lies in RPL_0. □

Example 3.2.1. Let $F = Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)$ be the characteristic formula from Example 3.1.1. Let $H = (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)$ and let $G = \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)$. We obtain the refutation $\rho =$

$$\dfrac{(\rho_1)\quad Pf^2a \vdash \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash F}{\vdash H}\wedge_{r_2}}{\vdash G}\wedge_{r_2}}{\vdash \neg P\alpha \vee Pf^2\alpha}\wedge_{r_2}}{\vdash \neg P\alpha, Pf^2\alpha}\vee_r}{P\alpha \vdash Pf^2\alpha}\neg_r}{Pa \vdash}\, res\{\alpha \leftarrow a\} \qquad \dfrac{\dfrac{\vdash F}{\vdash P(a)}\wedge_{r_1}}{\vdash Pa}\, r}{\vdash}\, res$$

where $\rho_1 =$

$$\dfrac{\dfrac{\dfrac{\vdash F}{\vdash H}\wedge_{r_2}}{\vdash G}\wedge_{r_2}}{\vdash \neg Pf^4a}\wedge_{r_1} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash F}{\vdash H}\wedge_{r_2}}{\vdash G}\wedge_{r_2}}{\vdash \neg P\alpha \vee Pf^2\alpha}\wedge_{r_2}}{\vdash \neg P\alpha, Pf^2\alpha}\vee_r}{P\alpha \vdash Pf^2\alpha}\neg_r}{Pf^2a \vdash}\, res\{\alpha \leftarrow f^2a\}$$

Grounding of ρ (by applying a total m.g.u) results in $\rho^* =$

$$\dfrac{(\rho_1^*)\quad Pf^2a \vdash \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash F\{\alpha \leftarrow a\}}{\vdash H\{\alpha \leftarrow a\}}\wedge_{r_2}}{\vdash G\{\alpha \leftarrow a\}}\wedge_{r_2}}{\vdash \neg Pa \vee Pf^2a}\wedge_{r_2}}{\vdash \neg Pa, Pf^2a}\vee_r}{Pa \vdash Pf^2a}\neg_r}{Pa \vdash}\, res \qquad \dfrac{\dfrac{\vdash F\{\alpha \leftarrow a\}}{\vdash P(a)}\wedge_{r_1}}{\vdash Pa}\, r}{\vdash}\, res$$

where $\rho_1^* =$

$$\dfrac{\dfrac{\dfrac{\vdash F\{\alpha \leftarrow f^2a\}}{\vdash H\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{\vdash G\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{\vdash \neg Pf^4a}\wedge_{r_1} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash F\{\alpha \leftarrow f^2a\}}{\vdash H\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{\vdash G\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{\vdash \neg Pf^2a \vee Pf^4a}\wedge_{r_2}}{\vdash \neg Pf^2a, Pf^4a}\vee_r}{Pf^2a \vdash Pf^4a}\neg_r}{Pf^2a \vdash}\, res$$

The set of substitutions in ρ is $\Sigma(\rho) = \{\{\alpha \leftarrow a\}, \{\alpha \leftarrow f^2a\}\}$.

The calculus RPL_0 can be used to construct a Herbrand sequent of F, where $\vdash F$ is the axiom used in the RPL_0 derivation. To obtain these Herbrand instances we have to consider the grounded RPL_0-refutation ρ of F. By construction, ρ is a RPL_0-refutation with axioms of the form $\vdash F\sigma_i$, where σ_i is a substitution obtained by grounding. Appending $F\sigma_i$ to the antecedent of these axioms, results in **LK**-axioms $F\sigma_i \vdash F\sigma_i$. When we propagate these formulas in the antecedents of our axioms to the root node, we obtain a derivation of $F\sigma_1, \dots F\sigma_n \vdash$. To obtain an **LK**-proof of $F\sigma_1, \dots F\sigma_n \vdash$ we only have to replace the resolution rules by cut rules.

Proposition 3.2.2. *Let ρ be a RPL_0-refutation of a characteristic formula F. Then there exists an* **LK***-proof ρ^* of $F\sigma_1, \dots, F\sigma_n \vdash$, where $\sigma_1, \dots, \sigma_n \in \Sigma(\rho)$ are the substitutions of the axioms $\vdash F$ in ρ after applying the total m.g.u.*

Proof. Applying the total m.g.u. of ρ to ρ is performed by regularizing ρ (i.e. renaming of variables) and then grounding the obtained derivation. After the total m.g.u. has been applied to ρ, we obtain a RPL_0-refutation ρ' from axioms $\vdash F\sigma_i$ for a substitution σ_i, where the resolution rules do not define substitutions. Therefore, we can replace the resolution rules by cut rules, hence obtaining an **LK**-refutation ρ'' from axioms $\vdash F\sigma_i$. Appending $F\sigma_i$ to the antecedent of the axioms results in **LK** axioms $F\sigma_i \vdash F\sigma_i$. Now we only have to propagate the antecedents of the axioms to the root node and we obtain an **LK**-proof ρ^* of $F\sigma_1, \dots, F\sigma_n \vdash$. Double occurrences in the end-sequent are contracted. □

Example 3.2.2. Consider the grounded refutation ρ^* from Example 3.2.1. Following the construction in the proof of Proposition 3.2.2 we obtain an **LK**-proof $\delta^* =$

$$\dfrac{\dfrac{(\delta_1^*)}{F\{\alpha \leftarrow a\}, F\{\alpha \leftarrow f^2a\}, F\{\alpha \leftarrow f^2a\}, Pa \vdash} \quad \dfrac{\dfrac{F\{\alpha \leftarrow a\} \vdash F\{\alpha \leftarrow a\}}{F\{\alpha \leftarrow a\} \vdash P(a)}\, r}{F\{\alpha \leftarrow a\} \vdash Pa}\, \wedge_{r_1}}{\dfrac{F\{\alpha \leftarrow a\}, F\{\alpha \leftarrow a\}, F\{\alpha \leftarrow f^2a\}, F\{\alpha \leftarrow f^2a\} \vdash}{F\{\alpha \leftarrow a\}, F\{\alpha \leftarrow f^2a\} \vdash}\, c_l}\, cut$$

where $\delta_1^* =$

$$\dfrac{\dfrac{(\delta_2^*)}{F\{\alpha \leftarrow f^2a\}, F\{\alpha \leftarrow f^2a\}, Pf^2a \vdash} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{F\{\alpha \leftarrow a\} \vdash F\{\alpha \leftarrow a\}}{F\{\alpha \leftarrow a\} \vdash H\{\alpha \leftarrow a\}}\, \wedge_{r_2}}{F\{\alpha \leftarrow a\} \vdash G\{\alpha \leftarrow a\}}\, \wedge_{r_2}}{F\{\alpha \leftarrow a\} \vdash \neg Pa \vee Pf^2a}\, \wedge_{r_2}}{F\{\alpha \leftarrow a\} \vdash \neg Pa, Pf^2a}\, \vee_r}{F\{\alpha \leftarrow a\}, Pa \vdash Pf^2a}\, \neg_r}{F\{\alpha \leftarrow a\}, F\{\alpha \leftarrow f^2a\}, F\{\alpha \leftarrow f^2a\}, Pa \vdash}\, cut$$

and $\delta_2^* =$

$$\dfrac{\dfrac{\dfrac{\dfrac{F\{\alpha \leftarrow f^2a\} \vdash F\{\alpha \leftarrow f^2a\}}{F\{\alpha \leftarrow f^2a\} \vdash H\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{F\{\alpha \leftarrow f^2a\} \vdash G\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{F\{\alpha \leftarrow f^2a\} \vdash \neg Pf^4a}\wedge_{r_1}}{F\{\alpha \leftarrow f^2a\}, Pf^4a \vdash}\neg_r \qquad \dfrac{\dfrac{\dfrac{\dfrac{F\{\alpha \leftarrow f^2a\} \vdash F\{\alpha \leftarrow f^2a\}}{F\{\alpha \leftarrow f^2a\} \vdash H\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{F\{\alpha \leftarrow f^2a\} \vdash G\{\alpha \leftarrow f^2a\}}\wedge_{r_2}}{F\{\alpha \leftarrow f^2a\} \vdash \neg Pf^2a \vee Pf^4a}\wedge_{r_2}}{F\{\alpha \leftarrow f^2a\} \vdash \neg Pf^2a, Pf^4a}\vee_r}{F\{\alpha \leftarrow f^2a\}, Pf^2a \vdash Pf^4a}\neg_r$$
$$F\{\alpha \leftarrow f^2a\}, F\{\alpha \leftarrow f^2a\}, Pf^2a \vdash \quad res$$

Now we can also give an example of a CERES normal form.

Example 3.2.3. Let φ be the **LK**-proof of Example 3.1.1. Then the CERES normal form of φ is $\pi =$

$$\dfrac{\begin{array}{cc}(\pi^*\sigma_2) & (\pi_1^*)\\ Pa, \forall x(Px \to Pfx) \vdash \exists z Pf^4z, F\sigma_2 & Pa, \forall x(Px \to Pfx), F\sigma_2 \vdash \exists z Pf^4z\end{array}}{Pa, \forall x(Px \to Pfx) \vdash \exists z Pf^4z}\ cut + c^*$$

where $\pi_1^* =$

$$\dfrac{\begin{array}{cc}(\pi^*\sigma_1) & (\delta^*)\\ Pa, \forall x(Px \to Pfx) \vdash \exists z Pf^4z, F\sigma_1 & F\sigma_1, F\sigma_2 \vdash\end{array}}{Pa, \forall x(Px \to Pfx), F\sigma_2 \vdash \exists z Pf^4z}\ cut$$

and where $\sigma_1 = \{\alpha \leftarrow a\}$, $\sigma_2 = \{\alpha \leftarrow f^2a\}$, δ^* is the **LK**-proof in Example 3.2.2, π^* is the proof projection in Example 3.1.1 and F is the characteristic formula $Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)$.

We can also extract a Herbrand sequent of π. Let μ_0 be the occurrence of the formula $\forall x(Px \to Pfx)$ and μ_1 the occurrence of the formula $\exists z Pf^4z$ in the end-sequent of π.

Then $S(\pi, \mu_0) = S(\pi^*\sigma_1, \mu_0) \cup S(\pi^*\sigma_2, \mu_0) = S(\pi^*, \mu_0)\sigma_1 \cup S(\pi^*, \mu_0)\sigma_2$.

From Example 3.1.1 we know that $S(\pi^*, \mu_0) = \{\{x \leftarrow \alpha\}, \{x \leftarrow f\alpha\}\}$. Therefore,

$$\begin{aligned} S(\pi, \mu_0) &= \{\{x \leftarrow \alpha\}, \{x \leftarrow f\alpha\}\}\sigma_1 \cup \{\{x \leftarrow \alpha\}, \{x \leftarrow f\alpha\}\}\sigma_2 \\ &= \{\{x \leftarrow a\}, \{x \leftarrow fa\}\} \cup \{\{x \leftarrow f^2a\}, \{x \leftarrow f^3a\}\}. \end{aligned}$$

$S(\pi, \mu_1) = S(\pi^*\sigma_1, \mu_1) \cup S(\pi^*\sigma_2, \mu_1) = S(\pi^*, \mu_1)\sigma_1 \cup S(\pi^*, \mu_1)\sigma_2$.

From Example 3.1.1 we know that $S(\pi^*, \mu_1) = \{\{z \leftarrow a\}\}$. Therefore,

$$S(\pi, \mu_1) = \{\{z \leftarrow a\}\}\sigma_1 \cup \{\{z \leftarrow a\}\}\sigma_2 = \{\{z \leftarrow a\}\}.$$

The Herbrand expansion of the formula $\forall x(Px \rightarrow Pfx)$ is

$$H(\pi, \mu_0) \quad = \quad Pa \rightarrow Pfa, Pfa \rightarrow Pf^2a, Pf^2a \rightarrow Pf^3a, Pf^3a \rightarrow Pf^4a$$

and the Herbrand expansion of the formula $\exists z Pf^4z$ is $H(\pi, \mu_1) = Pf^4a$.

The Herbrand sequent of π is

$$H(\pi) = Pa, Pa \rightarrow Pfa, Pfa \rightarrow Pf^2a, Pf^2a \rightarrow Pf^3a, Pf^3a \rightarrow Pf^4a \vdash Pf^4a.$$

RPL_0 can easily be extended to handling a logic with equality by adding paramodulation rules.

Definition 3.2.2 ($\mathrm{RPL}_0^{=}$)**.** The calculus $\mathrm{RPL}_0^{=}$ is RPL_0 extended by the following paramodulation rules:

We assume that the two clauses in the premises are always variable disjoint and that σ is a most general unifier of $\{s, s'\}$.

$$\frac{\Gamma \vdash \Delta, s = t \quad A[s']_\Psi, \Pi \vdash \Lambda}{A[t]_\Psi\sigma, \Gamma\sigma, \Pi\sigma \vdash \Delta\sigma, \Lambda\sigma} \qquad \frac{\Gamma \vdash \Delta, t = s \quad A[s']_\Psi, \Pi \vdash \Lambda}{A[t]_\Psi\sigma, \Gamma\sigma, \Pi\sigma \vdash \Delta\sigma, \Lambda\sigma}$$

for inference on the left side of the clauses and

$$\frac{\Gamma \vdash \Delta, s = t \quad \Pi \vdash \Lambda, A[s']_\Psi}{\Gamma\sigma, \Pi\sigma \vdash \Delta\sigma, \Lambda\sigma, A[t]_\Psi\sigma} \qquad \frac{\Gamma \vdash \Delta, t = s \quad \Pi \vdash \Lambda, A[s']_\Psi}{\Gamma\sigma, \Pi\sigma \vdash \Delta\sigma, \Lambda\sigma, A[t]_\Psi\sigma}$$

for the right side, where Ψ denotes a position of a subterm where s' is replaced by t. We call $s = t$ the *active equation* of the rules.

In Section 4.3 we will extend the calculus RPL_0 to a schematic structure.

3.3 Extraction of Herbrand Sequents from Non-Normalized Proofs

Usually, explicit information in the form of Herbrand sequents is extracted from a CERES normal form φ *after* the application of CERES. However, we show that the construction of φ for the extraction of Herbrand sequents can be omitted by proving Theorem 3.3.1: the crucial information about φ is already contained in the proof projection and the resolution refutation of F. It is possible to extract instances of quantified formulas

occurring in the end-sequent S from the proof projection φ^*. Substituting these instances with the total unifier obtained from the resolution refutation of F results in the Herbrand sequent of φ. Therefore, for computing Herbrand sequents, the construction of a CERES normal form is superfluous. This simplification of the method for Herbrand sequent extraction with CERES can only be performed because of the specific structure of the method itself. Compared to the clausal CERES method for the extraction of Herbrand sequents, as described in [BL00], this improvement yields a gain in asymptotic complexity. In particular we can show that the new method outperforms the old one. A detailed complexity analysis can be found in Section 3.5.3.

For each quantified formula occurrence μ_i in a prenex end-sequent of a skolemized **LK**-proof φ we construct a structure $\overline{H}(\varphi, \mu_i, \rho)$. This structure is constructed using specific features of CERES. First, we construct the projection π^* of φ. It is a crucial observation that for a proof φ and the corresponding proof projection π^* their end-sequents are the same, except that the end-sequent of π^* contains the characteristic formula F in the succedent. Thus, the quantified formula occurrence μ_i in the end-sequent of φ also occurs in the end-sequent of π^*. Therefore, we can extract Herbrand instances of the quantified formula occurrence μ_i in the end-sequent of π^* by collecting instances from π^*. Then we construct a RPL_0-refutation ρ of the characteristic formula F of φ, which is used to construct Herbrand instances of F. The obtained substitution instances are then used to substitute the Herbrand instances obtained from the projection. Merging these substituted Herbrand instances results in $\overline{H}(\varphi, \mu_i, \rho)$ and in fact defines the set of Herbrand instances for the quantified formula occurrence μ_i of the original **LK**-proof φ.

Definition 3.3.1. Let φ be an **LK**-proof of a skolemized prenex end-sequent with characteristic formula F, ρ a grounded RPL_0-refutation from $\vdash F$, let π^* be the projection of φ to F and for each quantified formula occurrence μ_i in the end-sequent of π^* let $S(\pi^*, \mu_i)$ be the set of Herbrand instances defined by π^*. Then we define a set

$$\overline{H}(\varphi, \mu_i, \rho) = \bigcup_{j=1}^{m} S(\pi^*, \mu_i)\sigma_j$$

for $\sigma_1, \ldots, \sigma_m \in \Sigma(\rho)$, where $\Sigma(\rho)$ is the set of substitutions obtained from ρ.

Example 3.3.1. Let π be the CERES normal form in Example 3.2.3 and let μ_0 be the occurrence of the formula $\forall x(Px \to Pfx)$ and μ_1 the occurrence of the formula $\exists z P f^4 z$ in the end-sequent of π. $\Sigma(\rho) = \{\{\alpha \leftarrow a\}, \{\alpha \leftarrow f^2 a\}\}$ as in Example 3.2.1.

$S(\pi^*, \mu_0) = \{\{x \leftarrow \alpha\}, \{x \leftarrow f\alpha\}\}$ and $S(\pi^*, \mu_1) = \{\{z \leftarrow a\}\}$ as in Example 3.1.1.

Then,

$$\begin{aligned}
\overline{H}(\varphi, \mu_0, \rho) &= S(\pi^*, \mu_0)\{\alpha \leftarrow a\} \cup S(\pi^*, \mu_0)\{\alpha \leftarrow f^2 a\} \\
&= \{\{x \leftarrow a\}, \{x \leftarrow fa\}\} \cup \{\{x \leftarrow f^2 a\}, \{x \leftarrow f^3 a\}\}, \\
&= \{\{x \leftarrow a\}, \{x \leftarrow fa\}, \{x \leftarrow f^2 a\}, \{x \leftarrow f^3 a\}\}, \\
\overline{H}(\varphi, \mu_1, \rho) &= S(\pi^*, \mu_1)\{\alpha \leftarrow a\} \cup S(\pi^*, \mu_1)\{\alpha \leftarrow f^2 a\} \\
&= \{\{z \leftarrow a\}\} \cup \{\{z \leftarrow a\}\} = \{\{z \leftarrow a\}\}.
\end{aligned}$$

The obtained sets $\overline{H}(\varphi, \mu_0, \rho)$ and $\overline{H}(\varphi, \mu_1, \rho)$ in the example above are equal to the sets $S(\pi, \mu_0)$ and $S(\pi, \mu_1)$ obtained in Example 3.2.3 from the CERES normal form. This is however no coincidence but rather a generally valid observation. Indeed, it can be shown that for any quantified formula occurrence μ_i the set $\overline{H}(\varphi, \mu_i, \rho)$ is equal to the set of Herbrand instances of the quantified formula occurrence μ_i in the end-sequent of the CERES normal form π.

Theorem 3.3.1. *Let φ be an* **LK***-proof of a skolemized prenex end-sequent $\Gamma \vdash \Delta$, let F be its characteristic formula and π^* the projection of φ to F. Let ρ be a grounded* RPL_0*-refutation from $\vdash F$ and π the* CERES *normal form of φ. Then for each quantified formula occurrence μ_i in φ, $\overline{H}(\varphi, \mu_i, \rho) = S(\pi, \mu_i)$.*

Proof. By induction on the number of substitutions in $\Sigma(\rho)$. Base case: assume there is only one substitution $\sigma \in \Sigma(\rho)$. Then the CERES normal form π is of the form

$$\dfrac{\dfrac{\pi^*\sigma}{\Gamma \vdash \Delta, F\sigma} \quad \dfrac{\rho^*}{F\sigma \vdash}}{\Gamma \vdash \Delta}\,(cut)$$

Note that ρ^* is the **LK**-proof of the Herbrand instances of F, in this case of $F\sigma \vdash$, obtained from ρ.

$S(\pi, \mu_i)$ is constructed by collecting instances of the formula at occurrence μ_i. By construction of π, the quantified formula occurrence μ_i in the end-sequent of π occurs also in the sequent $\Gamma \vdash \Delta, F\sigma$. Therefore,

$$S(\pi, \mu_i) = S(\pi^*\sigma, \mu_i) = S(\pi^*, \mu_i)\sigma = \overline{H}(\varphi, \mu_i, \rho).$$

Note that the formula at occurrence μ_i is a closed formula and does not change after substitution with one of the substitutions in $\Sigma(\rho)$.

Assume by IH that for n substitutions $\sigma_1, \ldots, \sigma_n \in \Sigma(\rho)$ the assumption holds, i.e. $S(\pi, \mu_i) = \{S(\pi_1^*, \mu_i), \ldots, S(\pi_n^*, \mu_i)\} = \bigcup_{j=1}^{n} S(\pi^*, \mu_i)\sigma_j = \overline{H}(\varphi, \mu_i, \rho)$, where $\pi_i^* = \pi^*\sigma_i$ and π is of the form

$$\dfrac{\dfrac{\pi_n^*}{\Gamma \vdash \Delta, F_n} \quad \dfrac{\dfrac{\dfrac{\pi_2^*}{\Gamma \vdash \Delta, F_2} \quad \dfrac{\dfrac{\pi_1^*}{\Gamma \vdash \Delta, F_1} \quad \dfrac{\rho^*}{F_1, \ldots, F_n \vdash}\,(cut)}{\Gamma, F_2, \ldots, F_n \vdash \Delta}\,(cut + c^*)}{\Gamma, F_3, \ldots, F_n \vdash \Delta} \quad \vdots}{\Gamma \vdash \Delta}\,(cut + c^*)$$

and $F_i = F\sigma_i$.

Now consider $n+1$ substitutions $\sigma_1, \ldots, \sigma_n, \sigma_{n+1} \in \Sigma(\rho)$. Here we have to consider a proof ρ^* of $F_1, \ldots, F_n, F_{n+1} \vdash$ and the additional substitution instance π_{n+1}^* of the projection π^*:

$$\dfrac{\dfrac{\dfrac{\pi_1^*}{\Gamma \vdash \Delta, F_1} \quad \dfrac{\rho^*}{F_1, \ldots, F_n, F_{n+1} \vdash}}{\Gamma, F_2, \ldots, F_n, F_{n+1} \vdash \Delta}\,(cut) \quad \vdots}{\dfrac{\dfrac{\pi_{n+1}^*}{\Gamma \vdash \Delta, F_{n+1}} \quad \dfrac{\dfrac{\pi_n^*}{\Gamma \vdash \Delta, F_n} \quad \Gamma, F_n, F_{n+1} \vdash \Delta}{\nu\colon \Gamma, F_{n+1} \vdash \Delta}\,(cut + c^*)}{\Gamma \vdash \Delta}\,(cut + c^*)}$$

At node ν we still have only n instances of the proof projection π^*, only that $F_{n+1} \vdash$ is propagated through the proof tree starting from ρ^*. By IH the Herbrand instances for μ_i at node ν are

$$S(\pi_\nu, \mu_i) \quad = \quad \{S(\pi_1^*, \mu_i), \ldots, S(\pi_n^*, \mu_i)\} = \bigcup_{j=1}^{n} S(\pi^*, \mu_i)\sigma_j.$$

The Herbrand instances $S(\pi, \mu_i)$ are by construction

$$\begin{aligned} S(\pi, \mu_i) &= S(\pi_{n+1}^*, \mu_i) \cup S(\pi_\nu, \mu_i) \\ &= S(\pi_{n+1}^*, \mu_i) \cup \bigcup_{j=1}^{n} S(\pi^*, \mu_i)\sigma_j \\ &= S(\pi^*, \mu_i)\sigma_{n+1} \cup \bigcup_{j=1}^{n} S(\pi^*, \mu_i)\sigma_j \\ &= \bigcup_{j=1}^{n+1} S(\pi^*, \mu_i)\sigma_j = \overline{H}(\varphi, \mu_i, \rho) \end{aligned}$$

□

A crucial observation in the proof above is to notice that the quantified formula occurrences μ_i in the end-sequent of the original proof φ are the same as in the CERES normal form π and in the proof projection π^*! Moreover, the end-sequent of the projection is the end-sequent of π concatenated with the characteristic formula F and all logical rules in π take place in the instantiations of the projection. Therefore, all the rules that operate on quantified formulas occurring in the end-sequent of π are in the sub-derivations that are instantiations of the projection.

3.4 Extraction of Expansion Proofs from Non-Normalized Proofs

Also the extraction of expansion proofs is usually performed after the construction of a CERES normal form. However, as in Section 3.3, we can show that the construction of the proof projection and the resolution refutation of the characteristic formula F suffice.

By the structure of a CERES normal form, its expansion proof can be composed of the expansion proof of the projection after removal of the expansion tree of F and then substituting with the substitutions given by regularizing and grounding the refutation of F. Merging these substituted expansion proofs results in the expansion proof of the CERES normal form. Below we define an expansion tree $\overline{E}(\varphi, \rho)$ which is defined by merging the expansion trees of the instantiated proof projection after removal of the expansion tree of F. Note that the characteristic formula F is a quantifier-free formula, hence its expansion tree is a formula which we can remove easily from any expansion tree. Moreover note that this formula in fact has to be removed from the final structure, as in the CERES normal form the characteristic formula does not occur in the end-sequent, but only as a cut-formula. Thus, the expansion proof of the CERES normal form cannot contain F. This is the reason why we remove F from the expansion tree of the proof projection immediately.

Definition 3.4.1. Let φ be an **LK**-proof of a skolemized and normalized end-sequent, π^* its projection, F the characteristic formula and ρ the grounded refutation of F. We define

$$\overline{E}(\varphi, \rho) = \text{merge}((\text{ET}(\pi^*)\setminus \vdash F)\sigma_1, \ldots, (\text{ET}(\pi^*)\setminus \vdash F)\sigma_k),$$

where $\sigma_1, \ldots, \sigma_k \in \Sigma(\rho)$.

To prove that $\overline{E}(\varphi, \rho)$ is actually the expansion proof of the CERES normal form π of φ, we first need to show that for every subproof of the CERES normal form, the expansion proof can be composed of the expansion proofs of the instantiated proof projections occurring in the subproof, after removal of all obsolete instances of the characteristic formula.

Theorem 3.4.1. *Let φ be an* **LK***-proof of a skolemized and normalized end-sequent, π^* its proof projection, F its characteristic formula and ρ the grounded refutation of F. Let π be the* CERES *normal form of φ containing n cuts and $\sigma_1, \ldots, \sigma_n \in \Sigma(\rho)$. Then for every subproof π' of π containing k cuts,*

$$\begin{aligned} \text{ET}(\pi') \quad = \quad & \text{merge}(\text{merge}(\text{ET}(\pi^*)\setminus \vdash F)\sigma_1, \ldots, \text{ET}(\pi^*)\setminus \vdash F)\sigma_k), \\ & \{F_1, \ldots, F_n \vdash\}\setminus\{F_1, \ldots, F_k \vdash\}), \end{aligned}$$

where $F_i = F\sigma_i$.

Proof. By induction on the number of cuts in π'. Base case: assume there is only one cut. Then π' is of the form

$$\dfrac{\dfrac{\pi^*\sigma_1}{\Gamma \vdash \Delta, F_1} \quad \dfrac{\rho^*}{F_1, F_2, \ldots, F_n \vdash}}{\Gamma, F_2, \ldots, F_n \vdash \Delta}\,(cut)$$

where ρ^* is the **LK**-proof obtained from ρ following Proposition 3.2.2. Let Γ^* be the expansion tree of Γ and Δ^* the expansion tree of Δ. By construction

$$\begin{aligned}\mathrm{ET}(\pi') &= \Gamma^*, F_2, \ldots, F_n \vdash \Delta^* \\ &= \mathrm{merge}(\mathrm{ET}(\pi^*\sigma_1)\backslash \vdash F_1, \{F_1, \ldots, F_n \vdash\}\backslash\{F_1 \vdash\}) \\ &= \mathrm{merge}((\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_1, \{F_1, \ldots, F_n \vdash\}\backslash\{F_1 \vdash\}).\end{aligned}$$

Assume by IH that for k cuts in π' the assumption holds, i.e.

$$\begin{aligned}\mathrm{ET}(\pi') &= \mathrm{merge}(\mathrm{merge}(\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_1, \ldots, \mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_k), \\ &\quad \{F_1, \ldots, F_n \vdash\}\backslash\{F_1, \ldots, F_k \vdash\}),\end{aligned}$$

where $F_i = F\sigma_i$ and for $\sigma_1, \ldots, \sigma_n \in \Sigma(\rho)$.

Now consider $k+1$ cuts in π'. Here we have to consider one additional substitution instance π^*_{k+1} of the projection π^* in the construction of π'.

$$\dfrac{\dfrac{\dfrac{\pi_1^*}{\Gamma \vdash \Delta, F_1} \quad \dfrac{\rho^*}{F_1, \ldots, F_n \vdash}}{\Gamma, F_2, \ldots, F_n \vdash \Delta}\,(cut) \qquad \vdots}{\dfrac{\dfrac{\pi^*_{k+1}}{\Gamma \vdash \Delta, F_{k+1}} \quad \nu\colon \Gamma, F_{k+1}, \ldots, F_n \vdash \Delta}{\Gamma, F_{k+2}, \ldots, F_n \vdash \Delta}\,(cut + c^*)}$$

At node ν we still have only k cuts in the subproof π'_ν. By IH the expansion proof at node ν is

$$\begin{aligned}\mathrm{ET}(\pi'_\nu) &= \mathrm{merge}(\mathrm{merge}((\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_1, \ldots, \mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_k), \\ &\quad \{F_1, \ldots, F_n \vdash\}\backslash\{F_1, \ldots, F_k \vdash\})\end{aligned}$$

The expansion proof of π' is by construction

$$\begin{aligned}\mathrm{ET}(\pi') &= \mathrm{merge}(\mathrm{ET}(\pi^*_{k+1})\backslash\{\vdash F_{k+1}\}, \mathrm{ET}(\pi_\nu)\backslash\{F_{k+1} \vdash\}) \\ &= \mathrm{merge}(\mathrm{ET}(\pi^*\sigma_{k+1})\backslash\{\vdash F\sigma_{k+1}\}, \mathrm{ET}(\pi_\nu)\backslash\{F_{k+1} \vdash\}) \\ &= \mathrm{merge}((\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_{k+1}, \mathrm{ET}(\pi_\nu)\backslash F_{k+1} \vdash) \\ &= \mathrm{merge}((\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_{k+1}, \\ &\quad \mathrm{merge}(\mathrm{merge}((\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_1, \ldots, \mathrm{ET}(\pi^*)\backslash F)\sigma_k), \\ &\quad \{F_1, \ldots, F_n \vdash\}\backslash\{F_1, \ldots, F_k \vdash\})\backslash F_{k+1} \vdash) \\ &= \mathrm{merge}(\mathrm{merge}((\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_1, \ldots, (\mathrm{ET}(\pi^*)\backslash F)\sigma_k, (\mathrm{ET}(\pi^*)\backslash F)\sigma_{k+1}), \\ &\quad \{F_1, \ldots, F_n \vdash\}\backslash\{F_1, \ldots, F_k \vdash\}\}\backslash F_{k+1} \vdash) \\ &= \mathrm{merge}(\mathrm{merge}((\mathrm{ET}(\pi^*)\backslash \vdash F)\sigma_1, \ldots, (\mathrm{ET}(\pi^*)\backslash F)\sigma_k, (\mathrm{ET}(\pi^*)\backslash F)\sigma_{k+1}), \\ &\quad \{F_1, \ldots, F_n \vdash\}\backslash\{F_1, \ldots, F_k, F_{k+1} \vdash\})\end{aligned}$$

□

Now we obtain the main result of this section.

Theorem 3.4.2. *Let φ be an* **LK***-proof of a skolemized and normalized end-sequent, π^* its proof projection, F its characteristic formula and ρ the grounded refutation of F. Let π be the* CERES *normal form of φ. Then*

$$\overline{E}(\varphi, \rho) = \mathrm{ET}(\pi).$$

Proof. By Theorem 3.4.1, take $k = n$ and consider π instead of a proper subproof of π. □

3.5 Herbrand's Theorem with Clausal CERES + Equality

Many mathematical proofs use equality rules, therefore the method for proof analysis with CERES has to take into account equality as well. In this section we develop the clausal CERES method for **LK** + equality rules, which is based on the original clausal CERES method from [BL00]. In this method we extract a characteristic clause set and construct several proof projections to these clauses. As demonstrated in Section 3.1 we will not construct a CERES normal form, but extract the crucial information from the projections and the refutation of the characteristic clause set. Moreover, we provide a complexity analysis of the new method. Note that the investigation as described in this section was published [LL19]. An application to real mathematical proofs, based on the method from this section, can be found in Section 6.1. Note that the investigation in this section is not needed in order to proceed with the construction of a proof analysis method for proofs with induction rules. In fact, the method in Chapter 5 is based only on the investigations regarding the CERES method for first-order logic used in previous sections. Nevertheless, clausal CERES as well as the corresponding proof analysis method provide some interesting insights, particularly when considering the complexity analysis compared to the old methods.

3.5.1 The Clausal CERES Method

In this section we will describe the original CERES method for first-order logic, which is based on a characteristic clause set and a corresponding set of proof projections. We will show that the CERES method from Section 3.1 simulates the clausal CERES method.

Intuitively, the clause set extraction consists in collecting all atomic ancestors of the cuts which occur in the axioms of the proof. The clauses are formed depending on how these atoms are related via binary inferences in the proof.

Definition 3.5.1 (characteristic clause-set [BL00])**.** Let φ be a proof of a skolemized sequent. The characteristic clause set is built recursively from the leaves of the proof to the end-sequent. Let ν be the occurrence of a sequent in this proof. Then:

- If ν is an axiom, then $\mathrm{CL}(\nu)$ contains the sub-sequent of ν composed only of cut-ancestors.
- If ν is the result of the application of a unary rule on a sequent μ, then $\mathrm{CL}(\nu) = \mathrm{CL}(\mu)$
- If ν is the result of the application of a binary rule on sequents μ_1 and μ_2, then we distinguish two cases:
  - If the rule is applied to ancestors of the cut formula, then $\mathrm{CL}(\nu) = \mathrm{CL}(\mu_1) \cup \mathrm{CL}(\mu_2)$
  - If the rule is applied to ancestors of the end-sequent, then $\mathrm{CL}(\nu) = \mathrm{CL}(\mu_1) \times \mathrm{CL}(\mu_2)$

where

$$\mathrm{CL}(\mu_1) \times \mathrm{CL}(\mu_2) = \{C \circ D \mid C \in \mathrm{CL}(\mu_1), D \in \mathrm{CL}(\mu_2)\}.$$

If ν_0 is the root node $\mathrm{CL}(\nu_0)$ is called the characteristic clause set of φ.

The following example demonstrates the construction of a characteristic clause set.

Example 3.5.1. The set of axioms Ax_s is defined as $Ax_s = Ax \cup \{\vdash f^2z = gz\}$. Let φ be a proof of the sequent $Pa, \forall x(Px \to Pfx) \vdash \exists z Pf^4z$:

$$\dfrac{\dfrac{}{\forall x(Px \to Pfx) \vdash \forall x(Px \to Pgx)}(\varphi_1) \qquad \dfrac{}{Pc, \forall x(Px \to Pgx) \vdash Pg^2c}(\varphi_2)}{Pc, \forall x(Px \to Pfx) \vdash Pg^2c}\, cut$$

φ_1 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pz \vdash Pz} \qquad \dfrac{\dfrac{\dfrac{}{Pf^2z \vdash Pf^2z} \qquad \dfrac{}{\vdash f^2z = gz}}{Pf^2z \vdash Pgz} =_{r_1}}{Pfz \vdash Pfz} \qquad \dfrac{}{Pf^2z \vdash Pgz}}{Pfz, Pfz \to Pf^2z \vdash Pgz} \to_l}{Pz, Pz \to Pfz, Pfz \to Pf^2z \vdash Pgz} \to_l}{Pz, \forall x(Px \to Pfx) \vdash Pgz} 2 \times \forall_l + c_l}{\forall x(Px \to Pfx) \vdash Pz \to Pgz} \to_r}{\forall x(Px \to Pfx) \vdash \forall x(Px \to Pgx)} \forall_r$$

φ_2 is

$$\dfrac{\dfrac{Pc \vdash Pc \qquad \dfrac{Pgc \vdash Pgc \qquad Pg^2c \vdash Pg^2c}{Pgc, Pgc \to Pg^2c \vdash Pg^2c}\to_l}{Pc, Pc \to Pgc, Pgc \to Pg^2c \vdash Pg^2c}\to_l}{Pc, \forall x(Px \to Pgx) \vdash Pg^2c}\, 2 \times \forall_l + c_l$$

The characteristic clause set of φ is constructed as follows:

We consider the following cut-ancestors (in axioms) in φ_1

$$\{Pz \vdash\}; \{\vdash Pf^2z\}; \{\vdash f^2z = gz\}$$

$=_{r_1}$ operates on cut-ancestors, therefore we get

$$S_1 = \{\vdash Pf^2z\} \ \cup \ \{\vdash f^2z = gz\}$$

$\to_l$ operates on end-sequent ancestors, hence

$$S = \{Pz \vdash\} \times S_1 = \{Pz \vdash Pf^2z;\ Pz \vdash f^2z = gz\}$$

We proceed analogously for the cut-ancestors in φ_2 and obtain

$$S' = \{\vdash Pc;\ Pgc \vdash Pgc;\ Pg^2c \vdash\}$$

The characteristic clause set is $S \cup S'$

$$\mathrm{CL}(\varphi) = \{Pz \vdash Pf^2z;\ \ Pz \vdash f^2z = gz;\ \ \vdash Pc;\ \ Pgc \vdash Pgc;\ \ Pg^2c \vdash\}.$$

By construction it is easy to see that the characteristic formula is composed of the clauses of the characteristic clause set. In fact, the characteristic formula is a conjunctive normal form (CNF), where the conjuncts define the clauses of the characteristic clause set.

Proposition 3.5.1. *Let φ be an* **LK***-proof, F the characteristic formula of φ and* $\mathrm{CL}(\varphi)$ *the characteristic clause set of φ. Then* $\mathrm{CL}(\varphi)$ *is derivable from F.*

Proof. $\mathrm{CL}(\varphi)$ is easily derivable from F in RPL_0: we start with the axiom $\vdash F$. F is a conjunctive normal form, thus the only rules we can apply are $\wedge_{r_1}$ or $\wedge_{r_2}$. In both cases, we decompose F into its conjuncts. We continue decomposing all the conjuncts until we reach conjunct c_i, which cannot be decomposed using $\wedge_{r_1}$ or $\wedge_{r_2}$ any more. We make a case distinction on c_i:

1. c_i is atomic. Then $c \in \mathrm{CL}(\varphi)$.
2. $c_i = \neg c_i'$. We apply the negation rule $\neg_r$ and continue with c_i'.
3. $c_i = c_1 \vee c_2$ for clauses c_1 and c_2, we apply the $\vee_r$ inference to obtain $\vdash c_1, c_2$. Depending on the structure of c_1 and c_2 we distinguish the cases

a) c_1, c_2 are atoms or negated atoms. We eliminate all possible occurrences of $\neg$ with $\neg_r$ and obtain a clause $c \in \mathrm{CL}(\varphi)$.

b) c_1 or c_2 is not atomic nor a negated atom. We continue decomposing c_1 or c_2 until we reach atoms or negated atoms.

□

Example 3.5.2. Let $F = Pa \wedge (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)$ be the characteristic formula in Example 3.1.1. We obtain the clauses of $\mathrm{CL}(\varphi)$ by:

$$\dfrac{\vdash F}{\vdash Pa}\,\wedge_{r_1}$$

$$\dfrac{\dfrac{\dfrac{\vdash F}{\vdash (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)}\,\wedge_{r_2}}{\vdash (Pf^2a \vee \neg Pf^2a)}\,\wedge_{r_1}}{\dfrac{\vdash Pf^2a, \neg Pf^2a}{Pf^2a \vdash Pf^2a}\,\neg_r}\,\vee_r$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash F}{\vdash (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)}\,\wedge_{r_2}}{\vdash \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)}\,\wedge_{r_2}}{\vdash \neg Pf^4a}\,\wedge_{r_1}}{Pf^4a \vdash}\,\neg_r$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash F}{\vdash (Pf^2a \vee \neg Pf^2a) \wedge \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)}\,\wedge_{r_2}}{\vdash \neg Pf^4a \wedge (\neg P\alpha \vee Pf^2\alpha)}\,\wedge_{r_2}}{\vdash \neg P\alpha \vee Pf^2\alpha}\,\wedge_{r_2}}{\vdash \neg P\alpha, Pf^2\alpha}\,\vee_r}{P\alpha \vdash Pf^2\alpha}\,\neg_r$$

The next step is to obtain a resolution refutation of $\mathrm{CL}(\varphi)$ using the PR-calculus.

Definition 3.5.2 (PR-calculus)**.** The PR-calculus works on clauses and consists of the following rules:

1. the resolution rule:

$$\dfrac{\Gamma \vdash \Delta, A_1, \ldots, A_m \quad \Gamma', A'_1, \ldots, A'_n \vdash \Delta'}{\Gamma\sigma, \Gamma'\sigma \vdash \Delta\sigma, \Delta'\sigma}\,R$$

Where $n, m \geq 1$ and σ is a most general unifier of $\{A_1, \ldots, A_m, A'_1, \ldots, A'_n\}$. It is also required that $\Gamma \vdash \Delta, A$ and $\Gamma', A' \vdash \Delta'$ are variable disjoint.

2. the paramodulation rules:
   We assume that the two clauses in the premises are always variable disjoint and that σ is a most general unifier of $\{s, s'\}$.

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad A[s']_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda\sigma, \Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma} \qquad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad A[s']_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda\sigma, \Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma}$$

   for inference on the left side of the clauses and

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad \Gamma_2 \vdash \Delta_2, A[s']_\Lambda}{\Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma, A[t]_\Lambda\sigma} \qquad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad \Gamma_2 \vdash \Delta_2, A[s']_\Lambda}{\Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma, A[t]_\Lambda\sigma}$$

   for the right side, where Λ denotes a position of a subterm where s' is replaced by t. We call $s = t$ the *active equation* of the rules.

A *PR-derivation* from a set of clauses $\mathscr{C}$ is a tree derivation based on the rules above where all clauses in the leaves are variants of clauses in $\mathscr{C}$. A *PR-derivation* of $\vdash$ from $\mathscr{C}$ is called a *PR-refutation* of $\mathscr{C}$.

Note that the PR-calculus is defined as a simple resolution calculus extended by paramodulation. In fact, it can be simulated by $\mathrm{RPL}_0^=$, as the clauses in the characteristic clause set can be obtained from the corresponding characteristic formula by the rules of RPL_0.

It is important to show that the characteristic clause set is always refutable.

Theorem 3.5.2. *Let φ be a proof of a skolemized end-sequent. Then the characteristic clause set* $\mathrm{CL}(\varphi)$ *is refutable by resolution and paramodulation.*

Proof. In [BL11, BHL+06]. □

Example 3.5.3. We give a PR-refutation γ of $\mathrm{CL}(\varphi)$ for φ in Example 3.5.1.

$$\dfrac{\dfrac{\dfrac{(\pi)}{\vdash Pgc} \quad \dfrac{}{Pz \vdash f^2z = gz}\,R}{\vdash f^2gc = g^2c}\,para \qquad \dfrac{\dfrac{(\pi)}{\vdash Pgc} \quad \dfrac{}{Pz \vdash Pf^2z}\,R}{\vdash Pf^2gc}\,para}{\vdash Pg^2c \qquad Pg^2c \vdash}\,R}{\vdash}\,R$$

where π is

$$\dfrac{\dfrac{\vdash Pc \quad \dfrac{}{Pz \vdash Pf^2z}\,R}{\vdash Pf^2c}\,para \qquad \dfrac{\vdash Pc \quad \dfrac{}{Pz \vdash f^2z = gz}\,R}{\vdash f^2c = gc}\,para}{\vdash Pgc}\,R$$

To each clause in the characteristic clause set a projection is associated with it. Note that we do not always need all clauses in CL(φ) for a refutation of CL(φ) and construct only projections to those clauses, that actually occur in the refutation.

Definition 3.5.3. For a characteristic clause set CL(φ) and a refutation R of CL(φ) we define

$$\mathrm{CL}(\varphi, R) = \{C_i \,|\, C_i \in \mathrm{CL}(\varphi) \text{ and } C_i \text{ occurs in R}\}.$$

A projection of a clause C is a derivation built from φ by taking the axioms in which the atoms of C occur and all the inferences that operate on end-sequent ancestors. As a result, the end-sequent of a projection will be the end-sequent of φ extended by the atoms of C.

Definition 3.5.4 (proof projections, [BL00])**.** Let φ be a proof of a skolemized end-sequent $\Gamma \vdash \Delta$ in $\mathbf{LK}_=$. For nodes ν in φ we define inductively the set of cut-free proofs $p(\nu)$. If ν_0 is the root node and $\psi \in p(\nu_0)$ we call ψ a *projection*. Let ν be a node in φ such that $S(\nu) = \Gamma \vdash \Delta$; then $\Gamma \vdash \Delta = \Gamma_c, \Gamma_e \vdash \Delta_e, \Delta_c$ where $\Gamma_c \vdash \Delta_c$ consists of cut-ancestors and $\Gamma_e \vdash \Delta_e$ of ancestors of the end-sequent.

(a) ν is a leaf in φ. Then the sequent at ν is an axiom and we define $p(\nu) = \{\nu\}$. The clause part of ν is the subsequent CL(ν).

(b) ν is the conclusion of a unary rule ξ with premise μ.

(b1) The principal formula of ξ is an ancestor of a cut. Then $\varphi.\nu$ is of the form

$$\dfrac{\begin{matrix}(\varphi.\mu)\\ \Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e\end{matrix}}{\Gamma'_c, \Gamma_e \vdash \Delta'_c, \Delta_e}\ \xi$$

We define $p(\nu) = p(\mu)$.

(b2) The principal formula of ξ is an ancestor of the end- sequent. Then $\varphi.\nu$ is of the form

$$\dfrac{\begin{matrix}(\varphi.\mu)\\ \Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e\end{matrix}}{\Gamma_c, \Gamma'_e \vdash \Delta_c, \Delta'_e}\ \xi$$

Let $\psi \in p(\mu)$ be a proof of $C, \Gamma_e \vdash \Delta_e, D$ where $C \vdash D$ is the clause part of ψ. Then $\psi' \in p(\nu)$ for $\psi' =$

$$\dfrac{\begin{matrix}(\psi)\\ C, \Gamma_e \vdash \Delta_e, D\end{matrix}}{C, \Gamma'_e \vdash \Delta'_e, D}\ \xi$$

and $C \vdash D$ is the clause part of ψ'.

(c) ν is the conclusion of a binary rule ξ with premises μ_1, μ_2.

(c1) The auxiliary formulas of ξ are ancestors of a cut. Then $\varphi.\nu$ is of the form

$$\frac{\begin{array}{cc}(\varphi.\mu_1) & (\varphi.\mu_2)\\ \Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e & \Pi_c, \Pi_e \vdash \Lambda_c, \Lambda_e\end{array}}{\Gamma'_c, \Pi'_c, \Gamma_e, \Pi_e \vdash \Delta'_c, \Lambda'_c, \Delta_e, \Lambda_e}\ \xi$$

Let $\psi \in p(\mu_1)$ such that ψ is a proof of $C, \Gamma_e \vdash \Delta_e, D$ where $C \vdash D$ is the clause part of ψ. Then $\psi' \in p(\nu)$ for $\psi' =$

$$\frac{\begin{array}{c}(\psi)\\ C, \Gamma_e \vdash \Delta_e, D\end{array}}{C, \Gamma_e, \Pi_e \vdash \Delta_e, \Lambda_e, D}\ w^*$$

and $C \vdash D$ is the clause part of ψ'.

Let $\psi \in p(\mu_2)$ such that ψ is a proof of $E, \Pi_e \vdash \Lambda_e, F$ where $E \vdash F$ is the clause part of ψ. Then $\psi' \in p(\nu)$ for $\psi' =$

$$\frac{\begin{array}{c}(\psi)\\ E, \Pi_e \vdash \Lambda_e, F\end{array}}{E, \Gamma_e, \Pi_e \vdash \Delta_e, \Lambda_e, F}\ w^*$$

and $E \vdash F$ is the clause part of ψ'.

(c2) The auxiliary formulas of ξ are ancestors of the end-sequent. Then $\varphi.\nu$ is of the form

$$\frac{\begin{array}{cc}(\varphi.\mu_1) & (\varphi.\mu_2)\\ \Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e & \Pi_c, \Pi_e \vdash \Lambda_c, \Lambda_e\end{array}}{\Gamma_c, \Pi_c, \Gamma'_e, \Pi'_e \vdash \Delta_c, \Lambda_c, \Delta'_e, \Lambda'_e}\ \xi$$

Let $\psi_1 \in p(\mu_1)$ such that ψ_1 is a proof of $C, \Gamma_e \vdash \Delta_e, D$ and $C \vdash D$ is the clause part of ψ_1; likewise let $\psi_2 \in p(\mu_2)$ such that ψ_2 is a proof of $E, \Pi_e \vdash \Lambda_e, F$ and $E \vdash F$ is the clause part of ψ_2. Then $\psi \in p(\nu)$ for $\psi =$

$$\frac{\begin{array}{cc}(\psi_1) & (\psi_2)\\ C, \Gamma_e \vdash \Delta_e, D & E, \Pi_e \vdash \Lambda_e, F\end{array}}{C, E, \Gamma'_e, \Pi'_e \vdash \Delta'_e, \Lambda'_e, D, F}\ \xi$$

and the clause part of ψ is $C, E \vdash D, F$.

Example 3.5.4. Let φ be the proof from Example 3.5.1. We define the projections of φ to the clauses $Pz \vdash Pf^2z$, $Pz \vdash f^2z = gz$, $\vdash Pc$ and $Pg^2c \vdash$:

$\varphi[Pz \vdash Pf^2z]$ is

$$\dfrac{\dfrac{Pz \vdash Pz \qquad \dfrac{Pfz \vdash Pfz \qquad Pf^2z \vdash Pf^2z}{Pfz, Pfz \to Pf^2z \vdash Pf^2z}\to_l}{Pz, Pz \to Pfz, Pfz \to Pf^2z \vdash Pf^2z}\to_l}{\dfrac{Pz, \forall x(Px \to Pfx) \vdash Pf^2z}{Pz, Pc, \forall x(Px \to Pfx) \vdash Pg^2c, Pf^2z}\,w : l + w : r}\,2 \times \forall_l + c_l$$

$\varphi[Pz \vdash f^2z = gz]$ is

$$\dfrac{\dfrac{\dfrac{Pz \vdash Pz \qquad \dfrac{\dfrac{Pfz \vdash Pfz \qquad \dfrac{\vdash f^2z = gz}{Pf^2z \vdash f^2z = gz}\, w:l}{Pfz, Pfz \to Pf^2z \vdash f^2z = gz}\, \to_l}{Pz, Pz \to Pfz, Pfz \to Pf^2z \vdash f^2z = gz}\, \to_l}{Pz, \forall x(Px \to Pfx) \vdash f^2z = gz}\, 2 \times \forall_l + c_l}{Pz, Pc, \forall x(Px \to Pfx) \vdash Pg^2c, f^2z = gz}\, w:l + w:r$$

$\varphi[\vdash Pc]$ is

$$\dfrac{Pc \vdash Pc}{Pc, \forall x(Px \to Pfx) \vdash Pg^2c, Pc}\, w_l + w:r$$

$\varphi[Pg^2c \vdash]$ is

$$\dfrac{Pg^2c \vdash Pg^2c}{Pg^2c, Pc, \forall x(Px \to Pfx) \vdash Pg^2c}\, w_l + w_r$$

If we apply all most general unifiers in the PR proof γ we obtain a proof in $\mathbf{LK}_=$ (in fact only contractions, cut and paramodulation remain). If $\gamma\sigma$ is such a proof and we apply a substitution replacing all variables by a constant symbol we obtain a ground PR refutation.

Example 3.5.5. The ground PR-refutation γ' is

$$\dfrac{\dfrac{\dfrac{(\pi)}{\vdash Pgc} \qquad \dfrac{Pgc \vdash f^2gc = g^2c}{\vdash f^2gc = g^2c}\, cut}{\vdash Pg^2c}\, para \qquad \dfrac{\dfrac{(\pi)}{\vdash Pgc} \qquad \dfrac{Pgc \vdash Pf^2gc}{\vdash Pf^2gc}\, cut}{Pg^2c \vdash}\, cut}{\vdash}\, cut$$

where π is

$$\dfrac{\dfrac{\vdash Pc \qquad \dfrac{Pc \vdash Pf^2c}{\vdash Pf^2c}\, cut}{\vdash Pgc}\, para \qquad \dfrac{\vdash Pc \qquad \dfrac{Pc \vdash f^2c = gc}{\vdash f^2c = gc}\, cut}{\vdash Pgc}\, cut}{\vdash}\, cut$$

Note that γ' is an $\mathbf{LK}_=$-refutation of ground instances of clauses in CL(φ).

γ' can be used as a skeleton of a proof φ^* with only atomic cuts of the original end-sequent S, the clausal CERES normal form of the original proof φ. Below we give a formal definition. First we define a type of top normal form defined by a PR-deduction.

Definition 3.5.5 (top normal form, [LL19])**.** Let $\mathscr{C}\colon \{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ be a set of clauses, $\Gamma \vdash \Delta$ be a skolemized sequent and φ_i cut-free proofs of $C_i, \Gamma \vdash \Delta, D_i$ in $\mathbf{LK}_=$ for $i = 1, \ldots, n$. Let $\Phi = \{\varphi_1, \ldots, \varphi_n\}$. Given a PR-deduction ϱ of a clause $C \vdash D$ from $\mathscr{C}$ we define an $\mathbf{LK}_=$-proof $\Theta(\varrho, \mathscr{C}, \Phi)$ of $C, \Gamma \vdash \Delta, D$ inductively on the length of ϱ.

- $\varrho = C_i \vdash D_i$: then $\Theta(\varrho, \mathscr{C}, \Phi) = \varphi_i$ and $top(\Theta(\varrho, \mathscr{C}, \Phi)) = \{\varphi_i\}$.
- The last inference in ϱ is R. Then ϱ is of the form

$$\dfrac{\overset{(\varrho_1)}{E_1 \vdash F_1, A^m} \quad \overset{(\varrho_2)}{A^k, E_2 \vdash F_2}}{E_1, E_2 \vdash F_1, F_2}\, R$$

  Let us assume that

$$\Theta(\varrho_1, \mathscr{C}, \Phi) = \overset{(\psi_1)}{E_1, \Gamma \vdash \Delta, F_1, A^m} \quad \Theta(\varrho_2, \mathscr{C}, \Phi) = \overset{(\psi_2)}{A^k, E_2, \Gamma \vdash \Delta, F_2}$$

  Then we define $\Theta(\varrho, \mathscr{C}, \Phi) =$

$$\dfrac{\dfrac{\overset{(\psi_1)}{E_1, \Gamma \vdash \Delta, F_1, A^m} \quad \overset{(\psi_2)}{A^k, E_2, \Gamma \vdash \Delta, F_2}}{E_1, E_2, \Gamma, \Gamma \vdash \Delta, \Delta, F_1, F_2}\, c^*}{E_1, E_2, \Gamma \vdash \Delta, F_1, F_2}\, cut$$

  and $top(\Theta(\varrho, \mathscr{C}, \Phi)) = top(\Theta(\varrho_1, \mathscr{C}, \Phi)) \cup top(\Theta(\varrho_1, \mathscr{C}, \Phi))$.

- The last inference in ϱ is a paramodulation rule. We consider only the case $=_{r1}$; for the other rules the construction is analogous. Then ϱ is of the form

$$\dfrac{\overset{(\varrho_1)}{E_1 \vdash F_1, s = t} \quad \overset{(\varrho_2)}{E_2 \vdash F_2, A[s]_\Lambda}}{E_1, E_2 \vdash F_1, F_2, A[t]_\Lambda}\, =_{r1}$$

  Let us assume that

$$\Theta(\varrho_1, \mathscr{C}, \Phi) = \overset{(\psi_1)}{E_1, \Gamma \vdash \Delta, F_1, s = t} \quad \Theta(\varrho_2, \mathscr{C}, \Phi) = \overset{(\psi_2)}{E_2, \Gamma \vdash \Delta, F_2, A[s]_\Lambda}$$

  Then we define $\Theta(\varrho, \mathscr{C}, \Phi) =$

$$\dfrac{\dfrac{\overset{(\psi_1)}{E_1, \Gamma \vdash \Delta, F_1, s = t} \quad \overset{(\psi_2)}{E_2, \Gamma \vdash \Delta, F_2, A[s]_\Lambda}}{E_1, E_2, \Gamma, \Gamma \vdash \Delta, \Delta, F_1, F_2, A[t]_\Lambda}\, =_{r1}}{E_1, E_2, \Gamma \vdash \Delta, F_1, F_2, A[t]_\Lambda}\, c^*$$

  and $top(\Theta(\varrho, \mathscr{C}, \Phi)) = top(\Theta(\varrho_1, \mathscr{C}, \Phi)) \cup top(\Theta(\varrho_1, \mathscr{C}, \Phi))$.

A proof ψ is called in *top normal form* if there are $\mathscr{C}, \Phi$ and ρ (defined as above) such that $\psi = \Theta(\rho, \mathscr{C}, \Phi)$.

Remark. The function *top* collects all cut-free subproofs in a top normal form which occur at the top and thus belong to Φ.

Definition 3.5.6 (clausal CERES normal form, [BL00])**.** Let φ be an $\mathbf{LK}_=$ proof of a skolemized sequent S. Let ϱ be a grounded PR-refutation of $\mathrm{CL}(\varphi)$, $\mathscr{C}$ be the set of all ground instances of clauses in $\mathrm{CL}(\varphi)$ appearing at the leaves of ϱ and Φ be the set of all grounded projections. Then the proof $\Theta(\varrho, \mathscr{C}, \Phi)$ is called a *clausal* CERES *normal form* of ϱ. As ϱ is a refutation $\Theta(\varrho, \mathscr{C}, \Phi)$ is a proof of S with only atomic cuts.

Remark. Note that not all top normal forms are CERES normal forms as the set of cut-free proofs Φ need not be projections.

Example 3.5.6. We define a clausal CERES normal form for the proof from Example 3.5.1 with respect to the grounded resolution refutation γ' of $\mathrm{CL}(\varphi)$ (in the following example $F = \forall x(Px \to Pfx)$):

$$\dfrac{\dfrac{\dfrac{(\varphi_1) \qquad (\varphi_2)}{Pc, F \vdash Pg^2c, f^2gc = g^2c \qquad Pc, F \vdash Pg^2c, Pf^2gc} =_{r1}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pg^2c}{Pc, F \vdash Pg^2c}\, c_l + c_r} \qquad \dfrac{\varphi[Pg^2c \vdash]}{Pg^2c, Pc, F \vdash Pg^2c}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c}{Pc, F \vdash Pg^2c}\, c_l + c_r}\, cut$$

where φ_1 is

$$\dfrac{\dfrac{\varphi_{11} \qquad \varphi[Pz \vdash f^2z = gz]\{z \leftarrow gc\}}{Pc, F \vdash Pg^2c, Pgc \qquad Pgc, Pc, F \vdash Pg^2c, f^2gc = g^2c}\, cut}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, f^2gc = g^2c}{Pc, F \vdash Pg^2c, f^2gc = g^2c}\, c_l + c_r}$$

φ_{11} is

$$\dfrac{\dfrac{\pi_1 \qquad \pi_2}{Pc, F \vdash Pg^2c, Pf^2c \qquad Pc, F \vdash Pg^2c, f^2c = gc} =_{r1}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pgc}{Pc, F \vdash Pg^2c, Pgc}\, c_l + c_r}$$

φ_2 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\pi_1 \qquad \pi_2}{Pc, F \vdash Pg^2c, Pf^2c \qquad Pc, F \vdash Pg^2c, f^2c = gc} =_{r1}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pgc} c_l + c_r}{Pc, F \vdash Pg^2c, Pgc} \qquad \dfrac{\varphi[Pz \vdash Pf^2z]\{z \leftarrow gc\}}{Pgc, Pc, F \vdash Pg^2c, Pf^2gc}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pf^2gc}{Pc, F \vdash Pg^2c, Pf^2gc} c_l + c_r} cut$$

π_1 is

$$\dfrac{\dfrac{\varphi[\vdash Pc] \qquad \varphi[Pz \vdash Pf^2z]\{z \leftarrow c\}}{Pc, F \vdash Pg^2c, Pc \qquad Pc, Pc, F \vdash Pg^2c, Pf^2c}}{Pc, F \vdash Pg^2c, Pf^2c} cut$$

and π_2 is

$$\dfrac{\dfrac{\varphi[\vdash Pc] \qquad \varphi[Pz \vdash f^2z = gz]\{z \leftarrow c\}}{Pc, F \vdash Pg^2c, Pc \qquad Pc, Pc, F \vdash Pg^2c, f^2c = gc}}{Pc, F \vdash Pg^2c, f^2c = gc} cut$$

3.5.2 Extraction of Expansion Proofs

All the definitions and theorems in this section are published in [LL19]. First, we will extend the definitions from Section 2.3 to a logic with equality. The extraction of expansion proofs from $\mathbf{LK}_=$-proofs requires quantifier-free cuts. Due to the structure of the clausal CERES method we consider proofs with only atomic cuts.

Definition 3.5.7. A proof φ in $\mathbf{LK}_=$ is in the subclass $\mathbf{LK}_0$ if

1. φ does not contain strong quantifier inferences.
2. All cuts in φ are atomic.
3. Equality rules are only applied to atoms.
4. The axiom set contains Ax.

Now we extend Definition 2.3.10 to handle proofs in $\mathbf{LK}_=$.

Definition 3.5.8 (extraction of s-expansion trees from proofs in $\mathbf{LK}_0$)**.** We extend the transformation ET from Definition 2.3.10 in such a way that it maps proofs in $\mathbf{LK}_0$ to s-expansion trees. The transformation is defined inductively on the number of inferences in a proof, we have the same cases as in Definition 2.3.10 only that we replace $\mathbf{LK}$-proofs with $\mathbf{LK}_0$-proofs and we add the cases

- If $\varphi =$

$$\dfrac{\begin{array}{cc}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, s=t & A[s]_\Lambda, \Gamma_2 \vdash \Delta_2\end{array}}{A[t]_\Lambda, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} =_{l1}$$

and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, s=t$ and $\mathrm{ET}(\pi_2) = A[s]_\Lambda, \Gamma_2^* \vdash \Delta_2^*$, then $\mathrm{ET}(\varphi) = \Gamma_1^*, \Gamma_2^*, A[t]_\Lambda \vdash \Delta_1^*, \Delta_2^*$.

- If $\varphi =$

$$\dfrac{\begin{array}{cc}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, s=t & \Gamma_2 \vdash \Delta_2, A[s]_\Lambda\end{array}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda} =_{r1}$$

and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, s=t$ and $\mathrm{ET}(\pi_1) = \Gamma_2^* \vdash \Delta_2^*, A[s]_\Lambda$, then $\mathrm{ET}(\varphi) = \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A[t]_\Lambda$.

$=_{l2}$ and $=_{r2}$ are omitted, the transformation of the these rules being analogous.

Again, we obtain a soundness result.

Proposition 3.5.3. *The transformation* ET *is sound: if φ is a proof in* $\mathbf{LK}_0$ *then* $\mathrm{ET}(\varphi)$ *is an s-expansion proof.*

Proof. As in the proof of Proposition 2.3.1, but we add a case for $=_{r_1}$ (the other cases are analogous).

- $(=_{r_1})$ $\varphi =$

$$\dfrac{\begin{array}{cc}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, s=t & \Gamma_2 \vdash \Delta_2, A[s]_\Lambda\end{array}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda} =_{r1}$$

and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, s=t$ and $\mathrm{ET}(\pi_1) = \Gamma_2^* \vdash \Delta_2^*, A[s]_\Lambda$ are s-expansion proofs. Therefore $\neg\Gamma_1^* \vee \Delta_1^* \vee s=t$ and $\neg\Gamma_2^* \vee \Delta_2^* \vee A[s]_\Lambda$ are expansion proofs and hence $\mathscr{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee s=t)$ and $\mathscr{A} \models Dp(\neg\Gamma_2^* \vee \Delta_2^* \vee A[s]_\Lambda)$. But then $\mathscr{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee A[t]_\Lambda)$ and $\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee A[t]_\Lambda)$ is an expansion proof. Therefore, $\Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A[t]_\Lambda$ $(= \mathrm{ET}(\varphi))$ is an s-expansion-proof.

□

To illustrate construction of an s-expansion proof from an $\mathbf{LK}_0$-proof , consider the following simple example.

Example 3.5.7. We work with $Ax \cup \{\vdash a = b\}$. Let φ be the proof of $S = P(a) \wedge Q(a, f(a)) \vdash \exists x(P(x) \wedge \exists y.Q(x, y))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash a = b \quad Q(a,f(a)) \vdash Q(a,f(a))}{Q(a,f(a)) \vdash Q(a,f(b))}\,=_{r_1}}{Q(a,f(a)) \vdash \exists y.Q(a,y)}\,\exists_r \qquad P(a) \vdash P(a)}{P(a), Q(a,f(a)) \vdash P(a) \wedge \exists y.Q(a,y)}\,\wedge_r}{P(a) \wedge Q(a,f(a)) \vdash P(a) \wedge \exists y.Q(a,y)}\,\wedge_l}{P(a) \wedge Q(a,f(a)) \vdash \exists x(P(x) \wedge \exists y.Q(x,y))}\,\exists_r$$

Note that S is not in prenex form. Therefore, extracting the Herbrand sequent by collecting instances is not possible. Instead we compute the expansion proof $\mathrm{ET}(\varphi)$.

Below we compute the s-expansion proof corresponding to φ. First we compute expansion trees for all formulas F in S and call them $\mathrm{ET}(F)$.

$$\begin{array}{lcl} \mathrm{ET}(P(a) \wedge Q(a, f(a))) & = & P(a) \wedge Q(a, f(a)) \\ \mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x,y))) & = & \exists x(P(x) \wedge \exists y.Q(x,y)) \\ & & +^a (P(a) \wedge (\exists y Q(a,y) +^{f(b)} Q(a, f(b)))) \end{array}$$

The s-expansion proof $\mathrm{ET}(\varphi)$ associated with the end-sequent S is:

$$\mathrm{ET}(P(a) \wedge Q(a, f(a))) \quad \vdash \quad \mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x,y)))$$

The corresponding expansion proof associated with $\mathrm{ET}(\varphi)$ is:

$$\neg \mathrm{ET}(P(a) \wedge Q(a, f(a))) \vee \mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x,y))).$$

To obtain the tautologous formula (corresponding to the Herbrand sequent) we construct the deep function for the expansion proof; we compute $Dp(\mathrm{ET}(S_i))$:

$$Dp(\mathrm{ET}(P(a) \wedge Q(a, f(a)))) \quad = \quad Dp(P(a)) \wedge Dp(Q(a, f(a))) \quad = \quad P(a) \wedge Q(a, f(a))$$

$$\begin{array}{ll} Dp(\mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x,y)))) & = \\ Dp(\exists x(P(x) \wedge \exists y.Q(x,y)) +^a (P(a) \wedge (\exists y Q(a,y) +^{f(b)} Q(a, f(b))))) & = \\ Dp(P(a) \wedge (\exists y Q(a,y) +^{f(b)} Q(a, f(b)))) & = \\ Dp(P(a)) \wedge Dp(\exists y Q(a,y) +^{f(b)} Q(a, f(b))) & = \\ P(a) \wedge Q(a, f(b)) & \end{array}$$

Hence, we obtain $P(a) \wedge Q(a, f(a)) \vdash P(a) \wedge Q(a, f(b))$. Note that this sequent is valid in $Ax \cup \{\vdash a = b\}$ (though it is not tautological).

The extraction of expansion proofs is usually performed after the construction of a proof in top normal form. However, only the *logical* parts of the proof play a role in the construction of expansion trees. These logical parts can be identified as the cut-free subproofs after removal of all cut-ancestors. Note that no cut-ancestor in such a subproof is principal formula of an inference; we identify such subsequents as *passive subsequent*.

Definition 3.5.9 (passive subsequent)**.** Let φ be a cut-free proof of $S : C, \Gamma \vdash \Delta, D$ such that $C \vdash D$ is a clause. The subsequent $C \vdash D$ of S is called *passive* in φ if no ancestor of $C \vdash D$ in φ contains a formula which is principal formula of an inference.

Note that the passive subsequents are just the clauses used to define a top normal form. Examples of proofs with passive clause parts are proof projections in CERES:

Proposition 3.5.4. *Let ψ be a cut-free proof of $C', \Gamma \vdash \Delta, D'$ which is an instance of a proof projection $\varphi[C \vdash D]$ in clausal* CERES*. Then $C' \vdash D'$ is passive in ψ.*

Proof. Immediate by induction on the length of ψ and by Definition 3.5.4. Note that the only case in Definition 3.5.4 where the clause part changes is (c2). Here the projection ψ is defined as

$$\dfrac{\begin{matrix}(\psi_1) & (\psi_2)\\ C, \Gamma_e \vdash \Delta_e, D & E, \Pi_e \vdash \Lambda_e, F\end{matrix}}{C, E, \Gamma'_e, \Pi'_e \vdash \Delta'_e, \Lambda'_e, D, F}\ \xi$$

By induction hypothesis $C \vdash D$ is passive in ψ_1 and $E \vdash F$ is passive in ψ_2. Therefore $C, E \vdash D, F$ is passive in ψ. □

Definition 3.5.10. Let φ be a cut-free proof of $C, \Gamma \vdash \Delta, D$ where $C \vdash D$ is passive in φ. We define $\varphi \backslash (C \vdash D)$ by induction on the number of nodes in φ.

- If φ is an axiom then $\varphi = C, C' \vdash D, D'$ (note that the whole sequent is passive in φ). We define $\varphi \backslash (C \vdash D) = C' \vdash D'$.
- Let $\varphi =$

$$\dfrac{\varphi'}{C, \Gamma \vdash \Delta, D}\ x$$

  where $C \vdash D$ is passive in φ. Then, by definition of passive subclauses, φ' is a proof of $C, \Gamma' \vdash \Delta', D$ for some Γ' and Δ'. Indeed, the subclause $C \vdash D$ does not contain a formula which is principal formula of an inference. By induction we have a proof $\varphi' \backslash (C \vdash D)$ of $\Gamma' \vdash \Delta'$ (note that $C \vdash D$ is also passive in φ') and we define $\varphi \backslash (C \vdash D) =$

$$\dfrac{\begin{matrix}\varphi' \backslash (C \vdash D)\\ \Gamma' \vdash \Delta'\end{matrix}}{\Gamma \vdash \Delta}\ x$$

- Let $\varphi =$

$$\dfrac{\begin{array}{cc}(\varphi_1) & (\varphi_2)\\ S_1 & S_2\end{array}}{C, \Gamma \vdash \Delta, D}\, x$$

where $C \vdash D$ is passive in φ. As C, D are not principal formulas of an inference we get that $S_1 = C_1, \Gamma_1 \vdash \Delta_1, D_1$, $S_2 = C_2, \Gamma_2 \vdash \Delta_2, D_2$, s.t. $C_1, C_2 \vdash D_1, D_2 = C \vdash D$ and $C_1 \vdash D_1$ is passive in φ_1, $C_2 \vdash D_2$ is passive in φ_2.

By induction we have a proof $\varphi_1 \backslash (C_1 \vdash D_1)$ of $\Gamma_1 \vdash \Delta_1$ and a proof $\varphi_2 \backslash (C_2 \vdash D_2)$ of $\Gamma_2 \vdash \Delta_2$. Then we obtain $\varphi \backslash (C \vdash D) =$

$$\dfrac{\begin{array}{cc}(\varphi_1 \backslash (C_1 \vdash D_1)) & (\varphi_2 \backslash (C_2 \vdash D_2))\\ \Gamma_1 \vdash \Delta_1 & \Gamma_2 \vdash \Delta_2\end{array}}{\Gamma \vdash \Delta}\, x$$

The function logical(φ) for a proof in top normal form takes the cut-free proofs on top and "subtracts" from them all ancestors of passive clauses.

Definition 3.5.11 (logical(φ))**.** Let $\varphi\colon \Theta(\rho, \mathscr{C}, \Phi)$ be a proof in top normal form s.t. $\mathscr{C} = \{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ and $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ such that φ_i is a cut-free proof of $C_i, \Gamma \vdash \Delta, D_i$. Assume that for all $i = 1, \ldots, n$ $C_i \vdash D_i$ is passive in φ_i. For every $\psi \in \mathit{top}(\varphi)$ and $\psi = \varphi_i$ we define $\psi' = \varphi_i \backslash (C_i \vdash D_i)$ and logical(φ) $= \{\psi' \mid \psi \in \mathit{top}(\varphi)\}$.

Below we define an expansion tree $\hat{E}(\varphi)$ which is defined by merging the expansion trees of logical(φ). This structure will be the key for the development of an efficient algorithm for extracting expansion trees from clausal CERES normal forms.

Definition 3.5.12. Let φ be a proof in top normal form of a skolemized and normalized end-sequent. We define

$$\hat{E}(\varphi) = \mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi)\}.$$

Theorem 3.5.5. *Let $\varphi\colon \Theta(\varrho, \mathscr{C}, \Phi)$ be a proof of a skolemized and normalized sequent $C, \Gamma \vdash \Delta, D$ in top normal form such that $\mathscr{C} = \{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ and $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, where φ_i is a cut-free proof of $C_i, \Gamma \vdash \Delta, D_i$. Assume that for all $i = 1, \ldots, n$ $C_i \vdash D_i$ is passive in φ_i. Then* $\mathrm{ET}(\varphi) = \hat{E}(\varphi) \circ (C \vdash D)$.

Proof. By induction on the number of nodes in ϱ.

case 1: if ρ consists of just one node then $\varphi = \varphi_i$ for some $i \in \{1, \ldots, n\}$. We have to show that $\mathrm{ET}(\varphi_i) = \hat{E}(\varphi_i) \circ (C_i \vdash D_i)$. But $\hat{E}(\varphi_i) = \mathrm{ET}(\varphi_i \backslash (C_i \vdash D_i))$ and thus it remains to show that

$$(\star)\ \mathrm{ET}(\varphi_i) = \mathrm{ET}(\varphi_i \backslash (C_i \vdash D_i)) \circ (C_i \vdash D_i).$$

($\star$) is obtained via an easy induction on the number of inferences in φ_i using Definition 3.5.10.

case 2: The last inference in ϱ is R. Then ϱ is of the form

$$\frac{\begin{matrix}(\varrho_1)\\ C_1 \vdash D_1, A^m\end{matrix} \quad \begin{matrix}(\varrho_2)\\ A^k, C_2 \vdash D_2\end{matrix}}{C_1, C_2 \vdash D_1, D_2}\ R$$

Then (by definition of φ as $\Theta(\varrho, \mathscr{C}, \Phi)$) $\varphi =$

$$\dfrac{\dfrac{\begin{matrix}(\varphi_1)\\ C_1, \Gamma \vdash \Delta, D_1, A^m\end{matrix} \quad \begin{matrix}(\varphi_2)\\ A^k, C_2, \Gamma \vdash \Delta, D_2\end{matrix}}{C_1, C_2, \Gamma, \Gamma \vdash \Delta, \Delta, D_1, D_2}\ cut}{C_1, C_2, \Gamma \vdash \Delta, D_1, D_2}\ c^*$$

Assume that

$$\begin{aligned} \mathrm{ET}(\varphi_1) &= C_1, \Gamma^* \vdash \Delta^*, D_1, A^m, \\ \mathrm{ET}(\varphi_2) &= A^k, C_2, \Gamma^+ \vdash \Delta^+, D_2, \end{aligned}$$

where $Seq(\Gamma^* \vdash \Delta^*) = Seq(\Gamma^+ \vdash \Delta^+)$. By Definition 2.3.10 we obtain

$$(1)\ \mathrm{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2) \circ \mathrm{merge}(\Gamma^* \vdash \Delta^*, \Gamma^+ \vdash \Delta^+)$$

Note that Γ^*, Γ^+ and Δ^*, Δ^+ are normalized. By induction hypothesis we have

$$\begin{aligned} \hat{E}(\varphi_1) \circ (C_1 \vdash D_1, A^m) &= \mathrm{ET}(\varphi_1), \\ \hat{E}(\varphi_2) \circ (A^k, C_2 \vdash D_2) &= \mathrm{ET}(\varphi_2). \end{aligned}$$

and therefore

$$(2)\ \hat{E}(\varphi_1) = \Gamma^* \vdash \Delta^*, \quad \hat{E}(\varphi_2) = \Gamma^+ \vdash \Delta^+.$$

By definition of the merge operator we get from (1) and (2)

$$(3)\ \mathrm{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2) \circ \mathrm{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)).$$

By Definition 3.5.12 we obtain

$$\begin{aligned} \hat{E}(\varphi_1) &= \mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi_1)\}, \\ \hat{E}(\varphi_2) &= \mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi_2)\}. \end{aligned}$$

Hence

$$\begin{aligned} &\mathrm{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)) = \\ &\mathrm{merge}(\mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi_1)\}, \mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi_2)\}) = \\ &\mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi)\}. \end{aligned}$$

But by Definition 3.5.12 $\hat{E}(\varphi) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}$. Combing this with (3) we obtain

$$\text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2) \circ \hat{E}(\varphi).$$

case 3: The last inference in ϱ is a paramodulation. We only consider the case $=_{r_1}$, the others are analogous. Then ϱ is of the form

$$\frac{\begin{array}{cc}(\varrho_1) & (\varrho_2)\\ C_1 \vdash D_1, s = t & C_2 \vdash D_2, A[s]\end{array}}{C_1, C_2 \vdash D_1, D_2, A[t]} =_{r_1}$$

Then, by definition of φ, we obtain $\varphi=$

$$\frac{\dfrac{\begin{array}{cc}(\varphi_1) & (\varphi_2)\\ C_1, \Gamma \vdash \Delta, D_1, s = t & C_2, \Gamma \vdash \Delta, D_2, A[s]\end{array}}{C_1, C_2, \Gamma, \Gamma \vdash \Delta, \Delta, D_1, D_2, A[t]} =_{r_1}}{C_1, C_2, \Gamma \vdash \Delta, D_1, D_2, A[t]}\, c^*$$

Assume that

$$\begin{array}{rcl}\text{ET}(\varphi_1) & = & C_1, \Gamma^* \vdash \Delta^*, D_1, s = t,\\ \text{ET}(\varphi_2) & = & C_2, \Gamma^+ \vdash \Delta^+, D_2, A[s]\end{array}$$

where $Seq(\Gamma^* \vdash \Delta^*) = Seq(\Gamma^+ \vdash \Delta^+)$. By Definition 2.3.10 we obtain

$$(4)\ \text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2, A[t]) \circ \text{merge}(\Gamma^* \vdash \Delta^*, \Gamma^+ \vdash \Delta^+).$$

By induction hypothesis we have

$$\begin{array}{rcl}\hat{E}(\varphi_1) \circ (C_1 \vdash D_1, s = t) & = & \text{ET}(\varphi_1),\\ \hat{E}(\varphi_2) \circ (C_2 \vdash D_2, A[s]) & = & \text{ET}(\varphi_2).\end{array}$$

and therefore

$$(5)\ \hat{E}(\varphi_1) = \Gamma^* \vdash \Delta^*, \quad \hat{E}(\varphi_2) = \Gamma^+ \vdash \Delta^+.$$

By definition of the merge operator we get from (4) and (5)

$$(6)\ \text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2, A[t]) \circ \text{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)).$$

By Definition 3.5.12 we obtain

$$\begin{array}{rcl}\hat{E}(\varphi_1) & = & \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi_1)\},\\ \hat{E}(\varphi_2) & = & \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi_2)\}.\end{array}$$

Hence, like in case 2, we get

$$\text{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}.$$

But by Definition 3.5.12 $\hat{E}(\varphi) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}$. Combing this with (6) we obtain

$$\text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2, A[t]) \circ \hat{E}(\varphi).$$

□

Corollary. *Let $\varphi\colon \Theta(\varrho, \mathscr{C}, \Phi)$ be a proof of a skolemized and normalized sequent $\Gamma \vdash \Delta$ in top normal form s.t. $\mathscr{C} = \{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ and $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ such that φ_i is a cut-free proof of $C_i, \Gamma \vdash \Delta, D_i$. Assume that for all $i = 1, \ldots, n$ $C_i \vdash D_i$ is passive in φ_i. Then $\mathrm{ET}(\varphi) = \hat{E}(\varphi)$.*

Proof. Immediate by Theorem 3.5.5: just define $C \vdash D$ as the empty sequent. □

Corollary. *Let φ be an $\mathbf{LK}_=$ proof of a skolemized and normalized sequent S. Let $\varphi^*\colon \Theta(\varrho, \mathscr{C}, \Phi)$ be a clausal CERES normal form of φ such that ϱ is a ground PR-refutation of $\mathscr{C}$, the set of all ground instances of clauses in $\mathrm{CL}(\varphi)$, and Φ is the set of all grounded projections. Then $\mathrm{ET}(\varphi) = \hat{E}(\varphi)$.*

Proof. Let ψ be a cut-free proof of $C, \Gamma \vdash \Delta, D$ which is an instance of a projection of φ. By Proposition 3.5.4 $C \vdash D$ is passive in ψ. As clausal CERES normal forms are top normal forms all conditions of Corollary 3.5.2 are fulfilled. □

The last corollary describes a method to compute an expansion tree from any proof in top normal form of a skolemized sequent S. Note that in case of a prenex sequent S we extract Herbrand sequents. The computation of an expansion tree is based on top normal forms. Clausal CERES normal forms φ^* of proofs φ are also in top normal form, therefore we can compute expansion trees in the same way. For clausal CERES it means that φ^* has to be constructed first. The usual algorithm for the extraction of expansion trees from the clausal CERES normal form is the algorithm EXP.

Definition 3.5.13 (algorithm EXP)**.**

Begin.

1. compute $\mathrm{CL}(\varphi)$;
2. find a PR refutation ρ of $\mathrm{CL}(\varphi)$;
3. compute a ground refutation R from ρ;
4. compute the projections $\varphi[C]$ for $C \in \mathrm{CL}(\varphi, R)$;
5. construct the clausal CERES normal form φ^* from the projections and R;
6. extract an expansion proof $\mathrm{ET}(\varphi)$ from φ^*.

End.

Instead of using algorithm EXP, we make use of Theorem 3.5.5 and define a new method that extracts expansion proofs more efficiently by extracting partial expansion trees from the projections. The idea is the following: we do not compute logical(φ^*) which would be the set of all instantiated projections. Note that the size of logical(φ^*) is roughly the size

of φ^* itself. Instead we use from a ground resolution refutation R of $\mathrm{CL}(\varphi)$ the general projections $\varphi[C]$ for $C \in \mathrm{CL}(\varphi, R)$ and the set of substitutions $\Sigma(C)$ for $C \in \mathrm{CL}(\varphi, R)$ which are the set of ground substitutions for the clause C in the refutation R.

Definition 3.5.14. For every projection $\varphi[C]\colon \vec{A}, \Gamma \vdash \Delta, \vec{B}$, where $\vec{A} \vdash \vec{B}$ is the clause part of $\varphi[C]$, we define $\varphi^-[C] = \varphi(C) \backslash (\vec{A} \vdash \vec{B})$ (note that $\vec{A} \vdash \vec{B}$ is a passive subsequent of $\vec{A}, \Gamma \vdash \Delta, \vec{B}$), where the $\backslash$-operator is defined as in Definition 3.5.10. Note that $\mathrm{ET}(\varphi^-[C])$ is a proof relative to the axioms in $\varphi^-[C]$, which may differ from the axioms of φ (axioms need not be tautological anyway). We define

$$T(\varphi, R) = \mathrm{merge}_{C \in \mathrm{CL}(\varphi,R)} \mathrm{merge}_{\sigma \in \Sigma(C)} \mathrm{ET}(\varphi^-[C])\sigma.$$

Then the computation of an expansion tree via clausal CERES for a proof φ (of a closed skolemized end-sequent S) can be defined with the algorithm EXP_{new} (note that we are defining EXP_{new} for the clausal CERES method and the algorithm is therefore based on the construction of a characteristic clause set and individual proof projections).

Definition 3.5.15 (algorithm EXP_{new})**.**

Begin.

1. compute $\mathrm{CL}(\varphi)$;
2. find a PR refutation ρ of $\mathrm{CL}(\varphi)$;
3. compute a ground refutation R from ρ and for every $C \in \mathrm{CL}(\varphi, R)$ the set $\Sigma(C)$;
4. compute the projections $\varphi[C]$ and $\varphi^-[C]$ for $C \in \mathrm{CL}(\varphi, R)$;
5. for every $C \in \mathrm{CL}(\varphi, R)$ compute $T[C]\colon \mathrm{merge}_{\sigma \in \Sigma(C)} \mathrm{ET}(\varphi^-[C])\sigma$;
6. compute $\mathrm{merge}_{C \in \mathrm{CL}(\varphi,R)} T[C]$ which is $T(\varphi, R)$.

End.

Note that the computation of the *Dp* function of an expansion proof via clausal CERES for a proof φ (of a closed skolemized end-sequent S) can be easily obtained by computing the *Dp* function of $T(\varphi, R)$.

Theorem 3.5.6. *Let φ be a proof of a skolemized, closed and normalized end-sequent and φ^* the clausal* CERES *normal form based on a ground refutation R of* $\mathrm{CL}(\varphi)$*. Then* $\mathrm{ET}(\varphi^*) = T(\varphi, R)$.

Proof. Let $\mathrm{CL}(\varphi, R) = \{C_1, \ldots, C_n\}$. Now $\mathrm{logical}(\varphi^*) = \Phi_1 \cup \ldots \cup \Phi_n$ where $\Phi_i = \{\varphi^-[C_i]\sigma \,|\, \sigma \in \Sigma(C_i)\}$. Let $\psi_i = \varphi^-[C_i]$. Then by Theorem 3.5.5 we know that

$$\mathrm{ET}(\varphi^*) = \mathrm{merge}(\mathrm{merge}_{\sigma \in \Sigma(C_1)} \mathrm{ET}(\psi_1\sigma), \cdots, \mathrm{merge}_{\sigma \in \Sigma(C_n)} \mathrm{ET}(\psi_n\sigma))$$

which is equal to

$$\mathrm{ET}(\varphi^*) = \mathrm{merge}(\mathrm{merge}_{\sigma\in\Sigma(C_1)}\mathrm{ET}(\psi_1)\sigma, \cdots, \mathrm{merge}_{\sigma\in\Sigma(C_n)}\mathrm{ET}(\psi_n)\sigma)$$

which, by Definition 3.5.14, is exactly $T(\varphi, R)$. So $\mathrm{ET}(\varphi^*) = T(\varphi, R)$. □

Theorem 3.5.6 also holds for the Dp function of expansion proofs, i.e. $Dp(\mathrm{ET}(\varphi^*)) = Dp(T(\varphi, R))$.

Corollary. *Let φ be a proof of a skolemized,closed and normalized sequent S and R be a refutation of* $\mathrm{CL}(\varphi)$. *Then $T(\varphi, R)$ is an expansion proof of S.*

Proof. By Theorem 3.5.5 and Theorem 3.5.6. □

Instead of computing all $\varphi[C_j]\sigma_i^j$ (obtained from the ACNF φ^*) the algorithm EXP_{new} computes the $\varphi[C_j]$ and extracts $\mathrm{ET}(\varphi^-[C_j]) \equiv T_j$, which is a partial expansion proof, then constructs $\mathrm{merge}_{\sigma\in\Sigma(C_j)}T_j\sigma$ for all j and merges them. Example 3.5.8 illustrates the main features of the method.

Example 3.5.8. Consider the proof φ as in Example 3.5.1 (where $F = \forall x(Px \to Pfx)$). The ACNF φ is:

$$\dfrac{\dfrac{\dfrac{(\varphi_1)}{Pc, F \vdash Pg^2c, f^2gc = g^2c} \quad \dfrac{(\varphi_2)}{Pc, F \vdash Pg^2c, Pf^2gc}\, para}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pg^2c}{Pc, F \vdash Pg^2c}\, c_l + c_r} \quad \dfrac{\varphi[Pg^2c \vdash]}{Pg^2c, Pc, F \vdash Pg^2c}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c}{Pc, F \vdash Pg^2c}\, c_l + c_r}\, cut$$

where φ_1 is

$$\dfrac{\dfrac{\varphi_{1_1}}{Pc, F \vdash Pg^2c, Pgc} \quad \dfrac{\varphi[Pz \vdash f^2z = gz]\{z \leftarrow gc\}}{Pgc, Pc, F \vdash Pg^2c, f^2gc = g^2c}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, f^2gc = g^2c}{Pc, F \vdash Pg^2c, f^2gc = g^2c}\, c_l + c_r}\, cut$$

φ_{1_1} is

$$\dfrac{\dfrac{\overset{\pi_1}{Pc, F \vdash Pg^2c, Pf^2c} \qquad \overset{\pi_2}{Pc, F \vdash Pg^2c, f^2c = gc}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pgc}\, para}{Pc, F \vdash Pg^2c, Pgc}\, c_l + c_r$$

φ_2 is

$$\dfrac{\dfrac{\dfrac{\overset{\pi_1}{Pc, F \vdash Pg^2c, Pf^2c} \qquad \overset{\pi_2}{Pc, F \vdash Pg^2c, f^2c = gc}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pgc}\, para}{Pc, F \vdash Pg^2c, Pgc}\, c_l + c_r \qquad \dfrac{\varphi[Pz \vdash Pf^2z]\{z \leftarrow gc\}}{Pgc, Pc, F \vdash Pg^2c, Pf^2gc}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, Pf^2gc}\, cut}{Pc, F \vdash Pg^2c, Pf^2gc}\, c_l + c_r$$

π_1 is

$$\dfrac{\dfrac{\varphi[\vdash Pc]}{Pc, F \vdash Pg^2c, Pc} \qquad \dfrac{\varphi[Pz \vdash Pf^2z]\{z \leftarrow c\}}{Pc, Pc, F \vdash Pg^2c, Pf^2c}}{Pc, F \vdash Pg^2c, Pf^2c}\, cut$$

and π_2 is

$$\dfrac{\dfrac{\varphi[\vdash Pc]}{Pc, F \vdash Pg^2c, Pc} \qquad \dfrac{\varphi[Pz \vdash f^2z = gz]\{z \leftarrow c\}}{Pc, Pc, F \vdash Pg^2c, f^2c = gc}}{Pc, F \vdash Pg^2c, f^2c = gc}\, cut$$

Now compute the Herbrand sequent of φ^* (with the old method):

$$H(\varphi^*) \quad = \quad Pc, Pc \to Pfc, Pfc \to Pf^2c, Pgc \to Pfgc, Pfgc \to Pf^2gc \vdash Pg^2c$$

Note that $H(\varphi^*)$ is a valid sequent in our axiom set ($\vdash f^2z = gz$ is an axiom).

With our new method we first compute $T_i = \mathrm{ET}(\varphi^-[C_i])$ and define the substitutions σ_i^j:

$$\begin{array}{lcl}
T_1 & = & Pc, \forall x(Px \to Pfx) +^z Pz \to Pfz +^{fz} Pfz \to Pf^2z \vdash Pg^2c \\
T_2 & = & Pc, \forall x(Px \to Pfx) +^z Pz \to Pfz +^{fz} Pfz \to Pf^2 \vdash Pg^2c \\
T_3 & = & Pc, \forall x(Px \to Pfx) \vdash Pg^2c \\
T_4 & = & Pc, \forall x(Px \to Pfx) \vdash Pg^2c \\
\sigma_1^1 & = & (z \leftarrow c) \quad \sigma_2^1 = (z \leftarrow gc) \\
\sigma_1^2 & = & (z \leftarrow c) \quad \sigma_2^2 = (z \leftarrow gc)
\end{array}$$

Note that $T_1 = T_2$. Now we compute $T(\varphi, R) = \text{merge}(T_1\ \sigma_1^1, T_1\ \sigma_2^1, T_2\ \sigma_1^2, T_2\ \sigma_2^2, T_3, T_4)$

$$\begin{array}{rcl} T(\varphi,R) & = & \text{merge}(Pc, \forall x(Px \to Pfx) +^c Pc \to Pfc +^{fc} Pfc \to Pf^2c \\ & & \vdash Pg^2c, \\ & & (Pc, \forall x(Px \to Pfx) +^{gc} Pgc \to Pfgc +^{fgc} Pfgc \to Pf^2gc \\ & & \vdash Pg^2c, \\ & & Pc, \forall x(Px \to Pfx) \vdash Pg^2c, \\ & & Pc, \forall x(Px \to Pfx) \vdash Pg^2c) \end{array}$$

$$\begin{array}{rcl} T(\varphi,R) & = & Pc, \forall x(Px \to Pfx) \quad +^c Pc \to Pfc, \\ & & \qquad\qquad\qquad\quad +^{fc} Pfc \to Pf^2c, \\ & & \qquad\qquad\qquad\quad +^{gc} Pgc \to Pfgc, \\ & & \qquad\qquad\qquad\quad +^{fgc} Pfgc \to Pf^2gc \quad \vdash Pg^2c \end{array}$$

The Dp function is

$$Dp(T(\varphi, R))) = Pc, Pc \to Pfc, Pfc \to Pf^2c, Pgc \to Pfgc, Pfgc \to Pf^2gc \vdash Pg^2c.$$

3.5.3 Complexity Analysis

In this section we prove that the algorithm EXP_{new} outperforms the old algorithm EXP. In particular we prove that the complexity of EXP_{new} is always better or equal to that of EXP. Then we define an infinite sequence of **LK**-proofs φ_n where the complexity of EXP is cubic in n while that of EXP_{new} is only quadratic. This implies that the computational complexity of EXP cannot be linearly bounded by that of EXP_{new}. The definitions and results of this section are published [LL19]. Our complexity measure will be the maximal logical complexity of objects constructed by the algorithms.

Definition 3.5.16 (size of a sequent)**.** Let $S : A_1, ..., A_n \vdash A_{n+1}, ..., A_m$ be a sequent, then the size of S ($\|S\|$) is

$$\|A_1, ..., A_n \vdash A_{n+1}, ..., A_m\| = \Sigma_{i=1}^m \|A_i\|$$

Definition 3.5.17 (size of an $\mathbf{LK}_=$-proof)**.** Let φ be an $\mathbf{LK}_=$-proof. If φ is an axiom then φ consists of just one node labelled by a sequent S; here we define $\|\varphi\| = \|S\|$. If φ is not an axiom then the end-sequent is a conclusion of a unary or of a binary rule. So we distinguish two cases:

(a) $\varphi =$

$$\dfrac{\varphi'}{S}\ x$$

Then $\|\varphi\| = \|\varphi'\| + \|S\|$.

(b) $\varphi =$

$$\dfrac{\varphi_1 \quad \varphi_2}{S}\ y$$

Then $\|\varphi\| = \|\varphi_1\| + \|\varphi_2\| + \|S\|$.

Definition 3.5.18 (size of an expansion tree)**.** Let E be an expansion tree, then the size of E ($\|E\|$) is inductively defined as follows

$$\begin{array}{lcl} \|E\| & = & \|E\|_f \text{ if } E \text{ is a quantifier-free formula ,} \\ \|\neg E\| & = & 1 + \|E\|, \\ \|E_1 \circ E_2\| & = & 1 + \|E_1\| + \|E_2\|, \circ \in \{\wedge, \vee, \rightarrow\}, \\ \|Qx.E +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n\| & = & 1 + \|E\| + \|E_1\| + \ldots + \|E_n\|, Q \in \{\forall, \exists\}. \end{array}$$

Definition 3.5.19 (size of an s-expansion proof)**.** Let $E : E_1, \ldots, E_n \vdash E_{n+1}, \ldots, E_m$ be an s-expansion proof, then the size of E ($\|E\|$) is

$$\|E_1, \ldots, E_n \vdash E_{n+1}, \ldots, E_m\| = \Sigma_{i=1}^{m} \|E_i\|.$$

In our algorithms EXP and EXP_{new} we do not only construct sequents, formulas and proofs, but also sets of clauses (which are finite sets of atomic sequents). If $\mathscr{C} = \{C_1, \ldots, C_n\}$ we define $\|\mathscr{C}\| = \|C_1\| + \ldots + \|C_n\|$. We call the objects produced by a proof transformation expressions.

Definition 3.5.20 (expression)**.** An *expression* is a formula, a sequent, a proof or a set of clauses.

Now we consider computations as sequences of expressions which are generated by an algorithmic proof transformation. So let A be an algorithm and φ be a proof serving as input to A. Then $E_A(\varphi)$ is the sequence of all expressions generated by A on input φ. Below we define a complexity function induced by A given by the maximal expression generated by A.

Definition 3.5.21. Let A be an algorithm on proofs. Then we define

$$C_A(\varphi) = \max\{\|x\| \mid x \in E_A(\varphi)\}.$$

Theorem 3.5.7. *$C_{\text{EXP}_{new}}(\varphi) \leq C_{\text{EXP}}(\varphi)$ for all proofs φ in* $\mathbf{LK}_=$.

Proof. The first 4 steps of EXP and EXP_{new} are identical. The sum of the sizes of the expansion trees generated by EXP_{new} is smaller or equal to the size of the clausal CERES normal form generated by EXP. □

We show now that EXP_{new} can be asymptotically better that EXP. To this aim we define the following sequence of **LK**-proofs φ_n:

$$\dfrac{\begin{array}{cc} (\omega_n) & (\pi_n) \\ \forall x(Px \rightarrow Pfx) \vdash \forall x(Px \rightarrow Pf^n x) & \forall x(Px \rightarrow Pf^n x), Pa \vdash Pf^{n^2} a \end{array}}{Pa, \forall x(Px \rightarrow Pfx) \vdash Pf^{n^2} a}\, cut$$

where ω_n is:

$$\dfrac{\dfrac{\dfrac{(\psi_n)}{Py, Py \to Pfy, \ldots, Pf^{n-1}y \to Pf^ny \vdash Pf^ny}\to_r}{Py \to Pfy, \ldots, Pf^{n-1}y \to Pf^ny \vdash Py \to Pf^ny}\forall_l(n\times)}{\forall x(Px \to Pfx) \vdash Py \to Pf^ny}\forall_r$$
$$\forall x(Px \to Pfx) \vdash \forall x(Px \to Pf^nx)$$

and π_n is:

$$\dfrac{\dfrac{(\chi_n)}{Pa \to Pf^na, Pf^na \to Pf^{2n}a, ..., Pf^{(n-1)n}a \to Pf^{n^2}a, Pa \vdash Pf^{n^2}a}\forall_l(n\times)}{\forall x(Px \to Pf^nx), Pa \vdash Pf^{n^2}a}$$

Recursive definition of ψ_n: $\psi_0 = Py \vdash Py$. And for $n > 0$ $\psi_n =$

$$\dfrac{\dfrac{Py \vdash Py \qquad \overset{\psi_{n-1}\{y \leftarrow fy\}}{Pfy, Pfy \to Pf^2y, ..., Pf^{n-1}y \to Pf^ny \vdash Pf^ny}}{Py, Py \to Pfy, ..., Pf^{n-1}y \to Pf^ny \vdash Pf^ny}\to_l$$

For our complexity measure we obtain

$$\|\psi_n\| = 2 + 2(n+1) + \|\psi_{n-1}\|$$

note that $\|\psi_{n-1}\| = \|\psi_{n-1}\{y \leftarrow fy\}\|$.

$$C_P(\psi_0) = 2$$

Obviously, there are constants a_1, a_2, b_1, b_2 (all > 0) such that

$$a_1 * n^2 \leq \|\psi_n\| \leq a_2 * (n+1)^2 \text{ and } b_1 * n^2 \leq \|\chi_n\| \leq b_2 * (n+1)^2.$$

Putting things together there are constants $c_1, c_2 > 0$ with

$$c_1 * n^2 \leq \|\varphi_n\| \leq c_2 * (n+1)^2.$$

Now we compute the characteristic clause sets of the φ_n. After elimination of tautologies we get

$$\mathrm{CL}(\varphi_n) = \{C_{1,n} : Py \vdash Pf^ny;\ C_2 : \vdash Pa;\ C_{3,n} : Pf^{n^2}a \vdash\}.$$

Now we compute the resolution refutation. The recursive definition is the following:

γ_1:

$$\frac{\vdash Pa \qquad Pa \vdash Pf^n a}{\vdash Pf^n a}\, R$$

γ_n:

$$\frac{\begin{matrix}(\gamma_{n-1})\\ \vdash Pf^{(n-1)n}a \end{matrix} \qquad Pf^{(n-1)n}a \vdash Pf^{n^2}a}{\vdash Pf^{n^2}a}\, R$$

We obtain $\|\gamma_n\| = \|\gamma_{n-1}\| + 3$. Thus, the resolution schema is δ_n:

$$\frac{\begin{matrix}(\gamma_n)\\ \vdash Pf^{n^2}a \end{matrix} \qquad Pf^{n^2}a \vdash}{\vdash}\, R$$

with substitutions $\{y \leftarrow a\}, ..., \{y \leftarrow f^{(n-1)n}a\}$. We then get

$$\|\delta_n\| = 3n + 2.$$

Concerning the projections we adapt an improved version, *minimal projections*, which is also used in the implementation of CERES. In this form of projection it is sufficient to derive just a subsequent of the end-sequent which reduces the number of weakenings and contractions in the ACNF.

$\varphi_n[C_{1,n}]$:

$$\frac{\begin{matrix}(\psi_n)\\ Py, Py \to Pfy, ..., Pf^{n-1}y \to Pf^n y \vdash Pf^n y \end{matrix}}{Py, \forall x(Px \to Pfx) \vdash Pf^n y}\, \forall_l(nx) + c_l(nx)$$

$\varphi_n[C_2]$ (minimal projection)

$$Pa \vdash Pa$$

$\varphi_n[C_3]$ (minimal projection):

$$Pf^{n^2}a \vdash Pf^{n^2}a$$

Now we can construct the ACNF. φ_n^* is the ACNF-schema:

φ_1^*:

$$\dfrac{\dfrac{Pa \vdash Pa \qquad \overset{(\varphi_n[C_{1,n}\{y \leftarrow a\}])}{Pa, F \vdash Pf^n a}}{Pa, F \vdash Pf^n a}\, cut \qquad \overset{(\varphi_n[C_{1,n}\{y \leftarrow f^n a\}])}{Pf^n a, F \vdash Pf^{2n} a}}{Pa, F \vdash Pf^{2n} a}\, cut + c^*$$

φ_n^*:

$$\dfrac{\dfrac{\overset{(\varphi^*_{n-1})}{\cdots}}{Pa, F \vdash Pf^{(n-1)n} a} \qquad \overset{(\varphi_n[C_{1,n}\{y \leftarrow f^{(n-1)n} a\}])}{Pf^{(n-1)n} a \vdash Pf^{n^2} a}}{Pa, F \vdash Pf^{n^2} a}\, cut + c^* \qquad \dfrac{(\varphi_n[C_{3,n}])}{Pf^{n^2} a \vdash Pf^{n^2} a}}{Pa, F \vdash Pf^{n^2} a}\, cut$$

where $F = \forall x(Px \to Pfx)$. In φ_n^* there are n substitution instances of the proof ψ_n and therefore the size of the ACNF-schema φ_n^* is

$$\|\varphi_n^*\| \geq a_1 * n^3.$$

As $C_{\mathrm{EXP}}(\varphi_n) \geq \|\varphi_n^*\|$ (the algorithm EXP contains the construction of φ_n^*) we finally obtain

$$C_{\mathrm{EXP}}(\varphi_n^*) \geq a_1 * n^3.$$

Therefore every expansion tree extraction from φ_n^* via EXP is at least cubic in n. Now we consider the complexity of our improved method for extraction of expansion trees. We construct the projections first, here we have the complexity $O(n^2)$ (just for $\varphi_n[C_{1,n}]$, otherwise constant). The construction of the refutation δ_n is in $O(n)$. For the construction of the partial expansion proof:

$$\begin{array}{rcll} T(y) & = & \forall x(Px \to Pfx) & +^{y} Py \to Pfy, \\ & & & +^{fy} Pfy \to Pf^2 y, \\ & & & \cdots \\ & & & +^{f^{n-1}y} Pf^{n-1} y \to Pf^n y \quad \vdash \end{array}$$

we obtain

$$\|T(y)\| = 3(n-1) + 4 = 3n + 1.$$

The last step is the computation of

$$\mathrm{merge}(Pa \vdash, T(y)\{y \leftarrow a\}, T(y)\{y \leftarrow f^n a\}, \ldots, T(y)\{y \leftarrow f^{(n-1)n} a\}, \vdash Pf^{n^2} a)$$

$O(n^2)$: concatenate sequents of complexity $3n + 1$ n-times. The last sequent is an expansion proof of φ_n^*. The total expense of EXP_{new} is therefore (k being a constant)

$$C_{\mathrm{EXP}_{new}}(\varphi_n) \leq k * (n+1)^2.$$

Now we put things together and obtain that EXP_{new} is never more expensive than EXP, but EXP cannot be linearly bounded in EXP_{new}:

Theorem 3.5.8. EXP_{new} *outperforms* EXP*, i.e.*

1. $C_{\mathrm{EXP}_{new}}(\varphi) \leq C_{\mathrm{EXP}}(\varphi)$ *for all proofs* φ *in* $\mathbf{LK}_{=}$.
2. *There exists no constant* d *such that for all proofs* φ *in* $\mathbf{LK}_{=}$*:* $C_{\mathrm{EXP}}(\varphi) \leq d * C_{\mathrm{EXP}_{new}}(\varphi)$.

Proof. 1. By Theorem 3.5.7. 2. Assume that such a constant d exists. By the construction of the proofs φ_n above we would obtain

$$\begin{array}{rcl} C_{\mathrm{EXP}}(\varphi_n) & \leq & d * C_{\mathrm{EXP}_{new}}(\varphi_n) \text{ for all } n \text{ and thus} \\ a_1 * n^3 & \leq & d * k * (n+1)^2 \text{ for all } n. \end{array}$$

But $a_1 * n^3 > d * k * (n+1)^2$ almost everywhere and we obtain a contradiction. □

Remark. We have shown that, for all proofs φ in $\mathbf{LK}_{=}$, $C_{\mathrm{EXP}_{new}}(\varphi) \leq C_{\mathrm{EXP}}(\varphi)$ and that a asymptotic speed-up of C_{EXP} via $C_{\mathrm{EXP}_{new}}$ is possible. The problem to define a sharp bound on C_{EXP} in terms of $C_{\mathrm{EXP}_{new}}$ remains open. Our conjecture is that C_{EXP} cannot be exponential in $C_{\mathrm{EXP}_{new}}$, i.e. that there exists a polynomial p such that

$$C_{\mathrm{EXP}}(\varphi) \leq p(C_{\mathrm{EXP}_{new}}(\varphi)) \text{ for all } \varphi \text{ in } \mathbf{LK}_{=}.$$

3.6 A Note on the Proof Theoretic Strength of CERES

In many areas of proof theory proofs are represented by their Herbrand sequents in order to classify and compare them w.r.t. some specific features. For instance, in terms of proof complexity, Herbrand sequents are frequently used to describe the complexity of a proof. More precisely, the complexity of a proof in propositional logic is given by its Herbrand complexity, which is the minimal size of a Herbrand disjunction [BL11]. Another interesting example is the area of proof equality. One can consider proofs to be equal whenever their Herbrand sequents, up to variable renaming, are. Of course, this is only one of many ways to define proof equality. However, given the non-triviality of this topic, comparing proofs by comparing their Herbrand sequents is a frequently used method.

This gives rise to the following problem: Cut-elimination, and hence the construction of Herbrand sequents, is non-confluent. In fact, in [BH11] it was shown that the constructive content of a proof in classical logic is not uniquely determined, but depends on the chosen method for extracting it. As an example one can consider Gentzen's original proof of the cut-elimination theorem, where a set of proof reductions is applied according to a particular strategy, chosen with regard to a general termination proof. At each stage of a cut-elimination process, different cuts can be reduced and furthermore, for a single cut there are different ways to reduce it. The method is non-deterministic and can lead to mathematical differences in the resulting elementary proofs. It is shown that there can be a large number of strongly different cut-free proofs corresponding to a single proof

with cuts via the standard set of proof reductions. The resulting normal forms not only represent pairwise different Herbrand sequents, but the Herbrand sequents of two normal forms also differ in their propositional structure. This means, that we cannot check for proof equality by asking "Given Herbrand sequents S_1 and S_2 of proofs φ_1 and φ_2, check whether φ_1 is equal to φ_2 by checking whether S_1 is equal to S_2".

However, using the method CERES we can obtain an interesting proof theoretic result. As already explained in [BL11], given a proof with cuts φ and a Herbrand sequent S it can be determined with CERES that S can never be extracted from φ. This can be done by comparing S to the structure obtained by the proof projection. So we can conclude that a given Herbrand sequent cannot be obtained from a given proof by any reductive cut-elimination method. It was noted already in [BL11] that this result could also be shown without even eliminating the cuts from the given proof. Indeed, by the results from Section 3.1, the construction of a CERES normal form and hence, of a proof with quantifier-free cuts, is not required for the construction of a Herbrand sequent.

CHAPTER 4

Inductive Structures

Mathematical induction is one of the most important principles in real mathematics, thus any substantial and relevant approach to analyze mathematical proofs has to take into account induction. But in systems with induction rules, essential proof theoretic concepts and transformation become problematic. In particular Gentzen's method of cut-elimination fails for general induction proofs and thus Herbrand's theorem cannot be realized [Tak87]. Roughly speaking, the reason is that in the cut-elimination method à la Gentzen cuts cannot be shifted over induction rules. In this section we give an example for the failure of Gentzen's method. A more detailed analysis of this example can be found in [LPW17], where the authors argued that the CERES method can provide a cut-free representation of the inductive proof. Consider the sequent

$$S\colon \forall y(P(x) \to P(f(x))) \vdash \forall n \forall x((P(\hat{f}(n,x)) \to P(g(n,x))) \to (P(x) \to P(g(n,x))))$$

where g is a binary function symbol, f is a unary function symbol and

$$\mathscr{E} = \{\hat{f}(0,x) = x, \quad \hat{f}(s(n),x) = f(\hat{f}(n,x))\}.$$

S is not valid in pure first-order logic and does not have a Herbrand sequent w.r.t. the theory $\mathscr{E}$ and hence cannot be proven without induction. Therefore, there is no proof of S. We need the following inductive lemma

$$\forall x(P(x) \to P(f(x))) \vdash \forall n \forall x(P(x) \to P(\hat{f}(n,x))).$$

A proof φ of this lemma in an extension of **LK** with a induction rule could be

$$\dfrac{(\varphi_1) \quad \vdash \forall x P(x) \to P(\hat{f}(0,x)) \qquad \dfrac{(\varphi_2) \quad F_1, \forall x(P(x) \to P(\hat{f}(k,x))) \vdash \forall x(P(x) \to P(\hat{f}(s(k),x)))}{F_1, \forall x(P(x) \to P(\hat{f}(0,x))) \vdash \forall x(P(x) \to P(\hat{f}(m,x)))}\, ind}{F_1, \forall x(P(x) \to P(\hat{f}(0,x))) \vdash \forall n \forall x(P(x) \to P(\hat{f}(n,x)))}\, \forall_r}{\forall x(P(x) \to P(f(x))) \vdash \forall n \forall x(P(x) \to P(\hat{f}(n,x)))}\, cut$$

where $F_1 = \forall x(P(x) \to P(f(x)))$. We omit the definitions of φ_1 and φ_2 (they are simple proofs without induction rules). We define π as

$$\dfrac{\dfrac{(\varphi)}{\forall x(P(x) \to P(f(x))) \vdash F_2} \qquad \dfrac{(\pi_1)}{F_2 \vdash \forall n \forall x((P(\hat{f}(n,x)) \to P(g(n,x))) \to (P(x) \to P(g(n,x))))}}{\forall x(P(x) \to P(f(x))) \vdash \forall n \forall x((P(\hat{f}(n,x)) \to P(g(n,x))) \to (P(x) \to P(g(n,x))))}\, cut$$

where $F_2 = \forall n \forall x(P(x) \to P(\hat{f}(n,x)))$. π_1 is a proof without induction of the form

$$\dfrac{\dfrac{\dfrac{(\pi_1')}{P(u) \to P(\hat{f}(i,u)) \vdash (P(\hat{f}(i,u)) \to P(g(i,u))) \to (P(u) \to P(g(i,u)))}\, \forall_l^*}{\forall n \forall x(P(x) \to P(\hat{f}(n,x))) \vdash (P(\hat{f}(i,u)) \to P(g(i,u))) \to (P(u) \to P(g(i,u)))}\, \forall_r^*}{\forall n \forall x(P(x) \to P(\hat{f}(n,x))) \vdash \forall n \forall x((P(\hat{f}(n,x)) \to P(g(n,x))) \to (P(x) \to P(g(n,x))))}$$

Using Gentzen's method of cut-elimination, we locate the place in the proof where $\forall n$ is introduced. In π_1 $\forall n \forall x(P(x) \to P(\hat{f}(n,x)))$ is obtained from $\forall x(P(x) \to P(\hat{f}(i,x)))$ by $\forall_l$. In φ we may delete the $\forall_r$ inference yielding the cut-formula and replace m by i. But in the attempt to eliminate the formula $\forall x(P(x) \to P(\hat{f}(i,x)))$ in φ we get stuck, as we cannot cross the *ind* rule. Note that also *ind* cannot be eliminated as i is a variable. This problem is not due to the specific form of φ nor of *ind*. In fact, there exists no proof of S with only atomic cuts, even if *ind* is used. Induction on the formula

$$\forall n \forall x((P(\hat{f}(n,x)) \to P(g(n,x))) \to (P(x) \to P(g(n,x))))$$

fails. To prove the end-sequent and inductive lemma is needed, i.e. something which implies $\forall n \forall x(P(x) \to P(\hat{f}(n,x)))$ and cannot be eliminated.

We also want to mention that there are methods for performing cut-elimination in presence of induction [BS11, MM00]. However, the resulting proofs do not have the subformula property and Herbrand's theorem cannot be realized. If induction is represented via schemata of proofs (see e.g. [DLRW13, LPW17]) schematic cut-elimination methods can be defined which allow the extraction of so-called *Herbrand systems*, i.e. a generalization of Herbrand's theorem to schematic proofs. An automated proof analysis using schemata was performed on Fürstenberg's proof of the infinitude of primes using topological concepts (see [F55̈, BHL+08]). Fürstenberg's proof was formalized as a sequence of proofs indexed by the number of primes assumed to exist and the method CERES was applied to the entire sequence. The analysis was performed in a semi-automated way; in fact, major parts of the analysis had to be performed by hand. Nevertheless, the analysis showed that from Fürstenberg's proof Euclid's elementary proof could be obtained. Though a fully automated analysis of this proof is not yet within reach, this example reveals the need for the development of a formal language for analyzing proofs with induction. Recent developments based on schematic CERES can be considered as a first

step in this direction. However, existing schematic calculi with a variant of Herbrand's theorem are either defined for a weak induction principle [LPW17], or do not guarantee a fully automated transformation and analysis of the given proof [DLRW13]. A main contribution of this work is the development of a novel schematic CERES method (see Chapter 5) which simplifies and generalizes existing methods. Before we can define the new method, we first need to introduce the schematic language and the concept of schematic derivations.

4.1 Schematic Language

For the most part of this section we will follow the definitions introduced in [CLL19]. In [CLL19] the focus was the development of a schematic RPL_0^Ψ calculus, which is used to refute schematic quantifier-free formulas. In this work however we also work with schematic formulas that are not quantifier-free (these formulas may occur in input proofs for schematic CERES). Therefore, we will distinguish between *input formula schemata* and general formula schemata.

We work in a two-sorted version of classical first-order logic. The first sort we consider is ω, in which every ground term normalizes to a *numeral* , i.e. a term inductively constructable by $N \Rightarrow s(N) \mid 0$, such that $s(N) \neq 0$ and $s(N) = s(N') \rightarrow N = N'$. Numerals will be denoted by lower-case Greek letters (α, β, γ, etc); for the numeral $s^\alpha 0$ and $\alpha \in \mathbb{N}$ we write $\bar{\alpha}$. The set of numerals is denoted by *Num* . Furthermore, the ω sort includes a countable set of variables $\mathscr{N}$ called parameters . We denote parameters by $n, m, n_1, n_2, \ldots, m_1, m_2, \ldots$. The set of parameters occurring in an expression E is denoted by $\mathscr{N}(E)$.

The second sort, the ι-sort (individuals), is a standard first-order term language extended by *defined function symbols*. Defined function symbols, i.e. primitive recursively defined functions, will be denoted with $\hat{\cdot}$.

We consider the following types of variables and corresponding infinite sets: the set of *individual* variables of type ι denoted by V , the set of *global* variables of type $\omega^\alpha \rightarrow \iota$ denoted by $V^G = \bigcup_{i=1}^{\infty} V_i^G$, where V_i^G are the i-ary global variables of type $\omega^i \rightarrow \iota$ and the set of formula variables of type o denoted by V^F . The set of individual variables V^ι is then defined as $\{X(\bar{\alpha}) \mid X \in V_i^G, \bar{\alpha} \in \omega^i \text{ for all } i \in \mathbb{N}\}$. Let $\alpha, \beta \in \omega$ and $X, Y \in V^G$ then we define $X(\bar{\alpha}) = Y(\bar{\beta})$ iff $X = Y$ and $\bar{\alpha} = \bar{\beta}$.

For terms we consider the set of function symbols of type τ, $\mathscr{F}^\tau$. The set of *defined function symbols* of type τ is denoted by $\hat{\mathscr{F}}^\tau$. The types τ are either of the form $\omega^\alpha \rightarrow \omega$ (for $\alpha \in \omega$) which we call *numeric types* or of type $\iota^\alpha \times \omega^\beta \rightarrow \iota$ for $\alpha > 0$ which we call *individual types*. We distinguish $\hat{\mathscr{F}}_\omega$ - the set of all defined function symbols of numeric type and $\hat{\mathscr{F}}_\iota$ - the set of all defined function symbols of individual type. We define $\mathscr{F}^\omega = \{\bar{0}\}$, $\mathscr{F}^{\omega \rightarrow \omega} = \{s\}$, $\mathscr{F}^\tau = \emptyset$ for all other numeric types τ. For all other types the sets $\mathscr{F}^\tau$ are infinite; moreover all sets $\hat{\mathscr{F}}^\tau$ for $\tau \neq \iota$ are infinite, $\hat{\mathscr{F}}^\iota = \emptyset$. The

symbols in $\hat{\mathscr{F}}_\omega$ and $\hat{\mathscr{F}}_\iota$ are partially ordered by $<_{\hat{\mathscr{F}}}$ where $<_{\hat{\mathscr{F}}}$ is irreflexive, transitive and Noetherian.

We define a similar signature for predicate symbols of type τ, where $\mathscr{P}_i^\tau$ is the (infinite) set of predicate symbols of type τ; the set of *defined predicate symbols* of type τ is denoted by $\hat{\mathscr{P}}^\tau$. For ordinary (α-ary) predicate symbols the types are $\iota^\alpha \to o$ as usual. The symbols in $\hat{\mathscr{P}}$ are partially ordered by $<_{\hat{\mathscr{P}}}$ where $<_{\hat{\mathscr{P}}}$ is irreflexive, transitive and Noetherian.

For the term language we consider *ω-terms* of type ω and *ι-terms* of type ι. Both term sets are defined via function symbols and defined function symbols.

Definition 4.1.1 (ω-terms T^ω)**.**

1. $\bar{0} \in T^\omega$, $\mathscr{N} \subseteq T^\omega$, and if $t \in T^\omega$ then $s(t) \in T^\omega$,
2. if $\hat{f} \in \hat{\mathscr{F}}_\omega^\tau$ for $\tau = \omega^\alpha \to \omega$ and $t_1, \ldots, t_\alpha \in T^\omega$ then $\hat{f}(t_1, \ldots, t_\alpha) \in T^\omega$.

The set T_0^ω denotes terms constructed using (1). Note that the set of parameter-free terms in T_0^ω is *Num*, the set of numerals.

For every defined function symbol $\hat{f} \in \hat{\mathscr{F}}_\omega$ there exists a set of defining equations $D(\hat{f})$ which expresses a primitive recursive definition of $\hat{f}$.

Definition 4.1.2 (defining equations for numeric function symbols)**.** For every $\hat{f} \in \hat{\mathscr{F}}_\omega$, $\hat{f}\colon \omega^{\alpha+1} \to \omega$ we define a set $D(\hat{f})$ consisting of two equations.

Let $\hat{f}$ be minimal in $<_{\hat{\mathscr{F}}}$ and $\hat{f}\colon \omega^{\alpha+1} \to \omega$. Then $D(\hat{f})$ consists of the equations

$$\hat{f}(n_1, \ldots, n_\alpha, \hat{0}) = t_B, \quad \hat{f}(n_1, \ldots, n_\alpha, s(m)) = t_S\{k \leftarrow \hat{f}(n_1, \ldots, n_\alpha, m)\}$$

where for minimal $\hat{f}$ $t_B, t_S \in T_0^\omega$, for nonminimal $\hat{f}$ $t_B, t_S \in T^\omega$ where t_B, t_S may contain only defined function symbols smaller than $\hat{f}$ in $<_{\hat{\mathscr{F}}}$. Furthermore $\mathscr{N}(t_B) \subseteq \{n_1, \ldots, n_\alpha\}$, and $\mathscr{N}(t_S) \subseteq \{n_1, \ldots, n_\alpha\} \cup \{m, k\}$.

We define $D(\hat{\mathscr{F}}_\omega) = \bigcup\{D(\hat{f}) \mid \hat{f} \in \hat{\mathscr{F}}_\omega\}$, which is the set of all defining equations in the numeric types.

Example 4.1.1. For $\hat{p} \in \hat{\mathscr{F}}^{\omega\to\omega}$, $D(\hat{p}) = \{\hat{p}(\bar{0}) = \bar{0},\ \hat{p}(s(m)) = m\}$, $t_B = \bar{0}$, $t_s = m$.

Let $\hat{f}, \hat{g} \in \hat{\mathscr{F}}^\tau$ for $\tau = \omega \times \omega \to \omega$, $\hat{f}$ be minimal and $\hat{f} <_{\hat{\mathscr{F}}} \hat{g}$. We define $D(\hat{f})$ as

$$\hat{f}(n, \bar{0}) = t_B, \quad \hat{f}(n, s(m)) = t_S\{k \leftarrow \hat{f}(n, m)\}$$

for $t_B = n$ and $t_S = s(k)$. Then, obviously, $\hat{f}$ defines $+$.

Now we define $D(\hat{g})$ as

$$\hat{g}(n, \bar{0}) = t'_B, \quad \hat{g}(n, s(m)) = t'_S\{k \leftarrow \hat{g}(n, m)\}$$

where $t'_B = \bar{0}$ and $t'_S = \hat{f}(n, k)$. Then $\hat{g}$ defines $*$.

It is easy to see that, given any parameter assignment, all terms in T^ω evaluate to numerals.

Definition 4.1.3 (parameter assignment)**.** A function $\sigma\colon \mathscr{N} \to Num$ is called a *parameter assignment*. σ is extended to terms homomorphically:

- $\sigma(\bar{\beta}) = \bar{\beta}$ for numerals $\bar{\beta}$.
- $\sigma(\hat{f}(t_1, \ldots, t_\alpha)) = \hat{f}(\sigma(t_1), \ldots, \sigma(t_\alpha))$ for $\hat{f}\colon \omega^\alpha \to \omega$ and $t_1, \ldots, t_\alpha \in T^\omega$.

The set of all parameter assignments is denoted by $\mathscr{S}$.

To simplify notation we use the following convention: if $\sigma \in \mathscr{S}$ and $\vec{n} = (n_1, \ldots, n_\alpha)$ we write $\sigma(\vec{n})$ for $(\sigma(n_1), \ldots, \sigma(n_\alpha))$.

Definition 4.1.4 (rewrite system $R(\hat{\mathscr{F}}_\omega)$)**.** Let $\hat{f} \in \hat{\mathscr{F}}_\omega$. Then $R(\hat{f})$ is the set of the following rewrite rules obtained from $D(\hat{f})$:

$$\hat{f}(n_1, \ldots, n_\alpha, \hat{0}) \to t_B, \quad \hat{f}(n_1, \ldots, n_\alpha, s(m)) \to t_S\{k \leftarrow \hat{f}(n_1, \ldots, n_\alpha, m)\}$$

$R(\hat{\mathscr{F}}_\omega) = \bigcup\{R(\hat{f}) \mid \hat{f} \in \hat{\mathscr{F}}_\omega\}$. When a numeric term $s \in T^\omega$ rewrites to t under $R(\hat{\mathscr{F}}_\omega)$ we write $s \to_\omega t$.

Proposition 4.1.1.

- *$R(\hat{\mathscr{F}}_\omega)$ is a canonical rewrite system.*
- *Let $t \in T^\omega$ and $\sigma \in \mathscr{S}$. Then the (unique) normal form of $\sigma(t)$ under $R(\hat{\mathscr{F}}_\omega)$ (denoted by $\sigma(t)\downarrow_\omega$) is a numeral .*

Proof. Straightforward: termination and confluence of $R(\hat{\mathscr{F}}_\omega)$ are well known, see e.g. [BN98]. In particular $\bar{0}, s$ and $R(\hat{\mathscr{F}})$ define a language for computing the set of primitive recursive functions; in particular the recursions are well founded. A formal proof of termination requires double induction on $<_{\hat{\mathscr{F}}}$ and the value of the recursion parameter. □

Definition 4.1.5 (the ι-terms T^ι)**.** The set T^ι is defined inductively as follows:

- all constants of type ι are in T^ι,
- for all $X \in V^G$ of appropriate arity and $\vec{t} \in T_0^\omega$ $X(\vec{t}) \in T^\iota$. We call the expression $X(\vec{t})$ (which is of type ι) a V-term. We define the set of V-terms as $T^\iota_V = \{X(\vec{t}) \mid X \in V^G, \vec{t} \in T_0^\omega\}$,
- if $f \in \mathscr{F}$, $f\colon \iota^\alpha \to \iota$, $s_1, \ldots, s_\alpha \in T^\iota$ then $f(s_1, \ldots, s_\alpha) \in T^\iota$,
- if $\hat{f} \in \hat{\mathscr{F}}$, $\hat{f}\colon \iota^\alpha \times \omega^{\beta+1} \to \iota$, $s_1, \ldots, s_\alpha \in T^\iota$, $t_1, \ldots, t_{\beta+1} \in T^\omega$ then $\hat{f}(s_1, \ldots, s_\alpha, t_1, \ldots, t_{\beta+1}) \in T^\iota$.

The set of all terms in T^ι which contain no defined symbols and neither parameters nor numerals is denoted by T^ι_0. T^ι_0 is a set of "ordinary" first-order terms.

Definition 4.1.6 (defining equations for ι-symbols)**.** Let $\hat{f} \in \hat{\mathscr{F}}^\tau_\iota$ for $\tau = \iota^\alpha \times \omega^{\beta+1} \to \iota$. The defining equations $D(\hat{f})$ are defined below.

$$\begin{aligned} \hat{f}(X(\bar{0}), \dots, X(\bar{\alpha}), n_1, \dots, n_\beta, \bar{0}) &= t_B, \\ \hat{f}(X(\bar{0}), \dots, X(\bar{\alpha}), n_1, \dots, n_\beta, s(m)) &= t_S\{Y(0) \leftarrow \hat{f}(X(\bar{0}), \dots, X(\bar{\alpha}), n_1, \dots, n_\beta, m)\}, \end{aligned}$$

where $X, Y \in V^G, X \neq Y$. For minimal $\hat{f}$ t_B is a term of type ι with $T^\iota_V(t_B) \subseteq \{X(\bar{0}), \dots, X(\bar{\alpha})\}$, $\mathscr{N}(t_B) \subseteq \{n_1, \dots, n_\beta\}$ and t_B contains no defined symbols from $\hat{\mathscr{F}}^\tau$ for nonnumeric types τ. For nonminimal $\hat{f}$, t_B may contain defined symbols $\hat{g}$ of type $\iota^{\alpha'} \times \omega^{\beta'+1} \to \iota$ with $\hat{g} <_{\hat{\mathscr{F}}} \hat{f}$.

t_S is a term of T^ι with $T^\iota_V(t_S) \subseteq \{X(\bar{0}), \dots, X(\bar{\alpha})\} \cup \{Y(0)\}$ and $\mathscr{N}(t_S) \subseteq \{n_1, \dots, n_\beta\} \cup \{m\}$. For all defined symbols $\hat{g}$ of type $\iota^{\alpha'} \times \omega^{\beta'+1} \to \iota$ occurring in t_S we must have $\hat{g} <_{\hat{\mathscr{F}}} \hat{f}$.

Like for the numeric terms we define $D(\hat{\mathscr{F}}_\iota) = \bigcup\{D(\hat{f}) \mid \hat{f} \in \hat{\mathscr{F}}_\iota\}$.

Example 4.1.2. Let $g \in \mathscr{F}^{\omega\to\omega}$ and $\hat{f} \in \hat{\mathscr{F}}^{\iota\times\omega\to\iota}$. We define $D(\hat{f})$ as

$$\hat{f}(X(0), 0) = X(0), \quad \hat{f}(X(0), m+1) = g(\hat{f}(X(0), m)).$$

Here, $t_B = X(0), t_S = g(Y(0))$.

While numeric terms evaluate to numerals under parameter assignments, terms in T^ι evaluate to terms in T^ι_0, i.e. to ordinary first-order terms. Like for the terms in T^ω the evaluation is defined via a rewrite system.

Definition 4.1.7 (rewrite system $R(\hat{\mathscr{F}}_\iota)$)**.** Let $\hat{f} \in \hat{\mathscr{F}}_\iota$. Then $R(\hat{f})$ is the set of the following rewrite rules obtained from $D(\hat{f})$:

$$\begin{aligned} \hat{f}(X(\bar{0}), \dots, X(\bar{\alpha}), n_1, \dots, n_\beta, \bar{0}) &\to t_B, \\ \hat{f}(X(\bar{0}), \dots, X(\bar{\alpha}), n_1, \dots, n_\beta, s(m)) &\to t_S\{Y(0) \leftarrow \hat{f}(X(\bar{0}), \dots, X(\bar{\alpha}), n_1, \dots, n_\beta, m)\}, \end{aligned}$$

$R(\hat{\mathscr{F}}_\iota) = \bigcup\{R(\hat{f}) \mid \hat{f} \in \hat{\mathscr{F}}_\iota\}$.

If a term s rewrites to t under $R(\hat{\mathscr{F}}_\iota)$ we write $s \to_\iota t$.

Proposition 4.1.2. *$R(\hat{\mathscr{F}}_\iota)$ is a canonical rewrite system.*

Proof. That $R(\hat{\mathscr{F}}_\iota)$ is strongly normalizing and locally confluent can be shown in the same way as for $R(\hat{\mathscr{F}}_\omega)$. □

To evaluate a term $t \in T^\iota$ under $\sigma \in \mathscr{S}$ to a numeral we have to combine $\to_\omega$ and $\to_\iota$.

Definition 4.1.8 (evaluation of T^ι)**.** Let $\sigma \in \mathscr{S}$ and $t \in T^\iota$. We define $\sigma(t)\downarrow_\iota$:

- if c is a constants of type ι then $\sigma(c)\downarrow_\iota = c$.
- If $X(\vec{t}) \in T^\iota_V$ and $\vec{t} = t_1, \ldots, t_\alpha$, $t_i \in T^\iota$ for $1 \leq i \leq \alpha$ then $\sigma(X(\vec{t}))\downarrow_\iota = X(\sigma(t_1), \ldots, \sigma(t_\alpha))$. Note that, by definition of T^ι_V $\sigma(t) = \sigma(t)\downarrow_\omega$.
- if $f \in \mathscr{F}$, $f\colon \iota^\alpha \to \iota$, $s_1, \ldots, s_\alpha \in T^\iota$ then

$$\sigma(f(s_1, \ldots, s_\alpha))\downarrow_\iota = f(\sigma(s_1)\downarrow_\iota, \ldots, \sigma(s_\alpha)\downarrow_\iota).$$

- if $\hat{f} \in \hat{\mathscr{F}}$, $\hat{f}\colon \iota^\alpha \times \omega^{\beta+1} \to \iota$, $s_1, \ldots, s_\alpha \in T^\iota$, $t_1, \ldots, t_{\beta+1} \in T^\omega$ then

$$\sigma(\hat{f}(s_1, \ldots, s_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_\iota = \hat{f}(\sigma(s_1)\downarrow_\iota, \ldots, \sigma(s_\alpha)\downarrow_\iota, \sigma(t_1)\downarrow_\omega, \ldots, \sigma(t_{\beta+1})\downarrow_\omega)\downarrow_\iota .$$

Under a parameter assignment every term in T^ι evaluates to a first-order term:

Proposition 4.1.3. *Let $t \in T^\iota$ and $\sigma \in \mathscr{S}$ then $\sigma(t)\downarrow_\iota \in T^\iota_0$.*

Proof. By induction on the complexity of the term definition and the fact that $\to_\iota$ and $\to_\omega$ are both terminating and confluent. For instance, let us consider the case $\sigma(t_0) = \sigma(\hat{f}(s_1, \ldots, s_\alpha, t_1, \ldots, t_\beta))\downarrow_\iota$ defined above. By induction $\sigma(s_i)\downarrow_\iota \in T^\iota_0$ and we know from Proposition 4.1.1 that $\sigma(t_j)\downarrow_\omega$ are numerals. So there are $t'_1, \ldots, t'_\alpha \in T^\iota_0$ and $p_1, \ldots, p_\beta \in \mathit{Num}$ such that $\sigma(t_0)$ reduces to

$$t'_0\colon \hat{f}(t'_1, \ldots, t'_\alpha, p_1, \ldots, p_\beta)\downarrow_\iota .$$

By induction on the value of p_β we can easily show that $t'_0 \in T^\iota_0$. □

Example 4.1.3. Let

$$\hat{f}(X(0), 0) = X(0), \quad \hat{f}(X(0), m+1) = g(\hat{f}(X(0), m)).$$

and $\sigma(n) = \bar{1}$, $\sigma(m) = \bar{2}$, $\sigma(k) = \bar{0}$ for $k \notin \{n, m\}$. Then

$$\begin{aligned}&\sigma(g(\hat{f}(X(n), m))\downarrow_\iota = g(\sigma(\hat{f}(X(n), m)\downarrow_\iota)) = g(\hat{f}(\sigma(X(n))\downarrow_\iota, \sigma(m)\downarrow_\omega)\downarrow_\iota) = \\ &g(\hat{f}(X(\bar{1}), s(s(\bar{0})))\downarrow_\iota) = g(g(\hat{f}(X(\bar{1}), s(\bar{0}))\downarrow_\iota)) = g(g(g(\hat{f}(X(\bar{1}), \bar{0})\downarrow_\iota))) \\ &= g(g(g(X(\bar{1})))).\end{aligned}$$

Substitutions on term schemata need to be schematic as well, particularly when we are interested in unification. We develop some formal tools below to describe such schemata.

Definition 4.1.9. Let $s_1, s_2 \in T^\omega_0$. Then s_1, s_2 are called *essentially distinct* if for all $\sigma \in \mathscr{S}$ $s_1\sigma \neq s_2\sigma$.

Example 4.1.4. n and $s(n)$ are essentially distinct and so are $\bar{0}$ and $s(n)$. m and $s(n)$ are not essentially distinct (just use σ with $\sigma(m) = \bar{1}$ and $\sigma(n) = \bar{0}$).

Definition 4.1.10 (s-substitution)**.** Let Θ be a finite set of pairs $(X(s_1,\ldots,s_i),t)$ where $X(s_1,\ldots,s_i)\in T^\iota_V$ and $t\in T^\iota$. Θ is called an *s-substitution* if for all $(X(s_1,\ldots,s_i),t)$, $(Y(s'_1,\ldots,s'_i),t')\in\Theta$ either $X\neq Y$ or s_j,s'_j, for some $j\in\{1,\ldots,i\}$, are essentially distinct. For $\sigma\in\mathscr{S}$ we define $\Theta[\sigma]=\{X(s_1\sigma,\ldots,s_i\sigma)\leftarrow t\sigma\downarrow_\iota|\ (X(s_1,\ldots,s_i),t)\in\Theta\}$. We define $dom(\Theta)=\{X(s_1,\ldots,s_i)\mid(X(s_1,\ldots,s_i),t)\in\Theta\}$ and $rg(\Theta)=\{t\mid(X(s_1,\ldots,s_i),t)\in\Theta\}$.

Proposition 4.1.4. *For all $\sigma\in\mathscr{S}$ and every s-substitution Θ $\Theta[\sigma]$ is a (first-order) substitution.*

Proof. It is enough to show that for all $(X(s_1,\ldots,s_i),t),(Y(s'_1,\ldots,s'_i),t')\in\Theta$ $X(s_1\sigma,\ldots,s_i\sigma)\neq Y(s'_1\sigma,\ldots,s'_i\sigma)$ for all $\sigma\in\mathscr{S}$. If $X\neq Y$ this is obvious; if $X=Y$ then, by definition of Θ, the pairs (s_1,s'_1), …, (s_i,s'_i) are essentially distinct and so $s_1\sigma\neq s'_1\sigma$, …, $s_i\sigma\neq s'_i\sigma$. Then $\Theta[\sigma]$ is indeed a substitution as for $X(s_1,\ldots,s_i)\in T^\iota_V$ $X(s_1\sigma,\ldots,s_i\sigma)\in V^\iota$. □

The application of an s-substitution Θ to terms in T^ι is defined inductively on the complexity of term definitions as usual.

Definition 4.1.11. Let Θ be an s-substitution. We define $t\Theta$ for terms $t\in T^\iota$.

- if c is a constants of type ι then $c\Theta=c$,
- if $X(s_1,\ldots,s_i)\in T^\iota$ and $X(s_1,\ldots,s_i)\notin dom(\Theta)$ then $X(s_1,\ldots,s_i)\Theta=X(s_1,\ldots,s_i)$; if $X(s_1,\ldots,s_i)\in dom(\Theta)$ and $(X(s_1,\ldots,s_i),t)\in\Theta$ then $X(s_1,\ldots,s_i)\Theta=t$,
- if $f\in\mathscr{F}$, $f\colon\iota^\alpha\to\iota$, $s_1,\ldots,s_\alpha\in T^\iota$ then $f(s_1,\ldots,s_\alpha)\Theta=f(s_1\Theta,\ldots,s_\alpha\Theta)$,
- if $\hat f\in\hat{\mathscr{F}}$, $\hat f\colon\iota^\alpha\times\omega^{\beta+1}\to\iota$, $s_1,\ldots,s_\alpha\in T^\iota$, $t_1,\ldots,t_{\beta+1}\in T^\omega$ then

$$\hat f(s_1,\ldots,s_\alpha,t_1,\ldots,t_{\beta+1})\Theta=\hat f(s_1\Theta,\ldots,s_\alpha\Theta,t_1,\ldots,t_{\beta+1}).$$

The composition of *s*-substitutions is not trivial as, in general, there is no uniform representation of composition under varying parameter assignments.

Example 4.1.5. Let

$$\Theta_1=\{(X_1(n),f(X_1(n))\}\ \Theta_2=\{(X_1(0),g(a))\}.$$

Then, for $\sigma\in\mathscr{S}$ such that $\sigma(n)=0$ we get

$$\Theta_1[\sigma]\circ\Theta_2[\sigma]=\{X_1(0)\leftarrow f(X_1(0))\}\circ\{X_1(0)\leftarrow g(a)\}=\{X_1(0)\leftarrow f(g(a))\}.$$

On the other hand, for $\sigma'\in\mathscr{S}$ with $\sigma'(n)=1$ we obtain

$$\begin{aligned}\Theta_1[\sigma']\circ\Theta_2[\sigma'] &= \{X_1(1)\leftarrow f(X_1(1))\}\circ\{X_1(0)\leftarrow g(a)\}\\ &= \{X_1(1)\leftarrow f(X_1(1)),X_1(0)\leftarrow g(a)\}.\end{aligned}$$

Or take $\Theta'_1 = \{(X_1(n), X_2(n))\}$ and $\Theta'_2 = \{(X_2(m), X_1(m))\}$.
Let $\sigma(n) = \sigma(m) = 0$ and $\sigma'(n) = 0, \sigma'(m) = 1$. Then

$$\begin{aligned} \Theta'_1[\sigma] \circ \Theta'_2[\sigma] &= \{X_2(0) \leftarrow X_1(0)\}, \\ \Theta'_1[\sigma'] \circ \Theta'_2[\sigma'] &= \{X_1(0) \leftarrow X_2(0), X_2(1) \leftarrow X_1(1)\}. \end{aligned}$$

The examples above suggest the following restrictions on s-substitutions with respect to composition. The first definition ensures that domain and range are variable-disjoint.

Definition 4.1.12. Let Θ be an s-substitution. Θ is called *normal* if for all $\sigma \in \mathscr{S}$ $dom(\Theta[\sigma]) \cap V^\iota(rg(\Theta[\sigma])) = \emptyset$.

Example 4.1.6. The substitutions Θ'_1 and Θ'_2 in Example 4.1.5 are normal. Θ_1 in Example 4.1.5 is not normal.

Proposition 4.1.5. *It is decidable whether a given s-substitution is normal.*

Proof. Let Θ be an s-substitution. We search for equal global variables in $dom(\Theta)$ and in $rg(\Theta)$; if there are none then Θ is trivially normal. So let $X \in V^G(dom(\Theta)) \cap V^G(rg(\Theta))$. For every $X(\vec{s}) \in dom(\Theta)$ and for every $X(\vec{t})$ occurring in $rg(\Theta)$ we test whether there exists a $\sigma \in \mathscr{S}$ such that $\sigma(\vec{s}) = \sigma(\vec{t})$. This test uses ordinary first-order unification on terms in T_0^ω. When we find $X(\vec{s}), X(\vec{t})$ such that there exists a $\sigma \in \mathscr{S}$ with $\sigma(\vec{s}) = \sigma(\vec{t})$ then Θ is not normal, and normal otherwise. □

Example 4.1.5 shows also that normal s-substitutions cannot always be composed to an s-substitution; thus we need an additional condition.

Definition 4.1.13. Let Θ_1, Θ_2 be normal s-substitutions. (Θ_1, Θ_2) is called *composable* if for all $\sigma \in \mathscr{S}$

1. $dom(\Theta_1[\sigma]) \cap dom(\Theta_2(\sigma)) = \emptyset$,
2. $dom(\Theta_1[\sigma]) \cap V^\iota(rg(\Theta_2[\sigma])) = \emptyset$.

Proposition 4.1.6. *It is decidable whether, for two normal s-substitutions Θ_1, Θ_2, (Θ_1, Θ_2) is composable.*

Proof. Like in Proposition 4.1.5: by unification tests on $X(\vec{s}), X(\vec{t})$ occurring in the sets under consideration. □

Definition 4.1.14. Let Θ_1, Θ_2 be normal s-substitutions and (Θ_1, Θ_2) composable. Assume that

$$\begin{aligned} \Theta_1 &= \{(X_1(\vec{s_1}), \vec{t_1}), \ldots, (X_\alpha(\vec{s_\alpha}), \vec{t_\alpha})\}, \\ \Theta_2 &= \{(Y_1(\vec{w_1}), \vec{r_1}), \ldots, (Y_\beta(\vec{w_\beta}), \vec{r_\beta})\}. \end{aligned}$$

Then the composition $\Theta_1 \star \Theta_2$ is defined as

$$\{(X_1(\vec{s_1}), \vec{t_1}\Theta_2), \ldots, (X_\alpha(\vec{s_\alpha}), \vec{t_\alpha}\Theta_2), (Y_1(\vec{w_1}), \vec{r_1}), \ldots, (Y_\beta(\vec{w_\beta}), \vec{r_\beta})\}.$$

The following proposition shows that $\Theta_1 \star \Theta_2$ really represents composition.

Proposition 4.1.7. *Let Θ_1, Θ_2 be normal s-substitutions and (Θ_1, Θ_2) be composable then for all $\sigma \in \mathscr{S}$ $(\Theta_1 \star \Theta_2)[\sigma] = \Theta_1[\sigma] \circ \Theta_2[\sigma]$.*

Proof. Let

$$\begin{aligned} \Theta_1 &= \{(X_1(\vec{s_1}), \vec{t_1}), \ldots, (X_\alpha(\vec{s_\alpha}), \vec{t_\alpha})\}, \\ \Theta_2 &= \{(Y_1(\vec{w_1}), \vec{r_1}), \ldots, (Y_\beta(\vec{w_\beta}), \vec{r_\beta})\}. \end{aligned}$$

Then $\Theta_1 \star \Theta_2$ is defined as

$$\{(X_1(\vec{s_1}), \vec{t_1}\Theta_2), \ldots, (X_\alpha(\vec{s_\alpha}), \vec{t_\alpha}\Theta_2), (Y_1(\vec{w_1}), \vec{r_1}), \ldots, (Y_\beta(\vec{w_\beta}), \vec{r_\beta})\}.$$

Let $\sigma(\vec{s_i}) = \vec{\gamma_i}, \sigma(\vec{w_j}) = \vec{\delta_j}$, $\vec{t_i}\sigma{\downarrow_\iota} = \vec{t'_i}$ and $\vec{r_j}\sigma{\downarrow_\iota} = \vec{r'_j}$. We write x_i for $X_i(\vec{\gamma_i})$ and y_j for $Y_j(\vec{\delta_j})$ and $\theta_1 = \Theta_1[\sigma], \theta_2 = \Theta_2[\sigma]$. Then

$$\begin{aligned} \theta_1 &= \{x_1 \leftarrow \vec{t'_1}, \ldots, x_\alpha \leftarrow \vec{t'_\alpha}\}, \\ \theta_2 &= \{(y_1 \leftarrow \vec{r'_1}, \ldots, y_\beta \leftarrow \vec{r'_\beta}\}. \end{aligned}$$

As (Θ_1, Θ_2) is composable we have

1. $\{x_1, \ldots, x_\alpha\} \cap \{y_1, \ldots, y_\beta\} = \emptyset$, and
2. $\{x_1, \ldots, x_\alpha\} \cap V^\iota(\{\vec{r'_1}, \ldots, \vec{r'_\beta}\}) = \emptyset$.

So

$$\begin{aligned} &\theta_1\theta_2 = \\ &\{x_1 \leftarrow \vec{t'_1}, \ldots, x_\alpha \leftarrow \vec{t'_\alpha})\}\theta_2 = \\ &\{x_1 \leftarrow \vec{t'_1}\theta_2, \ldots, x_\alpha \leftarrow \vec{t'_\alpha}\theta_2\} \cup \theta_2. \end{aligned}$$

The last substitution is just $(\Theta_1 \star \Theta_2)[\sigma]$. □

Proposition 4.1.8. *Let Θ_1, Θ_2 be normal s-substitutions and (Θ_1, Θ_2) composable. Then $\Theta_1 \star \Theta_2$ is normal.*

Proof. Like in the proof of Proposition 4.1.7 let $\Theta_1[\sigma] = \theta_1, \Theta_2[\sigma] = \theta_2$. We have to show that $dom(\theta_1\theta_2) \cap V^\iota(rg(\theta_1\theta_2)) = \emptyset$. We have

$$\theta_1\theta_2 = \{x_1 \leftarrow \vec{t'_1}\theta_2, \ldots, x_\alpha \leftarrow \vec{t'_\alpha}\theta_2\} \cup \theta_2.$$

As θ_1 is normal we have $V^\iota(\vec{t_i'}) \cap \{x_1, \ldots, x_\alpha\} = \emptyset$ for $i = 1, \ldots, \alpha$. By definition of composability $rg(\theta_2) \cap \{x_1, \ldots, x_\alpha\} = \emptyset$, and therefore

$$V^\iota(\{\vec{t_1'}\theta_2, \ldots, \vec{t_\alpha'}\theta_2\}) \cap \{x_1, \ldots, x_\alpha\} = \emptyset.$$

So $\{x_1 \leftarrow \vec{t_1'}\theta_2, \ldots, x_\alpha \leftarrow \vec{t_\alpha'}\theta_2\}$ is normal. As also Θ_2 is normal we have $dom(\theta_2) \cap V^\iota(rg(\theta_2)) = \emptyset$. Hence we obtain $dom(\theta_1\theta_2) \cap V^\iota(rg(\theta_1\theta_2)) = \emptyset$. □

Like for general unifiers, we can define s-unifiers from s-substitutions.

Definition 4.1.15 (s-unifier)**.** Let $t_1, t_2 \in T^\iota$. An s-substitution Θ is called an s-unifier of t_1, t_2 if for all $\sigma \in \mathscr{S}$ $(t_1\sigma{\downarrow_\iota})\Theta[\sigma] = (t_2\sigma{\downarrow_\iota})\Theta[\sigma]$. We refer to t_1, t_2 as s-unifiable if there exists an s-unifier of t_1, t_2. s-unifiability can be extended to more than two terms and to formula schemata (to be defined below) in an obvious way.

Definition 4.1.16. An s-unifier Θ of t_1, t_2 is called *restricted* to $\{t_1, t_2\}$ if $T^\iota_V(\Theta) \subseteq T^\iota_V(\{t_1, t_2\})$.

Remark. It is easy to see that for any s-sunifier Θ of $\{t_1, t_2\}$ there exists an s-unifier Θ' of $\{t_1, t_2\}$ which is *restricted* to $\{t_1, t_2\}$.

In this work we will use two notions of formula schema. The first notion is similar to the one in [LPW17] and used for defining formulas in proof schemata that serve as input proofs to the schematic CERES method. These proof schemata will be simple in the sense that they do not contain global variables. Only after regularizing the proof schemata we introduce global variables, which will be used to define the second concept of formula schema (see Definition 4.1.22). The formula schemata used in input proof schemata are referred to as *input formula schemata*.

Definition 4.1.17 (input formula schemata (FS_{in}))**.** We define the set FS_{in} inductively:

- Let ξ be a formula variable in V^F then $\xi \in \mathrm{FS}_{in}$.
- Let $P\colon \iota^\alpha \to o \in \mathscr{P}$ and $t_1, \ldots, t_\alpha \in T^\iota$. Then $P(t_1, \ldots, t_\alpha) \in \mathrm{FS}_{in}$.
- Let $\hat{P} \in \mathscr{P}^\tau$ for $\tau\colon \iota^\alpha \times \omega^{\beta+1} \to o$, $x_1, \ldots, x_\alpha \in V$, $t_1, \ldots, t_{\beta+1} \in T^\omega$ then $\hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_{\beta+1}) \in \mathrm{FS}_{in}$.
- Let $F \in \mathrm{FS}_{in}$ then $\neg F \in \mathrm{FS}_{in}$.
- If $F_1, F_2 \in \mathrm{FS}_{in}$ then $F_1 \wedge F_2 \in \mathrm{FS}_{in}$, $F_1 \vee F_2 \in \mathrm{FS}_{in}$ and $F_1 \to F_2 \in \mathrm{FS}_{in}$.
- If $F \in \mathrm{FS}_{in}$ then $\forall x F \in \mathrm{FS}_{in}$ and $\exists x F \in \mathrm{FS}_{in}$, where $x \in V$.

The subset of FS_{in} not containing formula variables is denoted by $\mathrm{FS}_{0_{in}}$. The subset of FS_{in} containing no defined symbols at all and neither parameters nor numerals is denoted by F_{0in} . F_{0in} is a set of ordinary first-order formulas.

Definition 4.1.18 (defining equations for input predicate symbols)**.** For every $\hat{P} \in \hat{\mathscr{P}}^\tau$ for $\tau\colon \iota^\alpha \times \omega^\beta \to o$ we define a set $D(\hat{P})$ of defining equations, where $\vec{y} = y_1, \ldots, y_\alpha$ and $\vec{n} = n_1, \ldots, n_\beta$. $D(\hat{P})$ consists of

$$\hat{P}(\vec{y}, \vec{n}, 0) = F_B, \quad \hat{P}(\vec{y}, \vec{n}, s(m)) = F_S\{\xi \leftarrow \hat{P}(\vec{y}, \vec{n}, m)\},$$

where, for a $<_{\hat{\mathscr{P}}}$-minimal $\hat{P}$ $F_B, F_S \in \mathrm{FS}0_{in}$. If $\hat{P}$ is not $<_{\hat{\mathscr{P}}}$-minimal then $F_B, F_S \in \mathrm{FS}_{in}$ such that for every $\hat{Q} \in \hat{\mathscr{P}}$ occurring in F_B, F_S we have $\hat{Q} <_{\hat{\mathscr{P}}} \hat{P}$. The only variables and parameters occurring in F_B are $\vec{y}$ and $\vec{n}$ respectively. The only variables in F_S are $\vec{y}$ and besides $\vec{n}$ F_S may include a formula variable ξ and a parameter m. Like for $\hat{\mathscr{F}}_\omega$ and $\hat{\mathscr{F}}_\iota$ we define

$$D(\hat{\mathscr{P}}) = \bigcup\{D(\hat{P}) \mid \hat{P} \in \hat{\mathscr{P}}\}.$$

Definition 4.1.19. The evaluation of a formula $F \in \mathrm{FS}_{in}$ is denoted by$\downarrow_o$ and is defined inductively. Let $\sigma \in \mathscr{S}$; we define $\sigma(F)\downarrow_o$.

1. Let ξ be a formula variable in V^F then $\sigma(\xi)\downarrow_o = \xi$.
2. Let $P\colon \iota^\alpha \to o \in \mathscr{P}$ and $t_1, \ldots, t_\alpha \in T^\iota$. Then
$$\sigma(P(t_1, \ldots, t_\alpha))\downarrow_o = P(\sigma(t_1)\downarrow_\iota, \ldots, \sigma(t_\alpha)\downarrow_\iota).$$
3. Let $\hat{P} \in \mathscr{P}^\tau$ and $F = \hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_{\beta+1})$. Let $D(\hat{P}) =$
$$\begin{array}{rcl} \hat{P}(x_1, \ldots, x_\alpha, n_1, \ldots, n_\beta, 0) & = & F_B, \\ \hat{P}(x_1, \ldots, x_\alpha, n_1, \ldots, n_\beta, m+1) & = & F_S\{\xi \leftarrow \hat{P}(x_1, \ldots, x_\alpha, n_1, \ldots, n_\beta, m)\}. \end{array}$$
we distinguish two cases:
(a) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{0}$. Then
$$\sigma(\hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_B)\downarrow_o$$
for $F'_B = F_B\{n_1 \leftarrow t_1, \ldots, n_\beta \leftarrow t_\beta\}$.
(b) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{p}$ and $p > 0$. Then
$$\sigma(\hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_S)\downarrow_o .$$
For
$$\begin{array}{rcl} F'_S & = & F_S\{\xi \leftarrow \hat{P}(x_1, \ldots, x_\alpha, n_1, \ldots, n_\beta, m)\} \\ & & \{n_1 \leftarrow t_1, \ldots, n_\beta \leftarrow t_\beta, m \leftarrow \overline{p-1}\}. \end{array}$$
4. $\sigma(\neg F)\downarrow_o = \neg\sigma(F)\downarrow_o$.
5. $\sigma(F_1 \circ F_2)\downarrow_o = \sigma(F_1)\downarrow_o \circ \sigma(F_2)\downarrow_o$ for $\circ \in \{\wedge, \vee, \rightarrow\}$.

6. $\sigma(\forall x F)\downarrow_o = \forall x \sigma(F)\downarrow_o$ and $\sigma(\exists x F)\downarrow_o = \exists x \sigma(F)\downarrow_o$.

Proposition 4.1.9. *Let $F \in \mathrm{FS}_{0in}$ and $\sigma \in \mathscr{S}$. Then $\sigma(F)\downarrow_o \in \mathrm{F}_{0in}$.*

Proof. If there are no defined predicate symbols in F then, obviously, $\sigma(F)\downarrow_o \in \mathrm{F}_{0in}$. Here only the cases (1),(2),(4) and (5) in Definition 4.1.19 apply.

If there are defined predicate symbols we proceed by induction on $<_{\hat{\mathscr{P}}}$ and the induction parameter.

Let $\hat{P}$ be minimal in $<_{\hat{\mathscr{P}}}$ and let $F = \hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_{\beta+1})$. We show that $\sigma(F)\downarrow_o \in \mathrm{F}_{0in}$:

(a) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{0}$. Then, by Definition 4.1.19

$$\sigma(\hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_B)\downarrow_o$$

As $\hat{P}$ is minimal the formula F'_B does not contain defined predicate symbols and so $\sigma(F'_B)\downarrow_o \in \mathrm{F}_{0in}$.

(b) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{p}$ and $p > 0$. Here we have

$$\sigma(\hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_S)\downarrow_o$$

For

$$\begin{aligned} F'_S \quad = \quad & F_S\{\xi \leftarrow \hat{P}(x_1, \ldots, x_\alpha, n_1, \ldots, n_\beta, m)\} \\ & \{n_1 \leftarrow t_1, \ldots, n_\beta \leftarrow t_\beta, m \leftarrow \overline{p-1}\}. \end{aligned}$$

Note that F_S itself does not contain defined predicate symbols; in F'_S we have the symbol $\hat{P}$ but with $\hat{P}(x_1, \ldots, x_\alpha, t_1, \ldots, t_\beta, \overline{p-1})$. Therefore we proceed by induction on the value of $\sigma(t_{\beta+1})$ and infer that also $\sigma(F'_S)\downarrow_o \in \mathrm{F}_{0in}$.

If $\hat{P}$ is not minimal the base case for $\hat{P}$ involves only smaller defined predicate symbols. So by induction on $<_{\hat{\mathscr{P}}}$ we get the desired result. □

In the schematic CERES method we will only consider weak schematic sequents. These are schematic sequents that contain only weak quantifier occurrences. Therefore, we introduce the notion of weak and strong input formula schema.

Definition 4.1.20 (weak input formula schema)**.** Let $F \in \mathrm{FS}_{in}$. Then F is called *weak* if for all $\sigma \in \mathscr{S}$ the formula $\sigma(F)\downarrow_o$ contains only weak quantifiers.

Definition 4.1.21 (strong input formula schema)**.** Let $F \in \mathrm{FS}_{in}$. Then F is called *strong* if for all $\sigma \in \mathscr{S}$ the formula $\sigma(F)\downarrow_o$ contains only strong quantifiers.

The second notion of formula schema we use in this work will be formula schemata defined in a way that also the number of variables in formulas can increase with the assignments of parameters. For this reason we use global variables in the definition. These formula schemata will be used in defining refutation schemata. We do not allow quantification over global variables.

Definition 4.1.22 (formula schemata (FS))**.** We define the set FS inductively:

- Let ξ be a formula variable in V^F then $\xi \in \mathrm{FS}$.
- Let $P\colon \iota^\alpha \to o \in \mathscr{P}$ and $t_1, \ldots, t_\alpha \in T^\iota$. Then $P(t_1, \ldots, t_\alpha) \in \mathrm{FS}$.
- Let $\hat{P} \in \mathscr{P}^\tau$ for $\tau\colon (\omega^{\alpha_1} \to \iota) \times \ldots \times (\omega^{\alpha_i} \to \iota) \times \omega^{\beta+1} \to o$, $X_1, \ldots, X_\alpha \in V^G$, $t_1, \ldots, t_{\beta+1} \in T^\omega$ then $\hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1}) \in \mathrm{FS}$.
- Let $F \in \mathrm{FS}$ then $\neg F \in \mathrm{FS}$.
- If $F_1, F_2 \in \mathrm{FS}$ then $F_1 \wedge F_2 \in \mathrm{FS}$ and $F_1 \vee F_2 \in \mathrm{FS}$.

The subset of FS not containing formula variables is denoted by FS_0 . The subset of FS containing no defined symbols at all and neither parameters nor numerals is denoted by F_0 . F_0 is a set of ordinary quantifier-free first-order formulas.

Definition 4.1.23 (defining equations for predicate symbols)**.** For every $\hat{P} \in \hat{\mathscr{P}}^\tau$ for $\tau\colon (\omega^{\alpha_1} \to \iota) \times \ldots \times (\omega^{\alpha_i} \to \iota) \times \omega^{\beta+1} \to o$ we define a set $D(\hat{P})$ of defining equations, where $\vec{Y} = Y_1, \ldots, Y_\alpha$ and $\vec{n} = n_1, \ldots, n_\beta$. $D(\hat{P})$ consists of

$$\hat{P}(\vec{Y}, \vec{n}, 0) = F_B, \quad \hat{P}(\vec{Y}, \vec{n}, s(m)) = F_S\{\xi \leftarrow \hat{P}(\vec{Y}, \vec{n}, m)\},$$

where, for a $<_{\hat{\mathscr{P}}}$-minimal $\hat{P}$ $F_B, F_S \in \mathrm{FS}_0$. If $\hat{P}$ is not $<_{\hat{\mathscr{P}}}$-minimal then $F_B, F_S \in \mathrm{FS}$ such that for every $\hat{Q} \in \hat{\mathscr{P}}$ occurring in F_B, F_S we have $\hat{Q} <_{\hat{\mathscr{P}}} \hat{P}$. The only global variables and parameters occurring in F_B are $\vec{Y}$ and $\vec{n}$ respectively. The only global variables in F_S are $\vec{Y}$ and besides $\vec{n}$ F_S may include a formula variable ξ and a parameter m. Like for $\hat{\mathscr{F}}_\omega$ and $\hat{\mathscr{F}}_\iota$ we define

$$D(\hat{\mathscr{P}}) = \bigcup\{D(\hat{P}) \mid \hat{P} \in \hat{\mathscr{P}}\}.$$

The evaluation of a formula $F \in \mathrm{FS}$ is denoted by$\downarrow_o$ and is defined inductively.

Definition 4.1.24. Let $\sigma \in \mathscr{S}$; we define $\sigma(F)\downarrow_o$ for $F \in \mathrm{FS}$.

1. Let ξ be a formula variable in V^F then $\sigma(\xi)\downarrow_o = \xi$.
2. Let $P\colon \iota^\alpha \to o \in \mathscr{P}$ and $t_1, \ldots, t_\alpha \in T^\iota$. Then

$$\sigma(P(t_1, \ldots, t_\alpha))\downarrow_o = P(\sigma(t_1)\downarrow_\iota, \ldots, \sigma(t_\alpha)\downarrow_\iota).$$

3. Let $\hat{P} \in \mathscr{P}^\tau$ and $F = \hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1})$. Let $D(\hat{P}) =$

$$\begin{aligned} \hat{P}(X_1, \ldots, X_\alpha, n_1, \ldots, n_\beta, 0) &= F_B, \\ \hat{P}(X_1, \ldots, X_\alpha, n_1, \ldots, n_\beta, m+1) &= F_S\{\xi \leftarrow \hat{P}(X_1, \ldots, X_\alpha, n_1, \ldots, n_\beta, m)\}. \end{aligned}$$

we distinguish two cases:
(a) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{0}$. Then

$$\sigma(\hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_B)\downarrow_o$$

for $F'_B = F_B\{n_1 \leftarrow t_1, \ldots, n_\beta \leftarrow t_\beta\}$.
(b) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{p}$ and $p > 0$. Then

$$\sigma(\hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_S)\downarrow_o .$$

For

$$\begin{aligned} F'_S &= F_S\{\xi \leftarrow \hat{P}(X_1, \ldots, X_\alpha, n_1, \ldots, n_\beta, m)\} \\ &\quad \{n_1 \leftarrow t_1, \ldots, n_\beta \leftarrow t_\beta, m \leftarrow \overline{p-1}\}. \end{aligned}$$

4. $\sigma(\neg F)\downarrow_o = \neg\sigma(F)\downarrow_o$.
5. $\sigma(F_1 \circ F_2)\downarrow_o = \sigma(F_1)\downarrow_o \circ \sigma(F_2)\downarrow_o$ for $\circ \in \{\wedge, \vee\}$.

Proposition 4.1.10. *Let $F \in \mathrm{FS}_0$ and $\sigma \in \mathscr{S}$. Then $\sigma(F)\downarrow_o \in \mathrm{F}_0$.*

Proof. If there are no defined predicate symbols in F then, obviously, $\sigma(F)\downarrow_o \in \mathrm{F}_0$; indeed, here only the cases (1),(2),(4) and (5) in Definition 4.1.24 apply.

If there are defined predicate symbols we proceed by induction on $<_{\hat{\mathscr{P}}}$ and the induction parameter.

Let $\hat{P}$ be minimal in $<_{\hat{\mathscr{P}}}$ and let $F = \hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1})$. We show that $\sigma(F)\downarrow_o \in F_0$:

(a) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{0}$. Then, by Definition 4.1.24

$$\sigma(\hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_B)\downarrow_o$$

As $\hat{P}$ is minimal the formula F'_B does not contain defined predicate symbols and so $\sigma(F'_B)\downarrow_o \in \mathrm{F}_0$.

(b) $\sigma(t_{\beta+1})\downarrow_\iota = \bar{p}$ and $p > 0$. Here we have

$$\sigma(\hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1}))\downarrow_o = \sigma(F'_S)\downarrow_o$$

For

$$\begin{aligned} F'_S &= F_S\{\xi \leftarrow \hat{P}(X_1,\ldots,X_\alpha,n_1,\ldots,n_\beta,m)\} \\ &\quad \{n_1 \leftarrow t_1,\ldots,n_\beta \leftarrow t_\beta, m \leftarrow \overline{p-1}\}. \end{aligned}$$

Note that F_S itself does not contain defined predicate symbols; in F'_S we have the symbol $\hat{P}$ but with $\hat{P}(X_1,\ldots,X_\alpha,t_1,\ldots,t_\beta,\overline{p-1})$. Therefore we proceed by induction on the value of $\sigma(t_{\beta+1})$ and infer that also $\sigma(F'_S)\downarrow_o \in \mathrm{F}_0$.

If $\hat{P}$ is not minimal the base case for $\hat{P}$ involves only smaller defined predicate symbols. So by induction on $<_{\hat{\mathscr{P}}}$ we get the desired result. □

Definition 4.1.25 (unsatisfiable schemata)**.** Let $F \in \mathrm{FS}$. Then F is called *unsatisfiable* if for all $\sigma \in \mathscr{S}$ the formula $\sigma(F)\downarrow_o$ is unsatisfiable.

Example 4.1.7. Let a be a constant symbol of type ι, $P \in \mathscr{P}^{\iota\times\iota\to o}$, $\hat{f}$ as in Example 4.1.3, $\hat{P} \in \hat{\mathscr{P}}^{\tau}$ for $\tau = \iota\times\omega\to o$, and $\hat{Q} \in \hat{\mathscr{P}}^{\tau'}$ for $\tau' = \iota\times\omega\times\omega\to o$. Concerning the ordering we have $\hat{P} <_{\hat{\mathscr{P}}} \hat{Q}$. The defining equations for $\hat{P}$ and $\hat{Q}$ are:

$$\begin{aligned} \hat{P}(X,\bar{0}) &= \neg P(X(\bar{0}),\hat{f}(a,0)), \\ \hat{P}(X,s(n)) &= \hat{P}(X,n)\vee\neg P(X(s(n)),\hat{f}(a,s(n))). \end{aligned}$$

$$\begin{aligned} \hat{Q}(X,Y,n,\bar{0}) &= P(\hat{f}(Y(\bar{0}),\bar{0}),Y(\bar{1}))\wedge\hat{P}(X,n), \\ \hat{Q}(X,Y,n,s(m)) &= P(\hat{f}(Y(\bar{0}),s(m)),Y(\bar{1}))\wedge\hat{P}(X,n). \end{aligned}$$

where X and Y are of type $\omega\to\iota$. It is easy to see that the schema $\hat{Q}(X,Y,n,m)$ is unsatisfiable. We compute $\sigma(\hat{Q}(X,Y,n,m))\downarrow_o$ for σ with $\sigma(m)=\bar{2},\sigma(n)=\bar{3}$:

$$\begin{aligned} &\sigma(\hat{Q}(X,Y,n,m))\downarrow_o = P(\hat{f}(Y(0),2),Y(1))\wedge\sigma(\hat{P}(X,n)) = \\ &P(\hat{f}(Y(0),2),Y(1))\wedge(\hat{P}(X,2)\vee\neg P(X(3),\hat{f}(a,3))) = \\ &P(\hat{f}(Y(0),2),Y(1))\wedge(\hat{P}(X,1)\vee\neg P(X(2),\hat{f}(a,2))\vee\neg P(X(3),\hat{f}(a,3))) = \\ &....P(g(g(Y(0)),Y(1))\wedge(\neg P(X(0),a)\vee\neg P(X(1),g(a))\vee \\ &\neg P(X(2),g(g(a)))\vee\neg P(X(3),g(g(g(a))))). \end{aligned}$$

Note that, for $\sigma(n)=\bar{\alpha}$ the number of different variables in $\sigma(\hat{Q}(X,Y,n,m))\downarrow_o$ is $\alpha+2$; so the number of variables increases with the parameter assignments.

4.2 Proof Schemata

The concept of proof schema was introduced in [DLRW13] and later extended in [LPW17]. The basic idea is, that proof schemata are formalized as an ordered collection of so-called LKS-proofs with certain constraints. The LKS-calculus is based on so-called *proof links*, which are inferences combining proofs of base and step cases. To be more precise, if φ

is a proof symbol with end-sequent $S(x_1, \ldots, x_\alpha)$ and $t_1, \ldots, t_\alpha$ are terms of the same types as $x_1, \ldots, x_\alpha$, then the expression $\varphi(t_1, \ldots, t_\alpha)$ is called a proof link. LKS is then defined as the extension of **LK** by proof links as initial sequents and an equational theory for schematic formulas and terms. We simplify and extend the schematic formalism from [DLRW13, LPW17]. First of all, we omit the definition of proof links by allowing a more flexible formalism with a more general set of axioms. Moreover, we extend the formalism to *multiple parameters* (in [DLRW13] and in [LPW17] only schemata defined via one parameter were admitted) and strongly extend the recursive proof specifications by allowing mutual recursion.

In this section we will introduce a novel notion of schematic derivations and proof schemata. Schematic derivations are constructed using derivations in the calculus **LKE**, which is an extension of **LK** by an equational theory. As before, we will consider parameters, under which the derivations can be evaluated to a simple **LKE**-derivation. As mentioned above, the most interesting feature discussed in this section, compared to concepts of schematic proofs as introduced in [DLRW13, LPW17], is that we allow several parameters instead of only one. First, let us define the concept of schematic sequents.

Definition 4.2.1 (schematic sequents)**.** A *schematic sequent* is a sequent of the form $F_1, \ldots, F_\alpha \vdash G_1, \ldots, G_\beta$ where the F_i and G_j for $1 \leq i \leq \alpha$ and $1 \leq j \leq \beta$ are input formula schemata.

To omit schematic skolemization in the schematic CERES method we will work with weak end-sequents only.

Definition 4.2.2 (weak schematic sequent)**.** Let $S\colon F_1, \ldots, F_\alpha \vdash G_1, \ldots, G_\beta$ be a schematic sequent. S is called *weak* if the F_i for $1 \leq i \leq \alpha$ are strong input formula schemata and the G_j for $1 \leq j \leq \beta$ are weak input formula schemata.

We define the schematic standard axiom set $\mathscr{A}_x = \{S_\gamma \vdash S_\gamma \mid S_\gamma \text{ atomic input formula schema}\}$.

Definition 4.2.3 (**LKE**)**.** Let $\mathscr{E}$ be an equational theory. **LKE** is an extension of **LK** by the $\mathscr{E}$ inference rule $\dfrac{S(t)}{S(t')}\mathscr{E}$ where the term or input formula schema t in the sequent S is replaced by a term or input formula schema t' for $t = t' \in \mathscr{E}$ (or $t \leftrightarrow t' \in \mathscr{E}$).

In the definitions below we will use the schematic standard axiom set $\mathscr{A}_s$.

Definition 4.2.4 (schematic standard axiom set)**.** Let $\mathscr{A}_s$ be the smallest set of schematic sequents that is closed under substitution containing all sequents of the form $A \vdash A$ for arbitrary schematic atomic formulas A. Then $\mathscr{A}_s$ is called the *schematic standard axiom set*.

We assume to have an unordered infinite set of proof symbols $\mathbf{\Delta}^* = \{\delta_0, \delta_1, \ldots\}$ (proof symbols are simple symbols used to distinguish derivations in a schematic derivation).

Roughly speaking, schematic derivations can be understood as a finite set of tuples, where each tuple defines for a proof symbol δ a base-case (for parameter 0) and a step-case (for parameter $m+1$) proof. Therefore, we only consider a finite subset $\mathbf{\Delta}$ of $\mathbf{\Delta}^*$ and define a partial ordering $<$ on the proof symbols in $\mathbf{\Delta}$ such that $<$ is Noetherian and there exists a maximal element δ_0. Initial sequents are either axioms, or they are end-sequents from previously defined (base- or step-case) proofs, which occur in the finite set of tuples. In general, the step-case proof (for parameter $m+1$) for some proof symbol δ uses as initial sequent its own end-sequent, but under parameter assignment m. Evaluating a schematic derivation means that initial sequents, which are no axioms, have to be replaced by their derivations.

As already mentioned, schematic derivations can be defined over several parameters. Therefore, we will consider a vector of global parameters $\vec{n}$ and a local parameter m, which can be interpreted as the *active* parameter, i.e. the parameter over which the induction is performed.

Definition 4.2.5 (parameter replacement)**.** Let $\vec{m}, \vec{n}$ be tuples of parameters. A parameter replacement on $\vec{n}$ with respect to $\vec{m}$ is a replacement substituting every parameter p in $\vec{n}$ by a term t_p, where the parameters of t_p are in $\vec{m}$.

Definition 4.2.6 (schematic **LKE**-deduction)**.** Let $\mathbf{\Delta}$ be a finite ordered subset of $\mathbf{\Delta}^*$ such that there exists a $\delta_0 \in \mathbf{\Delta}$ with $\delta_0 > \delta'$ for all $\delta' \in \mathbf{\Delta}$ such that $\delta' \neq \delta_0$ and, for each δ, let $\vec{n}_\delta$ be a (possibly empty) list of global parameters and m_δ an active parameter. A finite set of tuples

$$\mathscr{D}\colon \{(\delta, \rho(\delta, \vec{n}_\delta, 0), \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$$

is called a *schematic* **LKE**-*deduction* from a finite set of schematic sequents $\mathscr{S}$ if the following conditions hold for every $\delta \in \mathbf{\Delta}$:
There exists a (possibly empty) finite set of sequents $\mathscr{C}(\delta)$ and a sequent $S(\delta)$ such that

1. $\rho(\delta, \vec{n}_\delta, 0)$ is an **LKE**-deduction of $S(\delta)\{m_\delta \leftarrow 0\}$ from $\mathscr{S} \cup \mathscr{C}(\delta)$,
2. $\rho(\delta, \vec{n}_\delta, m_\delta + 1)$ is an **LKE**-deduction of $S(\delta)\{m_\delta \leftarrow m_\delta + 1\}$ from $\{(\delta, \Psi)\colon S(\delta)\} \cup \mathscr{S} \cup \mathscr{C}(\delta)$, where (δ, Ψ) is a label, $S(\delta)$ the end-sequent of δ and Ψ the empty parameter replacement,
3. for all $S' \in \mathscr{C}(\delta)$, $S' = (\delta', \Psi)\colon S(\delta')\Psi$ where (δ', Ψ) is a label, $\delta' \in \mathbf{\Delta}$ with $\delta > \delta'$ and Ψ is a parameter replacement on $\vec{n}_{\delta'}$ w.r.t. $\vec{n}_\delta$ such that the conditions 1. and 2. hold for δ' .

If $\mathscr{S} = \mathscr{A}_s$ we call $\mathscr{D}$ an **LKE**-*proof schema* (or *proof schema*) of $S(\delta_0)$. As input proof schemata for the schematic CERES method we will only consider proof schemata of weak schematic end-sequents $S(\delta_0)$.

Example 4.2.1. In this example we will construct a proof schema based on two proof symbols δ and δ' using only one parameter. It is the same example as in [LPW17], with exception that we use our new formalism. Let us define a proof schema of

$$\forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))$$

where $\hat{f}(0,x) = x$, $\hat{f}(s(n),x) = f(\hat{f}(n,x))$.

First, we construct an **LKE**-proof schema $\mathscr{D}_1 = \{(\delta, \rho(\delta,0), \rho(\delta,n{+}1))\}$, where $S(\delta)\colon \forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(n,x)))$. We define $\rho(\delta,0)$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(0,a)) \vdash P(\hat{f}(0,a))}{P(a) \vdash P(\hat{f}(0,a))}\,\mathscr{E}}{\vdash P(a) \to P(\hat{f}(0,a))}\,{\to_r}}{\vdash \forall x(P(x) \to P(\hat{f}(0,x)))}\,\forall : r}{\forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(0,x)))}\,w_l$$

$\rho(\delta, n+1)$ is defined as follows:

$$\dfrac{(\delta,\emptyset)\colon S(\delta) \qquad (1)}{\forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(n+1,x)))}\,cut, c_l$$

where (1) is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(n+1,a)) \vdash P(\hat{f}(n+1,a))}{P(f(\hat{f}(n,a))) \vdash P(\hat{f}(n+1,a))}\,\mathscr{E}}{P(\hat{f}(n,a)) \vdash P(\hat{f}(n,a)) \qquad P(f(\hat{f}(n,a))) \vdash P(\hat{f}(n+1,a))}\,{\to_l}}{P(\hat{f}(n,a)), P(\hat{f}(n,a)) \to P(f(\hat{f}(n,a))) \vdash P(\hat{f}(n+1,a))}\,\forall_l}{P(a) \to P(a) \qquad P(\hat{f}(n,a)), \forall x(P(x) \to P(f(x))) \vdash P(\hat{f}(n+1,a))}\,{\to_l}}{P(a), P(a) \to P(\hat{f}(n,a)), \forall x(P(x) \to P(f(x))) \vdash P(\hat{f}(n+1,a))}\,{\to_r}}{P(a) \to P(\hat{f}(n,a)), \forall x(P(x) \to P(f(x))) \vdash P(a) \to P(\hat{f}(n+1,a))}\,\forall_l}{\forall x(P(x) \to P(\hat{f}(n,x))), \forall x(P(x) \to P(f(x))) \vdash P(a) \to P(\hat{f}(n+1,a))}\,\forall_r}{\forall x(P(x) \to P(\hat{f}(n,x))), \forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(n+1,x)))}$$

Now we construct the proof schema $\mathscr{D} = \{(\delta', \rho(\delta',n), \rho(\delta',n{+}1))\} \cup \mathscr{D}_1$, where $S(\delta')\colon \forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))$ and $\delta' > \delta$. There is no internal recursion in δ' needed, hence we only define $\rho(\delta',n)$ as follows:

$$\dfrac{(\delta,\emptyset)\colon S(\delta) \qquad (2)}{\forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))}\,cut$$

where (2) is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(n,c)) \vdash P(\hat{f}(n,c)) \qquad P(g(n,c)) \vdash P(g(n,c))}{P(c) \to P(c) \qquad P(\hat{f}(n,c)) \to P(g(n,c)), P(\hat{f}(n,c)) \vdash P(g(n,c))} \to_l}{P(c), P(\hat{f}(n,c)) \to P(g(n,c)), P(c) \to P(\hat{f}(n,c)) \vdash P(g(n,c))} \to_l}{P(\hat{f}(n,c)) \to P(g(n,c)), P(c) \to P(\hat{f}(n,c)) \vdash P(c) \to P(g(n,c))} \to_r}{P(c) \to P(\hat{f}(n,c)) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))} \to_r}{\forall x(P(x) \to P(\hat{f}(n,x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))} \forall_l$$

As our formalism is capable of handling several induction parameters, we can easily formalize common proofs in PA, as for instance commutativity, as a proof schema. In the following examples we will use the standard Peano axioms.

Definition 4.2.7 (Peano axioms)**.** The Peano axioms are defined as

$$\begin{aligned} A1 &: \vdash 0 \neq s(x) \\ A2 &: \vdash s(x) = s(y) \to x = y \\ A3 &: \vdash x + 0 = x \\ A4 &: \vdash x + s(y) = s(x + y) \\ A5 &: \vdash x \times 0 = 0 \\ A6 &: \vdash x \times s(y) = x \times y + x \end{aligned}$$

Frequently, we will denote $s(0)$ as 1 and add the axiom

$$A7 \quad : \quad \vdash s(x) = x + 1$$

which can be proven easily using $A3$, $A4$ and the definition of 1.

Example 4.2.2. In the following examples we will use the schematic standard axiom set extended by Peano axioms and the usual equality rules (denoted by $\mathscr{E}$).

Proof of 0 *is a left-identity*: We define a proof schema of $\vdash 0 + m = m$. Let $\mathscr{D} = \{(\delta, \rho(\delta, 0), \rho(\delta, m+1))\}$ where $S(\delta) = \vdash 0 + m = m$ and we define $\rho(\delta, 0)$ as follows:

$$\dfrac{\dfrac{}{\vdash 0 = 0}}{\vdash 0 + 0 = 0} A3$$

$\rho(\delta, m+1)$ is defined as follows:

$$\dfrac{\dfrac{\dfrac{}{(\delta, \emptyset) \colon S(\delta)}}{\vdash s(0 + m) = s(m)} A2}{0 + s(m) = s(m)} A4$$

Proof of *associativity*: We define a proof schema of $\vdash (a + b) + m = a + (b + m)$. Let $\mathscr{D}_1 = \{(\delta_1, \rho(\delta_1, a, b, 0), \rho(\delta_1, a, b, m+1))\}$ where $S(\delta_1) = \vdash (a + b) + m = a + (b + m)$ and we define $\rho(\delta_1, a, b, 0)$ as follows:

$$\dfrac{\dfrac{\vdash a+b=a+b}{\vdash a+b=a+(b+0)}\,A3}{\vdash (a+b)+0=a+(b+0)}\,A3$$

$\rho(\delta_1, a, b, m+1)$ is defined as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\delta_1,\emptyset)\colon S(\delta_1)}{\vdash s((a+b)+m)=s(a+(b+m))}\,A2}{\vdash s((a+b)+m)=a+s(b+m)}\,A4}{\vdash s((a+b)+m)=a+(b+s(m))}\,A4}{\vdash (a+b)+s(m)=a+(b+s(m))}\,A4$$

To prove commutativity, we first need to define a proof schema of $\vdash m+1=1+m$. Let $\mathscr{D}_2=\{(\delta_2,\rho(\delta_2,0),\rho(\delta_2,m+1))\}\cup\mathscr{D}$, where $\delta_2>\delta$, $S(\delta_2)=\vdash m+1=1+m$ and we define $\rho(\delta_2,0)$ as follows:

$$\dfrac{(\delta,\{m\leftarrow 1\})\colon S(\delta)\{m\leftarrow 1\}}{\vdash 0+1=1+0}\,A3$$

$\rho(\delta_2, m+1)$ is defined as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\delta_2,\emptyset)\colon S(\delta_2)}{\vdash s(m+1)=s(1+m)}\,A2 \qquad \dfrac{\vdash 1+s(m)=1+s(m)}{\vdash s(1+m)=1+s(m)}\,A4}{\vdash s(m+1)=s(0)+s(m)}\,\mathscr{E},def}{\vdash s((m+1)+0)=s(0)+s(m)}\,A3}{\vdash s(s(m)+0)=s(0)+s(m)}\,A7}{\vdash s(m)+s(0)=s(0)+s(m)}\,A4}{\vdash s(m)+1=1+s(m)}\,def$$

Now we define a proof schema of $\vdash n+m'=m'+n$. Let $\mathscr{D}_3=\{(\delta_3,\rho(\delta_3,n,0),\rho(\delta_3,n,m'+1))\}\cup\mathscr{D}_2\cup\mathscr{D}_1$, where $\delta_3>\delta_2$, $\delta_3>\delta_1$, $S(\delta_3)=\vdash n+m'=m'+n$ and we define $\rho(\delta_3,n,0)$ as follows:

$$\dfrac{\dfrac{(\delta,\{m\leftarrow n\})\colon S(\delta)\{m\leftarrow n\}}{\vdash n=0+n}\,\mathscr{E}}{\vdash n+0=0+n}\,A3$$

$\rho(\delta_3, n, m'+1)$ is defined as follows:

$$\dfrac{\begin{array}{l}(\delta_1, \{a \leftarrow n, b \leftarrow m', m \leftarrow 1\}):\\ S(\delta_1)\{a \leftarrow n, b \leftarrow m', m \leftarrow 1\}\\ \vdash n + (m'+1) = (n+m') + 1\end{array} \qquad \dfrac{\dfrac{\dfrac{S(\delta_3)}{\dfrac{\vdash n + m' = m' + n}{\vdash s(n+m') = s(m'+n)}\,A2} \qquad \phi_1}{\vdash s(n+m') = (m'+1) + n}\,\mathscr{E}}{\vdash s((n+m')+0) = (m'+1) + n}\,A3}{\vdash (n+m') + s(0) = (m'+1) + n}\,A4}{\vdash (n+m') + 1 = (m'+1) + n}\,def}{\vdash n + (m'+1) = (m'+1) + n}\,\mathscr{E}$$

where ϕ_1 is

$$\dfrac{\dfrac{\begin{array}{l}(\delta_1, \{a \leftarrow m', b \leftarrow 1, m \leftarrow n\}):\\ S(\delta_1)\{a \leftarrow m', b \leftarrow 1, m \leftarrow n\}\\ \vdash m' + (1+n) = (m'+1) + n\end{array} \qquad \dfrac{(\delta_1, \{m \leftarrow n\}) : S(\delta_2)\{m \leftarrow n\}}{\vdash n+1 = 1+n}\,\mathscr{E}}{\vdash 1+n = n+1}\,\mathscr{E}}{\vdash m' + (n+1) = (m'+1) + n}\,def}{\vdash m' + s(n) = (m'+1) + n}\,A4}{\vdash s(m'+n) = (m'+1) + n}$$

For schematic cut-elimination we need *regular proofs*. To ensure that the schemata will evaluate to regular proofs we have to regularize the schemata themselves. As we are describing infinite sequences of proofs (instead single ones) we introduce global variables as a way to syntactically describe the process of regularization which can always be performed on the proof resulting form normalization of a proof schema. Essentially, global variables describe meta-syntactic properties of proof schema and thus inherently introduce second-order notions into the formalism. Note that this notion of regularization is stronger than the fact that non-schematic proofs can be regularized, we refer to the regularization implied by global variable introduction as *uniform regularization*, i.e. uniform eigenvariable renaming for an infinite sequence of proofs.

Definition 4.2.8 (regular proof schemata)**.** Let $\mathscr{D}$ be a proof schema as in Definition 4.2.6. Let the size of $\vec{n}_\delta$ be γ_δ. We will replace the α eigenvariables in the proofs $\rho(\delta, \vec{n}_\delta, 0)$ and the β eigenvariables in $\rho(\delta, \vec{n}_\delta, m_\delta + 1)$ for $\delta \in \mathbf{\Delta}$ by new variable schemata

$$X_1^\delta(\vec{n}_\delta, 0), \ldots X_\alpha^\delta(\vec{n}_\delta, 0),$$
$$X_1^\delta(\vec{n}_\delta, m_\delta + 1), \ldots X_\beta^\delta(\vec{n}_\delta, m_\delta + 1),$$

for $X_i^\delta(\vec{n}_\delta, 0)\sigma, X_j^\delta(\vec{n}_\delta, m_\delta + 1)\sigma \colon \omega^{\gamma_\delta + 1} \to \omega$ for any parameter assignment σ. For every $\delta \in \mathbf{\Delta}$ we first regularize the proofs $\rho(\delta, \vec{n}_\delta, 0), \rho(\delta, \vec{n}_\delta, m_\delta + 1)$ to $\rho'(\delta, \vec{n}_\delta, 0), \rho'(\delta, \vec{n}_\delta, m_\delta + 1)$. Let us assume that the eigenvariables in $\rho'(\delta, \vec{n}_\delta, 0)$ are $y_1, \ldots, y_\alpha$, the eigenvariables in $\rho'(\delta, \vec{n}_\delta, m_\delta + 1)$ $z_1, \ldots, z_\beta$. Now we apply the substitution

$$\{y_1 \leftarrow X_1^\delta(\vec{n}_\delta, 0), \ldots, y_\alpha \leftarrow X_\alpha^\delta(\vec{n}_\delta, 0)\}$$

to $\rho'(\delta, \vec{n}_\delta, 0)$ (and obtain $\rho''(\delta, \vec{n}_\delta, 0)$) and the substitution

$$\{z_1 \leftarrow X_1^\delta(\vec{n}_\delta, m_\delta + 1), \ldots, z_\beta \leftarrow X_\beta^\delta(\vec{n}_\delta, m_\delta + 1)\}$$

to $\rho'(\delta, \vec{n}_\delta, m_\delta + 1)$ (and obtain $\rho''(\delta, \vec{n}_\delta, m_\delta + 1)$). The new resulting proof schema $\mathscr{D}^r$ with ρ'' instead of ρ is called *regularized*.

In general we call a proof schema $\mathscr{D}$ *regular* if all the proofs $\rho(\delta, \vec{n}_\delta, 0), \rho(\delta, \vec{n}_\delta, m_\delta + 1)$ are regular and the eigenvariables in $\rho(\delta, \vec{n}_\delta, 0)$ are among $X_i^\delta(\vec{n}, 0)$, the eigenvariables in $\rho(\delta, \vec{n}_\delta, m_\delta + 1)$ among $X_i^\delta(\vec{n}, m)$.

Example 4.2.3. Consider the proof schema of Example 4.2.1, where

$$\mathscr{D} = \{(\delta', \rho(\delta', 0), \rho(\delta', n + 1))\} \cup \mathscr{D}_1$$

and $S(\delta')\colon \forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n, c)) \to P(g(n, c))) \to (P(c) \to P(g(n, c)))$.

After regularization we obtain for $\mathscr{D}_1 = \{(\delta, \rho(\delta, 0), \rho(\delta, n+1))\}$, where $S(\delta)\colon \forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(n, x)))$ the regularized proofs $\rho(\delta, 0) =$

$$\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(0, X_1^\delta(0))) \vdash P(\hat{f}(0, X_1^\delta(0)))}{P(X_1^\delta(0)) \vdash P(\hat{f}(0, X_1^\delta(0)))}\,\mathscr{E}}{\vdash P(X_1^\delta(0)) \to P(\hat{f}(0, X_1^\delta(0)))}\,{\to_r}}{\vdash \forall x(P(x) \to P(\hat{f}(0, x)))}\,{\forall : r}}{\forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(0, x)))}\,w_l$$

and $\rho(\delta, n + 1) =$

$$\dfrac{(\delta, \emptyset)\colon S(\delta) \qquad (1)}{\forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(n+1, x)))}\,cut, c_l$$

where (1) is

$$\dfrac{\dfrac{\dfrac{\dfrac{P(X_1^\delta(n+1)) \to P(X_1^\delta(n+1)) \qquad (2)}{P(X_1^\delta(n+1)), P(X_1^\delta(n+1)) \to P(\hat{f}(n, X_1^\delta(n+1))), \forall x(P(x) \to P(f(x))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}\,{\to_l}}{P(X_1^\delta(n+1)) \to P(\hat{f}(n, X_1^\delta(n+1))), \forall x(P(x) \to P(f(x))) \vdash P(X_1^\delta(n+1)) \to P(\hat{f}(n+1, X_1^\delta(n+1)))}\,{\to_r}}{\forall x(P(x) \to P(\hat{f}(n, x))), \forall x(P(x) \to P(f(x))) \vdash P(X_1^\delta(n+1)) \to P(\hat{f}(n+1, X_1^\delta(n+1)))}\,{\forall_l}}{\forall x(P(x) \to P(\hat{f}(n, x))), \forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(n+1, x)))}\,{\forall_r}$$

and (2) is

$$\dfrac{\dfrac{P(\hat{f}(n, X_1^\delta(n+1))) \vdash P(\hat{f}(n, X_1^\delta(n+1))) \qquad \dfrac{P(\hat{f}(n+1, X_1^\delta(n+1))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}{P(f(\hat{f}(n, X_1^\delta(n+1)))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}\,\mathscr{E}}{P(\hat{f}(n, X_1^\delta(n+1))), P(\hat{f}(n, X_1^\delta(n+1))) \to P(f(\hat{f}(n, X_1^\delta(n+1)))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}\,{\to_l}}{P(\hat{f}(n, X_1^\delta(n+1))), \forall x(P(x) \to P(f(x))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}\,{\forall_l}$$

For $\mathscr{D} = \{(\delta', \rho(\delta', 0), \rho(\delta', n{+}1))\} \cup \mathscr{D}_1$, where $S(\delta')\colon \forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))$ and $\delta' > \delta$ we obtain the regularized proof $\rho(\delta', n) =$

$$\dfrac{(\delta, \emptyset)\colon S(\delta) \qquad (2)}{\forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))}\, cut$$

where (2) is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(n,c)) \vdash P(\hat{f}(n,c)) \qquad P(g(n,c)) \vdash P(g(n,c))}{P(\hat{f}(n,c)) \to P(g(n,c)), P(\hat{f}(n,c)) \vdash P(g(n,c))}\to_l}{P(c) \to P(c) \qquad P(c), P(\hat{f}(n,c)) \to P(g(n,c)), P(c) \to P(\hat{f}(n,c)) \vdash P(g(n,c))}\to_l}{P(\hat{f}(n,c)) \to P(g(n,c)), P(c) \to P(\hat{f}(n,c)) \vdash P(c) \to P(g(n,c))}\to_r}{P(c) \to P(\hat{f}(n,c)) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))}\to_r}{\forall x(P(x) \to P(\hat{f}(n,x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))}\,\forall_l$$

We are going to evaluate proof schemata under parameter assignments. If $\mathscr{D}$ is a proof schema and Θ a parameter assignment we will define $\mathscr{D}\sigma\downarrow$ as the pair $(\rho(\delta_0, \vec{n}, 0)\sigma\downarrow, \rho(\delta_0, \vec{n}, m+1)\sigma\downarrow)$. It remains to define the evaluation of the $\rho(\delta, \vec{n}, t)$ for $t = 0$ and $t = m + 1$.

Definition 4.2.9 (evaluation of proof schema)**.** Let $\mathscr{D}$ be a proof schema and σ a parameter assignment. In defining $\mathscr{D}\sigma\downarrow$ we proceed by double induction on the ordering of proof symbols and the assignments σ.

- Let δ be a minimal element in $\mathbf{\Delta}$.
  1. $\sigma(m_\delta) = 0$.
     Then, by definition of a proof schema, $\rho(\delta, \vec{n}_\delta, 0)$ is an **LKE**-proof; so its evaluation is $\rho(\delta, \vec{n}, 0)\sigma\downarrow$.
  2. $\sigma(m_\delta) = \alpha > 0$.
     Evaluate all sequents except the leaves $(\delta, \emptyset)\colon S(\delta)$ under σ. Afterwards replace $(\delta, \emptyset)\colon S(\delta)$ by the **LKE**-proofs $\rho(\delta, \vec{n}_\delta, m_\delta{+}1)\sigma[m_\delta/\alpha{-}1]\downarrow$ where $\sigma[m_\delta/\alpha{-}1]$ is defined as $\sigma[m_\delta/\alpha{-}1](p) = \sigma(p)$ for all $p \neq m_\delta$ and $\sigma[m_\delta/\alpha{-}1](m_\delta) = \alpha{-}1$. The result is an **LKE**-proof $\rho(\delta, \vec{n}_\delta, m_\delta + 1)\sigma\downarrow$.
- $\delta \in \mathbf{\Delta}$ is not minimal.
  1. $\sigma(m_\delta) = 0$.
     Evaluate all sequents except the leaves of the form $(\delta', \Psi)\colon S(\delta')\Psi$ for $\delta > \delta'$ and the corresponding parameter replacement Ψ under σ. Then replace $(\delta', \Psi)\colon S(\delta')\Psi$ by the **LKE**-proof $S(\delta')\Psi\sigma\downarrow$.

2. $\sigma(m_\delta) = \alpha > 0$.
   As for $\rho(\delta, \vec{n}_\delta, 0)$ except for the leaves $(\delta, \emptyset)$: $S(\delta)$ which are replaced by the **LKE**-proofs $\rho(\delta, \vec{n}_\delta, m_\delta + 1)\sigma[m_\delta/\alpha - 1]$.

$\mathscr{D}\sigma\downarrow$ is defined as $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow$ for the $<$-maximal symbol δ_0.

Proposition 4.2.1. *Let $\mathscr{D}$ be a regular proof schema and σ be a parameter assignment. Then $\mathscr{D}\sigma\downarrow$ is regular, i.e. the proofs $\rho(\delta_0, \vec{n}_{\delta_0}, 0)\sigma\downarrow$ and $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma\downarrow$ are regular.*

Proof. As $\mathscr{D}$ is a regular proof schema, the derivations $\rho(\delta_0, \vec{n}_{\delta_0}, 0)$ and $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)$ are regular. Evaluating them under a parameter assignment σ evaluates all sequents and the occurring formulas under σ. As no variables are introduced nor renamed in this process, it is closed under regularity. Thus, $\rho(\delta_0, \vec{n}_{\delta_0}, 0)\sigma\downarrow$ and $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma\downarrow$ are regular proofs. □

4.3 The Resolution Calculus RPL_0^Ψ

The basis of our calculus for refuting formula schemata is the calculus RPL_0 for quantifier-free formulas, introduced in Section 3.2. We extend RPL_0 by rules handling schematic formula definitions. Here, we have to consider another aspect as well: in inductive proofs the use of lemmas is vital, i.e. an ordinary refutational calculus, which has just a weak capacity of lemma generation, may fail to derive the desired invariant. To this aim we extend the calculus by adding some tautological sequent schemata. This will enrich RPL_0, which only decomposes formulas, by the potential to derive more complex formulas. Note that our aim is to use the calculi in an interactive way and not fully automatic, which justifies this process of "anti-refinement".

In extending RPL_0 to a schematic calculus we have to replace unification by s-unification. Formally we have to define how s-substitutions are extended to formula schemata and sequent schemata.

Definition 4.3.1. Let Θ be an s-substitution. We define $F\Theta$ for all quantifier-free $F \in \mathrm{FS}$ which do not contain formula variables.

- Let $P\colon \iota^\alpha \to o \in \mathscr{P}$ and $t_1, \ldots, t_\alpha \in T^\iota$. Then $P(t_1, \ldots, t_\alpha)\Theta = P(t_1\Theta, \ldots, t_\alpha\Theta)$
- Let $\hat{P} \in \mathscr{P}^\tau$ for $\tau\colon (\omega^{\alpha_1} \to \iota) \times \ldots \times (\omega^{\alpha_i} \to \iota) \times \omega^{\beta+1} \to o$, $X_1, \ldots, X_\alpha \in V^G$, $t_1, \ldots, t_{\beta+1} \in T^\omega$ then

$$\hat{P}(X_1, \ldots, X_\alpha, t_1, \ldots, t_{\beta+1})\Theta = \hat{P}(X_1, \ldots, X_\alpha, t_1\Theta \ldots, t_{\beta+1}\Theta).$$

- $(\neg F)\Theta = \neg F\Theta$.

- If $F_1, F_2 \in \mathrm{FS}$ then

$$\begin{aligned}(F_1 \wedge F_2)\Theta &= F_1\Theta \wedge F_2\Theta,\\ (F_1 \vee F_2)\Theta &= F_1\Theta \vee F_2\Theta.\end{aligned}$$

Let $S\colon A_1, \ldots, A_\alpha \vdash B_1, \ldots, B_\beta$ be a sequent schema. Then

$$S\Theta = A_1\Theta, \ldots, A_\alpha\Theta \vdash B_1\Theta, \ldots, B_\beta\Theta.$$

In the resolution rule we have to take care that the sets of variables in $\{A_1, \ldots, A_k\}$ and $\{B_1, \ldots, B_l\}$ are pairwise disjoint. We need a corresponding concept of disjointness for the schematic case.

Definition 4.3.2 (essentially disjoint)**.** Let $\mathscr{A}, \mathscr{B}$ be finite set of schematic variables in T^ι_V. $\mathscr{A}$ and $\mathscr{B}$ are called *essentially disjoint* if for all $\sigma \in \mathscr{S}$ $\mathscr{A}[\sigma] \cap \mathscr{B}[\sigma] = \emptyset$.

Definition 4.3.3 (RPL_0^Ψ)**.** Let Ψ be a schematic formula definition as in Definitions 4.1.22 and 4.1.23 where

$$\hat{P}(\vec{Y}, \vec{n}, 0) = F_B, \quad \hat{P}(\vec{Y}, \vec{n}, s(m)) = F_S\{\xi \leftarrow \hat{P}(\vec{Y}, \vec{n}, m)\},$$

then RPL_0^Ψ is the extension of RPL_0 by the rules

$$\frac{\Gamma \vdash \Delta, \hat{P}(\vec{Y}, \vec{n}, 0)}{\Gamma \vdash \Delta, F_B}\ B\hat{P}r \qquad \frac{\Gamma \vdash \Delta, \hat{P}(\vec{Y}, \vec{n}, s(m))}{\Gamma \vdash \Delta, F_S\{\xi \leftarrow \hat{P}(\vec{Y}, \vec{n}, m)\}}\ S\hat{P}r$$

$$\frac{\hat{P}(\vec{Y}, \vec{n}, 0), \Gamma \vdash \Delta}{F_B, \Gamma \vdash \Delta}\ B\hat{P}l \qquad \frac{\hat{P}(\vec{Y}, \vec{n}, s(m)), \Gamma \vdash \Delta}{F_S\{\xi \leftarrow \hat{P}(\vec{Y}, \vec{n}, m)\}, \Gamma \vdash \Delta}\ S\hat{P}l$$

for the elimination of defined symbols. For the introduction of defined symbols we invert the rules above:

$$\frac{\Gamma \vdash \Delta, F_B}{\Gamma \vdash \Delta, \hat{P}(\vec{Y}, \vec{n}, 0)}\ B\hat{P}r^+ \qquad \frac{\Gamma \vdash \Delta, F_S\{\xi \leftarrow \hat{P}(\vec{Y}, \vec{n}, m)\}}{\Gamma \vdash \Delta, \hat{P}(\vec{Y}, \vec{n}, s(m))}\ S\hat{P}r^+$$

$$\frac{F_B, \Gamma \vdash \Delta}{\hat{P}(\vec{Y}, \vec{n}, 0), \Gamma \vdash \Delta}\ B\hat{P}l^+ \qquad \frac{F_S\{\xi \leftarrow \hat{P}(\vec{Y}, \vec{n}, m)\}, \Gamma \vdash \Delta}{\hat{P}(\vec{Y}, \vec{n}, s(m)), \Gamma \vdash \Delta}\ S\hat{P}l^+$$

We also adapt the resolution rule to the schematic case:

Let $T^\iota_V(\{A_1, \ldots, A_\alpha\}), T^\iota_V(\{B_1, \ldots, B_\beta\})$ be essentially disjoint sets of schematic variables and Θ be an s-unifier of $\{A_1, \ldots, A_\alpha, B_1, \ldots, B_\beta\}$. Then the resolution rule is defined as

$$\frac{\Gamma \vdash \Delta, A_1, \ldots, A_\alpha \quad B_1, \ldots, B_\beta, \Pi \vdash \Lambda}{\Gamma\Theta, \Pi\Theta \vdash \Delta\Theta, \Lambda\Theta}\ res$$

Moreover we add the following tautological sequent schemata (ξ_1, ξ_2 are formula variables): $\xi_1, \xi_2 \vdash \xi_1 \wedge \xi_2$, $\xi_1 \wedge \xi_2 \vdash \xi_1$, $\xi_1 \wedge \xi_2 \vdash \xi_2$, $\xi_1 \vdash \xi_1 \vee \xi_2$, $\xi_2 \vdash \xi_1 \vee \xi_2$, $\vdash \xi_1, \neg\xi_1$, $\xi_1, \neg\xi_1 \vdash$. For comfort we may add arbitrary tautological sequent schemata to increase the flexibility and the practical use of the calculus.

It is easy to see that the added tautology schemata together with the cut rule simulate the logical introduction rules for $\wedge, \vee, \neg$. As *res* generalizes the cut rule this is possible also in RPL_0^Ψ. We could instead have added the introduction rules themselves which is logically equivalent. But note that adding additional tautology schemata (besides these defined above) increases the flexibility of rule specification via "macros".

Note that the refutational completeness of RPL_0^Ψ is not an issue as already RPL_0 is refutationally complete. RPL_0^Ψ is also sound if the defining equations are considered.

Proposition 4.3.1. *Assume that the sequent S is derivable in* RPL_0^Ψ. *Then* $D(\hat{\mathscr{F}}_\omega) \cup D(\hat{\mathscr{F}}_\iota) \cup D(\hat{\mathscr{P}}) \models S$.

Proof. The introduction and elimination rules for defined predicate symbols are sound with respect to $D(\hat{\mathscr{P}})$; the resolution rule (involving s-unification) is sound with respect to $D(\hat{\mathscr{F}}_\omega) \cup D(\hat{\mathscr{F}}_\iota)$. □

Definition 4.3.4. An RPL_0^Ψ derivation ϱ is called a *cut-derivation* if the s-unifiers of all resolution rules are empty.

Remark. A cut-derivation is an RPL_0^Ψ derivation with only propositional rules. Such a derivation can be obtained by combining all unifiers to a global unifier.

In computing global unifiers we have to apply s-substitutions to proofs. However, not every s-substitution applied to a RPL_0^Ψ derivation results in a RPL_0^Ψ derivation again. Just assume that an s-unifier in a resolution is of the form $(X_1(s), X_2(s'))$; if $\Theta = \{(X_1(s), a), (X_2(s'), b)\}$ for different constant symbols a, b then $X_1(s)\Theta$ and $X_2(s')\Theta$ are no longer unifiable and the resolution is blocked.

Definition 4.3.5. Let ρ be a derivation in RPL_0^Ψ which does not contain the resolution rule; then for any s-substitution Θ $\rho\Theta$ is the derivation in which every sequent occurrence S is replaced by $S\Theta$. We say that Θ is admissible for ρ. Now let $\rho =$

$$\frac{\overset{(\rho_1)}{\Gamma \vdash \Delta, A_1, \ldots, A_\alpha} \quad \overset{(\rho_2)}{B_1, \ldots, B_\beta, \Pi \vdash \Lambda}}{\Gamma\Theta', \Pi\Theta' \vdash \Delta\Theta', \Lambda\Theta'}\ res$$

where Θ' is an s-unifier of $\{A_1, \ldots, A_\alpha, B_1, \ldots, B_\beta\}$. Let us assume that Θ is admissible for ρ_1 and ρ_2. We define that Θ is admissible for ρ if the set

$$U\colon \{A_1\Theta, \ldots, A_\alpha\Theta, B_1\Theta, \ldots, B_\beta\Theta\}$$

is s-unifiable. If Θ^* is an s-unifier of U then we can define $\rho\Theta$ as

$$\frac{\overset{(\rho_1\Theta)}{\Gamma\Theta \vdash \Delta\Theta, A_1\Theta, \ldots, A_\alpha\Theta} \quad \overset{(\rho_2\Theta)}{B_1\Theta, \ldots, B_\beta\Theta, \Pi\Theta \vdash \Lambda\Theta}}{\Gamma\Theta\Theta^*, \Pi\Theta\Theta^* \vdash \Delta\Theta\Theta^*, \Lambda\Theta\Theta^*}\ res$$

Definition 4.3.6. Let ϱ be a RPL_0^Ψ derivation and Θ be an s-substitution which is admissible for ϱ. Θ is called a *global s-unifier* for ϱ if $\varrho\Theta$ is a cut-derivation.

In order to compute global unifiers we need RPL_0^Ψ derivations in some kind of "normal form". Below we define two necessary restrictions on derivations.

Definition 4.3.7. An RPL_0^Ψ derivation ϱ is called *normal* if all s-unifiers of resolution rules in ϱ are normal and restricted.

Remark. Note that, in case of s-unifiability, we can always find normal and restricted s-unifiers; thus the definition above does not really restrict the derivations, it only requires some renamings.

Definition 4.3.8. An RPL_0^Ψ derivation ϱ is called *regular* if for all subderivations ϱ' of ϱ of the form

$$\frac{\begin{array}{cc}(\varrho'_1) & (\varrho'_2)\\ \Gamma \vdash \Delta, A_1, \ldots, A_\alpha & B_1, \ldots, B_\beta, \Pi \vdash \Lambda\end{array}}{\Gamma\Theta, \Pi\Theta \vdash \Delta\Theta, \Lambda\Theta}\ res(\Theta)$$

we have $V^G(\varrho'_1) \cap V^G(\varrho'_2) = \emptyset$

Note that the condition $V^G(\varrho'_1) \cap V^G(\varrho'_2) = \emptyset$ in Definition 4.3.8 guarantees that, for all parameter assignments σ, $\varrho'_1[\sigma]$ and $\varrho'_2[\sigma]$ are variable-disjoint.

We write $\varrho' \leq_{ss} \varrho$ if there exists an s-substitution Θ such that $\varrho'\Theta = \varrho$.

Proposition 4.3.2. *Let ϱ be a normal RPL_0^Ψ derivation. Then there exists a RPL_0^Ψ derivation ϱ' such that $\varrho' \leq_{ss} \varrho$ and ϱ' is normal and regular.*

Proof. By renaming of variables in subproofs and in s-unifiers. □

Proposition 4.3.3. *Let ϱ be a normal and regular RPL_0^Ψ derivation. Then there exists a global s-unifier Θ for ϱ which is normal and $V^G(\Theta) \subseteq V^G(\varrho)$.*

Proof. By induction on the number of inferences in ϱ.

Induction base: ϱ is an axiom. $\emptyset$ is a global s-unifier which trivially fulfils the properties.

For the induction step we distinguish two cases.

- The last rule in ϱ is unary. Then ϱ is of the form

$$\frac{(\varrho')}{\Gamma' \vdash \Delta'}\ \xi$$

  By induction hypothesis there exists a global substitution Θ' which is a global unifier for ϱ' such that Θ' is normal and $V^G(\Theta') \subseteq V^G(\varrho')$. We define $\Theta = \Theta'$. Then, trivially, Θ is normal and a global unifier of ϱ. Moreover, by definition of the unary rules in RPL_0^Ψ, we have $V^G(\varrho') = V^G(\varrho)$ and so $V^G(\Theta) \subseteq V^G(\varrho)$.

- ϱ is of the form

$$\frac{\overset{(\varrho_1)}{\Gamma \vdash \Delta, A_1, \ldots, A_\alpha} \quad \overset{(\varrho_2)}{B_1, \ldots, B_\beta, \Pi \vdash \Lambda}}{\Gamma\Theta, \Pi\Theta \vdash \Delta\Theta, \Lambda\Theta}\ res(\Theta)$$

As ϱ is a normal RPL_0^Ψ derivation the unifier Θ is normal. By regularity of ϱ we have $V^G(\varrho_1) \cap V^G(\varrho_2) = \emptyset$.

By induction hypothesis there exist global normal unifiers Θ_1, Θ_2 for ϱ_1 and ϱ_2 such that $V^G(\Theta_1) \subseteq V^G(\varrho_1)$ and $V^G(\Theta_2) \subseteq V^G(\varrho_2)$. By $V^G(\varrho_1) \cap V^G(\varrho_2) = \emptyset$ we also have $V^G(\Theta_1) \cap V^G(\Theta_2) = \emptyset$.

We show now that (Θ_1, Θ) and (Θ_2, Θ) are composable. As Θ_1 is normal we have for all $\sigma \in \mathscr{S}$

$$V^\iota(\{A_1, \ldots, A_\alpha\}[\sigma]) \cap dom(\Theta_1[\sigma]) = \emptyset.$$

Similarly we obtain

$$V^\iota(\{B_1, \ldots, B_\beta\}[\sigma]) \cap dom(\Theta_2[\sigma]) = \emptyset.$$

As Θ is normal and restricted we have for all $\sigma \in \mathscr{S}$

$$V^\iota(\Theta[\sigma]) \subseteq V^\iota(\{A_1, \ldots, A_\alpha, B_1, \ldots, B_\beta\}[\sigma]).$$

Therefore (Θ_1, Θ) and (Θ_2, Θ) are both composable. As $\Theta_1, \Theta_2, \Theta$ are normal so are $\Theta_1 \star \Theta$ and $\Theta_2 \star \Theta$. As Θ_1, Θ_2 are essentially disjoint we can define

$$\Theta(\varrho) = \Theta_1 \star \Theta \cup \Theta_2 \star \Theta.$$

$\Theta(\varrho)$ is a normal s-substitution and $V^G(\Theta(\varrho)) \subseteq V^G(\varrho)$.

$\Theta(\varrho)$ is also a global unifier of ϱ. Indeed, $\varrho_1\Theta(\varrho) =$

$$\frac{}{\Gamma\Theta \vdash \Delta\Theta, A_1\Theta, \ldots, A_1\Theta}\ (\varrho_1\Theta(\varrho))$$

and $\varrho_2\Theta(\varrho) =$

$$\frac{}{A_1\Theta, \ldots, A_1\Theta, \Pi\Theta \vdash \Lambda\Theta}\ (\varrho_2\Theta(\varrho))$$

So we obtain the derivation

$$\frac{\varrho_1\Theta(\varrho) \quad \varrho_2\Theta(\varrho)}{\Gamma\Theta, \Pi\Theta \vdash \Delta\Theta, \Lambda\Theta}\ cut$$

which is an instance of ϱ and a cut derivation (note that every instance of a cut derivation is a cut derivation as well).

- ϱ is of the form

$$\frac{\begin{matrix}(\varrho_1) & (\varrho_2)\\ \Gamma \vdash \Delta & \Pi \vdash \Lambda\end{matrix}}{\Gamma', \Pi' \vdash \Delta', \Lambda'}\,\chi$$

where χ is a binary introduction rule.

As ϱ is a normal RPL_0^Ψ derivation all occurring s-unifiers in ϱ_1 and ϱ_2 are normal. By regularity of ϱ we have that $V^G(\varrho_1) \cap V^G(\varrho_2) = \emptyset$.

By induction hypothesis there exist global normal unifiers Θ_1, Θ_2 for ϱ_1 and ϱ_2 such that $V^G(\Theta_1) \subseteq V^G(\varrho_1)$ and $V^G(\Theta_2) \subseteq V^G(\varrho_2)$. By $V^G(\varrho_1) \cap V^G(\varrho_2) = \emptyset$ we also have $V^G(\Theta_1) \cap V^G(\Theta_2) = \emptyset$. Moreover, there is no overlap between the domain variables of the unifiers Θ_1 and Θ_2, i.e. $dom(\Theta_1[\sigma]) \cap dom(\Theta_2(\sigma)) = \emptyset$ for all $\sigma \in \mathscr{S}$. Therefore, we can define $\Theta = \Theta_1 \cup \Theta_2$, which is obviously a global s-unifier of ϱ. Furthermore, $V^G(\Theta) = V^G(\Theta_1) \cup V^G(\Theta_2)$, therefore $V^G(\Theta) \subseteq V^G(\varrho_1) \cup V^G(\varrho_2)$ and by definition of binary introduction rules in RPL_0^Ψ, we have $V^G(\Theta) \subseteq V^G(\varrho)$.

□

4.4 Simple Resolution Schemata

Schematic RPL_0^Ψ derivations can be constructed analogously to schematic derivations, see Definition 4.2.6. However, given that we work with derivations from non-axiomatic end-sequents, we need some further restrictions on how derivations in a schema may be connected. The main difference to proof schemata is that we introduce an invariant symbol $\hat{P}_i \in \mathbf{P}$ for which it holds that $\hat{P}_i(\vec{Y}, \vec{n}, m)$ is derivable from $\hat{P}_i(\vec{Y}, \vec{n}, m+1)$. Therefore we allow backwards recursion, which is needed in the construction of many schematic refutations. The rest of the definition of a simple refutation schema is similar to the definition of a proof schema, except that we allow invariant symbols as initial sequents (invariant symbols may occur in the set $\mathscr{C}(\delta)$ below).

Definition 4.4.1 (simple resolution schema)**.** Let Ψ be a schematic formula definition as in Definitions 4.1.22 and 4.1.23 and let $\mathbf{P}$ be the set of all defined predicate symbols in Ψ. We have that for all $\hat{P} \in \mathbf{P}$

$$\hat{P}(\vec{Y}, \vec{n}, 0) = F_B, \quad \hat{P}(\vec{Y}, \vec{n}, s(m)) = F_S\{\xi \leftarrow \hat{P}(\vec{Y}, \vec{n}, m)\}.$$

Let $\hat{P}_1(\vec{Y}, \vec{n}, m) \in \mathbf{P}$ be the $<_{\mathscr{D}}$-maximal symbol and let $\mathbf{\Delta}$ be a finite subset of $\mathbf{\Delta}^*$ such that there exists a $\delta_0 \in \mathbf{\Delta}$ with $\delta_0 > \delta'$ for all $\delta' \in \mathbf{\Delta}$ such that $\delta' \neq \delta_0$ and, for each δ, let $\vec{n}_\delta$ be a (possibly empty) list of global and m_δ an active parameter. There exists at least one symbol $\hat{P}_i \in \mathbf{P}$ which we call the invariant symbol . A finite set of tuples

$$\mathscr{D}\colon \{(\delta, \rho(\delta, \vec{n}_\delta, 0), \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$$

is called a *simple resolution schema* from $\vdash \hat{P}_1(\vec{Y}, \vec{n}, m)$ if the following conditions hold for every $\delta \in \mathbf{\Delta}$:

There exists a (possibly empty) finite set of sequents $\mathscr{C}(\delta)$ and a sequent $S(\delta)$ such that

1. $\rho(\delta, \vec{n}_\delta, 0)$ is a derivation in RPL_0^Ψ of $S(\delta)\{m_\delta \leftarrow 0\}$ from $\vdash \hat{P}_1(\vec{Y}, \vec{n}, 0) \cup \mathscr{C}(\delta)$,

2. $\rho(\delta, \vec{n}_\delta, m_\delta{+}1)$ is a derivation in RPL_0^Ψ of $S(\delta)$ from $\vdash \hat{P}_1(\vec{Y}, \vec{n}, m{+}1) \cup \{(\delta, \Psi) \colon S(\delta)\Psi\} \cup \mathscr{C}(\delta)$, where (δ, Ψ) is a label and $\Psi = \{m_\delta \leftarrow m_\delta + 1\}$,

3. for all $S' \in \mathscr{C}(\delta)$, $S' = (\delta', \Psi) \colon S(\delta')\Psi$ where (δ', Ψ) is a label, $\delta' \in \mathbf{\Delta}$ with $\delta > \delta'$ and Ψ is a parameter replacement on $\vec{n}_{\delta'}$ w.r.t. $\vec{n}_\delta$ such that the conditions (1) and (2) hold for δ',

4. $\hat{P}_i(\vec{Y}, \vec{n}, m)$ is derivable in RPL_0^Ψ from $\hat{P}_i(\vec{Y}, \vec{n}, m+1)$,

5. $\vdash \hat{P}_i(\vec{Y}, \vec{n}, 0)$ is derivable from $\vdash \hat{P}_1(\vec{Y}, \vec{n}, 0)$ by a RPL_0^Ψ-derivation π_0 and $\vdash \hat{P}_i(\vec{Y}, \vec{n}, m+1)$ is derivable from $\vdash \hat{P}_1(\vec{Y}, \vec{n}, m+1)$ by a RPL_0^Ψ-derivation π_{m+1} for the $<_{\hat{\mathscr{P}}}$-maximal symbol $\hat{P}_1$. For all initial sequents occurring in π_0 and π_{m+1} different to $\hat{P}_1$ conditions 4. and 5. hold.

If $S(\delta_0)$ is the empty clause $\vdash$ and $\rho(\delta_0, \vec{n}_{\delta_0}, 0)$ and $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}{+}1)$ are RPL_0^Ψ refutations we call $\mathscr{D}$ a *simple refutation schema* of $\hat{P}_1$.

Remark. Definition 4.4.1 is very similar to the definition of proof schema (Definition 4.2.6). The only difference is that we are working with derivations in RPL_0^Ψ instead of **LKE**-derivations. This means, that we allow non-axiomatic initial sequents of the form $\vdash \hat{P}_1$. To handle recursion over such initial sequents, we allow invariants $\hat{P}_i$ as initial sequents, with the restriction that these invariants are derivable from the main symbol $\hat{P}_1$. Moreover, we allow an up-recursion for invariants, from parameter $m+1$ to m. This is crucial as $\hat{P}_i(m)$ needs to be derivable from the main symbol $\hat{P}_1(m+1)$. We will illustrate the definition in a simple example below.

Example 4.4.1. We construct a simple refutation schema of Ψ:

$$\begin{aligned}
\hat{G}(0) &= \hat{H}(0) \wedge P(c) \wedge \neg P(\hat{f}(0, c)) \\
\hat{G}(n+1) &= \hat{H}(n+1) \wedge P(c) \wedge \neg P(\hat{f}(n+1, c)) \\
\hat{H}(0) &= P(\hat{f}(0, X_1^\delta(0))) \vee \neg P(\hat{f}(0, X_1^\delta(0))) \\
\hat{H}(n+1) &= \hat{H}(n) \wedge (P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))) \\
&\quad \wedge (P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1))))
\end{aligned}$$

We introduce $\hat{J}$, which will be used as an induction invariant (compare to $\hat{P}_i$ in case 4. of Definition 4.4.1), hence we obtain

$$
\begin{array}{rcl}
\hat{G}(0) & = & \hat{J}(0) \wedge \neg P(\hat{f}(0,c)) \\
\hat{G}(n+1) & = & \hat{J}(n+1) \wedge \neg P(\hat{f}(n+1,c)) \\
\hat{J}(0) & = & \hat{H}(0) \wedge P(c) \\
\hat{J}(n+1) & = & \hat{H}(n+1) \wedge P(c) \\
\hat{H}(0) & = & P(\hat{f}(0, X_1^\delta(0))) \vee \neg P(\hat{f}(0, X_1^\delta(0))) \\
\hat{H}(n+1) & = & \hat{H}(n) \wedge (P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))) \\
 & & \wedge (P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1))))
\end{array}
$$

First, we construct a resolution schema $\mathscr{D}_1 \colon \{(\delta_1, \rho(\delta_1, 0), \rho(\delta_1, n+1)\} \cup \mathscr{D}_2$ with end-sequent schema $S(\delta_1) = \vdash P(\hat{f}(n,c))$. $\rho(\delta_1, 0) =$

$$
\dfrac{\dfrac{(\delta_2, \{n \leftarrow 0\}) \colon \vdash \hat{J}(0)}{\vdash \hat{H}(0) \wedge P(c)} B\hat{J}r}{\vdash P(c) (= P(\hat{f}(0,c)))} \wedge_{r_2}
$$

and $\rho(\delta_1, n+1) =$

$$
\dfrac{(\delta_1, \{n \leftarrow n\}) \colon \vdash P(\hat{f}(n,c)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{(\delta_2, \{n \leftarrow n+1\}) \colon \vdash \hat{J}(n+1)}{\vdash \hat{H}(n+1) \wedge P(c)} S\hat{J}r}{\vdash \hat{H}(n+1)} \wedge_{r_1}}{\vdash P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1)))} S\hat{H}r + \wedge_{r_2} \times 2}{\vdash P(\hat{f}(n+1, X_1^\delta(n+1))), \neg P(\hat{f}(n, X_1^\delta(n+1)))} \vee_r}{P(\hat{f}(n, X_1^\delta(n+1))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))} \neg_r}{\vdash P(\hat{f}(n+1,c))} res\{X_1^\delta(n+1) \leftarrow c\}
$$

for $\delta_1 > \delta_2$.

$(\delta_2, \{n \leftarrow 0\}) \colon \vdash \hat{J}(0)$ and $(\delta_2, \{n \leftarrow n+1\}) \colon \vdash \hat{J}(n+1)$ above are the invariants and correspond to the sequents $S' \in \mathscr{C}(\delta_1)$ in case 3. of Definition 4.4.1. For δ_2 we have to show that conditions 1. and 2. of Definition 4.4.1 hold: we construct $\mathscr{D}_2 = \{(\delta_2, \rho(\delta_2, 0), \rho(\delta_2, n+1)\}$, where $\rho(\delta_2, 0) =$

$$
\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{J}(0) \wedge \neg P(\hat{f}(0,c))} B\hat{G}r}{\vdash \hat{J}(0)} \wedge_{r_1}
$$

and $\rho(\delta_2, n+1) =$

$$\dfrac{\dfrac{\vdash \hat{G}(n+1)}{\vdash \hat{J}(n+1) \wedge \neg P(\hat{f}(n+1,c))}\, B\hat{G}r}{\vdash \hat{J}(n+1)}\, \wedge_{r_1}$$

Now we are ready to define $\mathscr{D}\colon \{(\delta_0, \rho(\delta_0,0), \rho(\delta_0,n+1)\} \cup \mathscr{D}_1$, where $S(\delta_0) = \vdash\colon\ \rho(\delta_0,0) =$

$$\dfrac{(\delta_1, \{n \leftarrow 0\})\colon \vdash P(\hat{f}(0,c)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{H}(0) \wedge P(c) \wedge \neg P(\hat{f}(0,c))}\, B\hat{G}r}{\vdash P(c) \wedge \neg P(\hat{f}(0,c))}\, \wedge_{r_2}}{\vdash \neg P(\hat{f}(0,c))}\, \wedge_{r_2}}{P(\hat{f}(0,c)) \vdash}\, \neg_r}{\vdash}\, res$$

$\rho(\delta_0, n+1) =$

$$\dfrac{(\delta_1, \{n \leftarrow n+1\})\colon \vdash P(\hat{f}(n+1,c)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(n+1)}{\vdash \hat{J}(n+1) \wedge \neg P(\hat{f}(n+1,c))}\, S\hat{G}r}{\vdash \neg P(\hat{f}(n+1,c))}\, \wedge_{r_2}}{P(\hat{f}(n+1,c)) \vdash}\, \neg_r}{\vdash}\, res$$

Indeed, $\mathscr{D}$ is a refutation schema: the invariant $\hat{J}$ is derivable from $\hat{G}$ (as shown in $\mathscr{D}_2$) and $\hat{J}(n)$ is derivable from $\hat{J}(n+1)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{J}(n+1)}{\vdash \hat{H}(n+1) \wedge P(c)}\, S\hat{J}r}{\vdash \hat{H}(n+1)}\, \wedge_{r_1}}{\vdash \hat{H}(n)}\, S\hat{H}r + \wedge_{r_1} \qquad \dfrac{\dfrac{\vdash \hat{J}(n+1)}{\vdash \hat{H}(n+1) \wedge P(c)}\, S\hat{J}r}{\vdash P(c)}\, \wedge_{r_4}}{\vdash \hat{H}(n) \wedge P(c)}\, \wedge_r$$

$$\dfrac{\vdash \hat{H}(n) \wedge P(c)}{\vdash \hat{J}(n)}\, S\hat{J}^+r$$

In the example above, $\mathscr{D}$ defines a *simple unification schema*. It is possible to extract the occurring substitutions and combine them to a unification schema. From $\rho(\delta_0, 0)$ we obtain the substitution schema $\Theta(0) = \{\}$ and from $\rho(\delta_0, n+1)$ we obtain the substitution schema $\Theta(n+1) = \{X_1^\delta(n+1) \leftarrow c\} \cup \Theta(n)$. The formal definition of a simple unification schema is given below.

Definition 4.4.2 (simple unification schema)**.** Let $\mathscr{D}$ be a simple resolution schema. We define sets of substitutions (unification schemata) $\Theta(\delta, \vec{n}_\delta, 0)$ and $\Theta(\delta, \vec{n}_\delta, m_\delta + 1)$ for all $\delta \in \mathscr{D}$. We define the sets inductively beginning with the minimal δ.

- Let δ be a minimal element. Then $\mathscr{C}(\delta) = \emptyset$. In this case $\rho(\delta, \vec{n}_\delta, 0)$ is a derivation in RPL_0^Ψ of $S(\delta)\{m_\delta \leftarrow 0\}$.

  Let $\eta(\delta, \vec{n}_\delta, 0)$ be a substitution that substitutes for all variables occurring in $\rho(\delta, \vec{n}_\delta, 0)$ new fresh global variables $X_i^\delta(0)$. Let $\theta(\delta, \vec{n}_\delta, 0)$ be a global s-unifier of $\rho(\delta, \vec{n}_\delta, 0)\eta(\delta, \vec{n}_\delta, 0)$. Then we define

  $$\Theta(\delta, \vec{n}_\delta, 0) = \{\theta(\delta, \vec{n}_\delta, 0)\}.$$

  Now let $\eta(\delta, n, m+1)$ be a substitution that substitutes for all variables occurring in $\rho(\delta, \vec{n}_\delta, m_\delta + 1)$ new fresh global variables $X_i^\delta(m_\delta + 1)$.

  Let $\theta(\delta, \vec{n}_\delta, m_\delta + 1)$ be a global s-unifier of $\rho(\delta, \vec{n}_\delta, m_\delta + 1)\eta(\delta, \vec{n}_\delta, m_\delta + 1)$. Then

  $$\begin{aligned}\Theta(\delta, \vec{n}_\delta, m_\delta + 1) \quad = \quad & \{\theta(\delta, \vec{n}_\delta, m_\delta + 1)\} \cup \\ & \Theta(\delta, \vec{n}_\delta, m_\delta)\eta(\delta, \vec{n}_\delta, m_\delta + 1)\theta(\delta, \vec{n}_\delta, m_\delta + 1).\end{aligned}$$

- Let δ be a non-minimal element. Let

  $$\mathscr{C}(\delta) = \{(\delta_1, \Psi_1) \colon S(\delta_1)\Psi_1, \ldots, (\delta_l, \Psi_l) \colon S(\delta_l)\Psi_l\}$$

  where $\delta > \delta_j$. Let $\eta(\delta, \vec{n}_\delta, 0)$ as above and $\theta(\delta, \vec{n}_\delta, 0)$ be a global s-unifier of $\rho(\delta, \vec{n}_\delta, 0)\eta(\delta, \vec{n}_\delta, 0)$. Then

  $$\begin{aligned}\Theta(\delta, \vec{n}_\delta, 0) \quad = \quad & \{\theta(\delta, \vec{n}_\delta, 0)\} \cup \Theta(\delta_1, \vec{n}_{\delta_1}, 0)\Psi_1\eta(\delta, \vec{n}_\delta, 0)\theta(\delta, \vec{n}_\delta, 0) \cup \ldots \\ & \ldots \cup \Theta(\delta_l, \vec{n}_{\delta_l}, 0)\Psi_l\eta(\delta, \vec{n}_\delta, 0)\theta(\delta, \vec{n}_\delta, 0).\end{aligned}$$

  Similarly we obtain

  $$\begin{aligned}\Theta(\delta, \vec{n}_\delta, m_\delta + 1) \quad = \quad & \{\theta(\delta, \vec{n}_\delta, m_\delta + 1)\} \cup \\ & \Theta(\delta_1, \vec{n}_{\delta_1}, m_{\delta_1} + 1)\Psi_1\eta(\delta, \vec{n}_\delta, m_\delta + 1)\theta(\delta, \vec{n}_\delta, m_\delta + 1) \cup \ldots \\ & \ldots \cup \Theta(\delta_l, \vec{n}_{\delta_l}, m_{\delta_l} + 1)\Psi_l\eta(\delta, \vec{n}_\delta, m_\delta + 1)\theta(\delta, \vec{n}_\delta, m_\delta + 1) \\ & \cup \Theta(\delta, \vec{n}_\delta, m_\delta)\eta(\delta, \vec{n}_\delta, m_\delta + 1)\theta(\delta, \vec{n}_\delta, m_\delta + 1) \\ & \cup \Theta(\delta_i, \vec{n}_{\delta_i}, m_{\delta_i})\Psi_i\eta(\delta, \vec{n}_\delta, m_\delta + 1)\theta(\delta, \vec{n}_\delta, m_\delta + 1),\end{aligned}$$

where $\theta(\delta, \vec{n}_\delta, m_\delta + 1)$ is a global s-unifier of $\rho(\delta, \vec{n}_\delta, m_\delta + 1)\eta(\delta, \vec{n}_\delta, m_\delta + 1)$ and in case we have an occurrence of $(\delta_i, \Psi)\colon S(\delta_i) \in \mathscr{C}(\delta)$ (reference to the invariant $\hat{P}_i(\vec{Y}, \vec{m}, n)$) where Ψ is different to $\{n \leftarrow n+1\}$ let $\Theta(\delta_i, \vec{n}_{\delta_i}, m_{\delta_i})$ be defined as the unification schema obtained from the derivation of $\hat{P}_i(\vec{Y}, \vec{m}, n)$ from $\hat{P}_i(\vec{Y}, \vec{m}, n+1)$ and otherwise let it be the empty set.

Let δ_0 be the maximal proof symbol in $\mathscr{D}$. Then the tuple $(\Theta(\delta_0, \vec{n}_{\delta_0}, 0), \Theta(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}+1))$ is the unification schema of $\mathscr{D}$.

Example 4.4.2. Consider the simple refutation schema $\mathscr{D}$ from Example 4.4.1. The unification schema of $\mathscr{D}$ is given by the tuple $(\Theta(\delta_0, 0), \Theta(\delta_0, n+1))$, where

$$\begin{aligned} \Theta(\delta_0, 0) &= \{\} \cup \Theta(\delta_1, 0) \\ \Theta(\delta_0, n+1) &= \{\} \cup \Theta(\delta_1, n+1), \end{aligned}$$

where $(\Theta(\delta_1, 0), \Theta(\delta_1, n+1))$ is given by

$$\begin{aligned} \Theta(\delta_1, 0) &= \{\} \\ \Theta(\delta_1, n+1) &= \{X_1^\delta(n+1) \leftarrow c\} \cup \Theta(\delta_1, n). \end{aligned}$$

Simple resolution schemata can be evaluated under parameter assignments in analogy to the evaluation of proof schemata. If $\mathscr{D}$ is a resolution schema and σ a parameter assignment we will define $\mathscr{D}\sigma\downarrow$ as the pair $(\rho(\delta_0, \vec{n}, 0)\sigma\downarrow, \rho(\delta_0, \vec{n}, m+1)\sigma\downarrow)$.

Definition 4.4.3 (evaluation of simple resolution schema)**.** Let $\mathscr{D}$ be a simple resolution schema and σ a parameter assignment. In defining $\mathscr{D}\sigma\downarrow$ we proceed by double induction on the ordering of proof symbols and the assignments $\sigma(m)$.

- Let δ be a minimal element in $\mathbf{\Delta}$.
  1. $\sigma(m_\delta) = 0$.
     For $\rho(\delta, \vec{n}_\delta, 0)$ we evaluate all sequents in $\rho(\delta, \vec{n}_\delta, 0)$ under σ. The result is a derivation RPL_0^Ψ.
  2. $\sigma(m_\delta) = \alpha > 0$. We are now defining $\rho(\delta, \vec{n}_\delta, m_\delta + 1)\sigma'\downarrow$ where $\sigma'(p) = \sigma(p)$ for all $p \neq m_\delta$ and $\sigma'(m_\delta) = \alpha - 1$.
     Evaluate all sequents in $\rho(\delta, \vec{n}_\delta, m_\delta)$ under σ' except the leaves $(\delta, \Psi)\colon S(\delta)\Psi$ where Ψ is the corresponding parameter replacement. Then replace $(\delta, \Psi)\colon S(\delta)\Psi$ by $\rho(\delta, \vec{n}_\delta, m_\delta)\Psi\sigma'[m_\delta/\alpha]\downarrow$ where $\sigma'[m_\delta/\alpha]$ is defined as $\sigma'[m_\delta/\alpha](p) = \sigma'(p)$ for all $p \neq m_\delta$ and $\sigma'[m_\delta/\alpha](m_\delta) = \alpha - 1$. The result is a derivation in RPL_0^Ψ.
- $\delta \in \mathbf{\Delta}$ is not minimal.

1. $\sigma(m_\delta) = 0$.

   For $\rho(\delta, \vec{n}_\delta, 0)$ we evaluate all sequents in $\rho(\delta, \vec{n}_\delta, 0)$ under σ except the leaves $(\delta', \Psi')\colon S(\delta')\Psi'$ for $\delta > \delta'$ and the corresponding parameter replacement Ψ'. Then replace $(\delta', \Psi')\colon S(\delta')\Psi'$ by $\rho(\delta', \vec{n}_{\delta'}, m_{\delta'})\Psi'\sigma\downarrow$. The result is a derivation RPL_0^Ψ.

2. $\sigma(m_\delta) = \alpha > 0$. We are now defining $\rho(\delta, \vec{n}_\delta, m_\delta + 1)\sigma'\downarrow$ where $\sigma'(p) = \sigma(p)$ for all $p \neq m_\delta$ and $\sigma'(m_\delta) = \alpha - 1$.

   Evaluate all sequents in $\rho(\delta, \vec{n}_\delta, m_\delta)$ under σ except the leaves $(\delta_i, \Psi_i)\colon \vdash \hat{P}_i(\vec{Y}, \vec{n}, m)$, where $\hat{P}_i$ is the invariant symbol and Ψ_i is a parameter replacement different to $\{n \leftarrow n+1\}$, $(\delta, \Psi)\colon S(\delta)\Psi$ and $(\delta', \Psi')\colon S(\delta')\Psi'$ for $\delta > \delta'$ and the corresponding parameter replacements Ψ and Ψ'. Then replace $(\delta, \Psi)\colon S(\delta)\Psi$ by $\rho(\delta, \vec{n}_\delta, m_\delta)\Psi\sigma'[m_\delta/\alpha]\downarrow$ where $\sigma'[m/\alpha]$ is defined as $\sigma'[m_\delta/\alpha](p) = \sigma'(p)$ for all $p \neq m_\delta$ and $\sigma'[m_\delta/\alpha](m_\delta) = \alpha - 1$. Replace $(\delta', \Psi')\colon S(\delta')\Psi'$ by $\rho(\delta', \vec{n}_{\delta'}, m_{\delta'})\Psi'\sigma'\downarrow$. Finally, replace $(\delta_i, \Psi_i)\colon \vdash \hat{P}_i(\vec{Y}, \vec{n}, m)$ by the derivation $\pi\sigma'\downarrow$, where π is the derivation of $\vdash \hat{P}_i(\vec{Y}, \vec{n}, m)\Psi$ from the maximal symbol $\hat{P}_i(\vec{Y}, \vec{n}, m+1)$ (note that $\vdash \hat{P}_i(\vec{Y}, \vec{n}, m)$ is derivable from $\vdash \hat{P}_i(\vec{Y}, \vec{n}, m+1)$ and $\vdash \hat{P}_i(\vec{Y}, \vec{n}, m+1)$ is derivable from the maximal symbol $\hat{P}_1$). The result is a derivation in RPL_0^Ψ.

Example 4.4.3. Consider the simple refutation schema $\mathscr{D}$ from example 4.4.1 with proof symbols $\delta_0 > \delta_1 > \delta_2$. Let $\sigma(n) = 0$. Then we construct $\mathscr{D}\sigma\downarrow$ by starting with the minimal proof symbol δ_2: as $\sigma(n) = 0$ we construct $\rho(\delta_2, 0)\sigma\downarrow$:

$$\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{J}(0) \wedge \neg P(\hat{f}(0,c))}\, B\hat{G}r}{\vdash \hat{J}(0)}\, \wedge_{r_1}$$

We proceed with the proof symbol δ_1. $\rho(\delta_1, 0)\sigma\downarrow =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{J}(0) \wedge \neg P(\hat{f}(0,c))}\, B\hat{G}r}{\vdash \hat{J}(0)}\, \wedge_{r_1}}{\vdash \hat{H}(0) \wedge P(c)}\, B\hat{J}r}{\vdash P(c)(= P(\hat{f}(0,c)))}\, \wedge_{r_2}$$

Note that in constructing $\rho(\delta_1, 0)\sigma\downarrow$ we replaced the initial sequent $(\delta_2, \emptyset)\colon \vdash \hat{J}(0)$ with the derivation $\rho(\delta_2, 0)\sigma\downarrow$.

Finally, we construct $\mathscr{D}\sigma\downarrow$ which is $\rho(\delta_0, 0)\sigma\downarrow =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{J}(0) \wedge \neg P(\hat{f}(0,c))}\, B\hat{G}r}{\vdash \hat{J}(0)}\, \wedge_{r_1}}{\vdash \hat{H}(0) \wedge P(c)}\, B\hat{J}r}{\vdash P(c) (= P(\hat{f}(0,c)))}\, \wedge_{r_2} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{H}(0) \wedge P(c) \wedge \neg P(\hat{f}(0,c))}\, B\hat{G}r}{\vdash P(c) \wedge \neg P(\hat{f}(0,c))}\, \wedge_{r_2}}{\vdash \neg P(\hat{f}(0,c))}\, \wedge_{r_2}}{P(\hat{f}(0,c)) \vdash}\, \neg_r}{\vdash}\, res$$

Here again the axiom $(\delta_1, \{n \leftarrow 0\})\colon \vdash P(\hat{f}(0,c))$ was replaced by the derivation $\rho(\delta_1, 0)\sigma\downarrow$.

Now consider $\sigma'(n) = s(s(0))$. We construct $\mathscr{D}\sigma'\downarrow$ by first constructing $\rho(\delta_2, n+1)\sigma''\downarrow$, where σ'' is defined as $\sigma''(p) = \sigma'(p)$ for all $p \neq n$ and $\sigma''(n) = s(0)$:

$$\dfrac{\dfrac{\dfrac{\vdash \hat{G}(s(s(0)))}{\vdash \hat{J}(s(s(0))) \wedge \neg P(\hat{f}(s(s(0)),c))}\, B\hat{G}r}{\vdash \hat{J}(s(s(0)))}\, \wedge_{r_1}$$

Using $\rho(\delta_2, n+1)\sigma''\downarrow$ as replacement for the initial sequent $(\delta_2, n \leftarrow n+1)\colon \vdash \hat{J}(n+1)$ we construct $\rho(\delta_1, n+1)\sigma''\downarrow=$

$$\dfrac{\begin{array}{c}(\rho(\delta_1,n)\sigma''\downarrow)\\ \vdash P(\hat{f}(s(0),c))\end{array} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(s(s(0)))}{\vdash \hat{J}(s(s(0))) \wedge \neg P(\hat{f}(s(s(0)),c))}\, B\hat{G}r}{\vdash \hat{J}(s(s(0)))}\, \wedge_{r_1}}{\vdash \hat{H}(s(s(0))) \wedge P(c)}\, S\hat{J}r}{\vdash \hat{H}(s(s(0)))}\, \wedge_{r_1}}{\vdash P(\hat{f}(s(s(0)), X_1^\delta(s(s(0))))) \vee \neg P(\hat{f}(s(0), X_1^\delta(s(s(0)))))}\, S\hat{H}r + \wedge_{r_2} \times 2}{\vdash P(\hat{f}(s(s(0)), X_1^\delta(s(s(0))))), \neg P(\hat{f}(s(0), X_1^\delta(s(s(0))))))}\, \vee_r}{P(\hat{f}(s(0), X_1^\delta(s(s(0))))) \vdash P(\hat{f}(s(s(0)), X_1^\delta(s(s(0)))))}\, \neg_r}{\vdash P(\hat{f}(s(s(0)),c))}\, res\{X_1^\delta(s(s(0))) \leftarrow c\}$$

Note that the initial sequent $(\delta_1, \{n \leftarrow n\})\colon \vdash P(\hat{f}(n,c))$ was replaced by the derivation $\rho(\delta_1, n)\sigma''\downarrow=$

$$\dfrac{\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{J}(0) \wedge \neg P(\hat{f}(0,c))}\, B\hat{G}r}{\vdash \hat{J}(0)}\, \wedge_{r_1}}{\vdash \hat{H}(0) \wedge P(c)}\, B\hat{J}r \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(s(0))}{\vdash \hat{J}(s(0)) \wedge \neg P(\hat{f}(s(0),c))}\, B\hat{G}r}{\vdash \hat{J}(s(0))}\, \wedge_{r_1}}{\vdash \hat{H}(s(0)) \wedge P(c)}\, S\hat{J}r}{\vdash \hat{H}(s(0))}\, \wedge_{r_1}}{\vdash P(\hat{f}(s(0), X_1^\delta(s(0)))) \vee \neg P(\hat{f}(0, X_1^\delta(s(0))))}\, S\hat{H}r + \wedge_{r_2} \times 2$$

$$\dfrac{\vdash P(\hat{f}(s(0), X_1^\delta(s(0)))), \neg P(\hat{f}(0, X_1^\delta(s(0)))))}{P(\hat{f}(0, X_1^\delta(s(0)))))) \vdash P(\hat{f}(s(0), X_1^\delta(s(0))))}\, \neg_r$$

$$\vdash P(\hat{f}(s(0), c))\, res\{X_1^\delta(s(0)) \leftarrow c\}$$

Finally, we construct $\rho(\delta_0, n+1)\sigma''\downarrow =$

$$\dfrac{(\rho(\delta_1, n+1)\sigma''\downarrow) \qquad \dfrac{\dfrac{\dfrac{\vdash \hat{G}(s(s(0)))}{\vdash \hat{J}(s(s(0))) \wedge \neg P(\hat{f}(s(s(0)),c))}\, S\hat{G}r}{\vdash \neg P(\hat{f}(s(s(0)),c))}\, \wedge_{r_2}}{P(\hat{f}(s(s(0)),c)) \vdash}\, \neg_r}{\vdash P(\hat{f}(s(s(0)),c))}\, res$$

Note that the initial sequent $(\delta_1, \{n \leftarrow n+1\})\colon \vdash P(\hat{f}(n+1,c))$ was replaced with $\rho(\delta_1, n+1)\sigma''\downarrow$.

Simple unification schemata $\Theta(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})$ for some simple refutation schema $\mathscr{D}$ can be also evaluated in the obvious way and we denote the evaluation under σ as $\Theta(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow$. When evaluating the unification schema from Example 4.4.2 for some numeral α we obtain the sequence

$$\{X_1^\delta(\alpha) \leftarrow c, X_1^\delta(\alpha-1) \leftarrow c, \ldots, X_1^\delta(1) \leftarrow c\}.$$

Note that $\Theta(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow$ corresponds to the set of unifiers obtainable from the refutation $\mathscr{D}\sigma\downarrow$.

CHAPTER 5

Herbrand's Theorem in Inductive Structures

In this chapter we will extend the CERES method defined in Section 3.1 to the schematic case. Like the method for first-order logic, schematic CERES is based on the extraction of a schematic characteristic formula and a schematic refutation thereof. The first approach to a schematic CERES method can be found in [DLRW13], where also the concept of proof schema was developed. The schematic CERES method was based on the extraction of a schematic clause set and culminated in a complicated formalism and method, which does not guarantee a fully automated transformation and analysis of a given proof. In [LPW17] the investigation on proof schemata from [DLRW13] is continued and the CERES method for proof schemata is simplified. It is shown that proof schemata can be used to extract a generalization of mid-sequents, a so-called Herbrand system. Indeed, from a resolution proof schema and a substitution schema (both defined in [DLRW13, LPW17]) such a Herbrand system can be extracted. In contrast to the method in [DLRW13] the one in [LPW17], based on negation normal forms and a complicated translation to the n-clause calculus, can be fully automated, however at the cost of expressivity. The most important difference of the methods lies in the specification of refutation schemata. In fact, the development of a schematic resolution calculus has always been one of the stumbling blocks of the schematic CERES method. Using the novel calculus RPL_0^{Ψ} we managed to resolve the problems and constraints of the existing methods. The novel resolution calculus allows us to define a simplified schematic CERES method, which extends the existing methods by allowing several parameters and can be used for (semi-) automated proof analysis. In this chapter we will lift the proof analysis method based on CERES for first-order logic introduced in Chapter 3 to the schematic case.

5.1 The Schematic CERES Method

In principle, the schematic CERES method for proof analysis works exactly the same as the one for first-order logic, except that we work in a schematic setting. This means, that in a first step we have to extract a schematic characteristic formula. The usual first-order definition of the characteristic formula is based on the cut-status of the formula occurrences in a proof, i.e. whether a given formula occurrence is a cut-ancestor or not. However, a formula occurrence in a proof schema corresponds to many formula occurrences in its evaluation, some of which will be cut-ancestors and some will be not. Hence, we need some machinery to track the cut-status of formula occurrences in proof schemata, which leads to a first difference between the first-order and the schematic CERES method: we add a new parameter to the characteristic formula of a proof symbol which will denote the set of formula occurrences in the end-sequent of that proof symbol that are cut-ancestors.

Definition 5.1.1 (configuration, [LPW17])**.** A set Ω of formula occurrences in the end-sequent of a proof π is called a *configuration* for π. If ν is a node in π we define by $S(\nu, \Omega)$ the subsequent of $S(\nu)$ consisting of all formulas which are ancestors of Ω .

Now we can give the full definition of the schematic characteristic formula and the schematic projection.

Definition 5.1.2 (schematic characteristic formula and schematic projection, [LPW17])**.** Let $\mathscr{D}\colon \{(\delta,\ \rho(\delta, \vec{n}_\delta, 0),\ \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ be a regular proof schema. We proceed by induction on the ordering of proof symbols.

(A) Let δ be a minimal element in $\mathbf{\Delta}$, so $\rho(\delta, \vec{n}_\delta, 0)$ is an ordinary **LKE**-proof and $\rho(\delta, \vec{n}_\delta, m_\delta + 1)$ calls only itself.

Let $\pi = \rho(\delta, \vec{n}_\delta, 0)$ and Ω be a set of configurations for π and Ω' be the set of occurrences of cut formulas in π. For all nodes ν in π we introduce defined atoms $\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, m_\delta)$ and proof terms $\mathrm{Proj}(\pi, \nu, \Omega)$; for the root node ν_0 of π we have $C(\pi, \Omega) = \hat{P}[\delta, \nu_0, \Omega](\bar{X}, \vec{n}_\delta, m_\delta)$, where $\bar{X}$ are the global variables occurring in π and $P(\pi, \Omega) = \rho(\delta_p, \vec{n}_{\delta_p}, m_{\delta_p}) = \mathrm{Proj}(\pi, \nu_0, \Omega)$, where δ_p is the proof symbol in the projection schema $\{(\delta_p, \rho(\delta_p, \vec{n}_{\delta_p}, 0), \rho(\delta_p, \vec{n}_{\delta_p}, m_{\delta_p} + 1))\}$ that corresponds to the proof symbol δ in the proof schema. Note that the ordering on proof symbols in the projection schema by construction follows the ordering of proof symbols in the proof schema $\mathscr{D}$. The ordering on defined symbols $\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, m_\delta)$ is given by $\hat{P}[\delta', \nu', \Omega](\bar{X}, \vec{n}_\delta, m_\delta) <_{\hat{\mathscr{D}}} \hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, m_\delta)$, where $\delta' < \delta$ and in case $\delta = \delta'$ the node ν' is the premise of ν.

- Let ν be a leaf node in π. Then $S(\nu) = A \vdash A$ for some schematic atom A. We distinguish the cases:

  1. $S(\nu, \Omega \cup \Omega') = \vdash$.
     Then $\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, 0) = \bot$ and $\mathrm{Proj}(\pi, \nu, \Omega) = \nu\colon A \vdash A$.

2. $S(\nu, \Omega \cup \Omega') = \vdash A$.
   Then $\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, 0) = A$ and $\mathrm{Proj}(\pi, \nu, \Omega) = \nu \colon A \vdash A$.
3. $S(\nu, \Omega \cup \Omega') = A \vdash$.
   Then $\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, 0) = \neg A$ and $\mathrm{Proj}(\pi, \nu, \Omega) =$.

$$\frac{A \vdash A}{\vdash A, \neg A}\ \neg \colon r$$

4. $S(\nu, \Omega \cup \Omega') = A \vdash A$.
   Then $\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, 0) = A \vee \neg A$ and $\mathrm{Proj}(\pi, \nu, \Omega) =$.

$$\frac{\dfrac{A \vdash A}{\vdash A, \neg A}\ \neg \colon r}{\vdash A \vee \neg A}\ \vee \colon r$$

- Now let us assume that ξ is a unary inference in π with premise ν' and conclusion ν in π. Inductively we assume that $\hat{P}[\delta, \nu', \Omega](\bar{X}, \vec{n}_\delta, 0)$ and $\mathrm{Proj}(\pi, \nu', \Omega)$ have been defined and $\mathrm{Proj}(\pi, \nu', \Omega)$ is a proof of $\Gamma \vdash \Delta, \hat{P}[\delta, \nu', \Omega](\bar{X}, \vec{n}_\delta, 0)$, where $\Gamma \vdash \Delta = S(\nu', \bar{\Omega})$ and $\bar{\Omega}$ are the formula occurrences in the end-sequent of π which are not in Ω.

  We define

$$\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, 0) = \hat{P}[\delta, \nu', \Omega](\bar{X}, \vec{n}_\delta, 0).$$

  For defining $\mathrm{Proj}(\pi, \nu, \Omega)$ we distinguish two cases:

  1. the principal formula of ξ is an ancestor of $\Omega \cup \Omega'$ then $\mathrm{Proj}(\pi, \nu, \Omega) = \mathrm{Proj}(\pi, \nu', \Omega)$.
  2. The principal formula of ξ is an ancestor of the complement of $\Omega \cup \Omega'$, denoted by $\bar{\Omega}$. Then $\mathrm{Proj}(\pi, \nu, \Omega) =$

$$\frac{\begin{array}{c}(\mathrm{Proj}(\pi, \nu', \Omega)\\ \Gamma \vdash \Delta, \hat{P}[\delta, \nu', \Omega](\bar{X}, \vec{n}_\delta, 0)\end{array}}{\Gamma' \vdash \Delta', \hat{P}[\delta, \nu', \Omega](\bar{X}, \vec{n}_\delta, 0)}\ \xi$$

  As $\hat{P}[\delta, \nu, \Omega](\bar{X}, \vec{n}_\delta, 0) = \hat{P}[\delta, \nu', \Omega](\bar{X}, \vec{n}_\delta, 0)$ and $\Gamma' \vdash \Delta' = S(\nu, \bar{\Omega})$ the proof $\mathrm{Proj}(\pi, \nu, \Omega)$ is of the required form.

- Assume that ξ is a binary inference with premises ν_1, ν_2 and conclusion ν. Assume further that $\hat{P}[\delta, \nu_1, \Omega](\bar{X}, \vec{n}_\delta, 0)$, $\hat{P}[\delta, \nu_2, \Omega](\bar{X}, \vec{n}_\delta, 0)$ are defined and $\mathrm{Proj}(\pi, \nu_i, \Omega)$ are derivations of $\Gamma_i \vdash \Delta_i, \hat{P}[\delta, \nu_i, \Omega](\bar{X}, \vec{n}_\delta, 0)$ (for $i = 1, 2$) such that $\Gamma_i \vdash \Delta_i = S(\nu_i, \bar{\Omega})$. We distinguish two cases:

  1. the auxiliary formulas in ξ are ancestors of $\Omega \cup \Omega'$. Then we define $\mathrm{Proj}(\pi, \nu, \Omega) =$

$$\frac{\begin{array}{cc}(\mathrm{Proj}(\pi, \nu_1, \Omega)) & (\mathrm{Proj}(\pi, \nu_2, \Omega))\\ \Gamma_1 \vdash \Delta_1, \hat{P}[\delta, \nu_1, \Omega](\bar{X}, \vec{n}_\delta, 0) & \Gamma_2 \vdash \Delta_2, \hat{P}[\delta, \nu_2, \Omega](\bar{X}, \vec{n}_\delta, 0)\end{array}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, \hat{P}[\delta, \nu_1, \Omega](\bar{X}, \vec{n}_\delta, 0) \wedge \hat{P}[\delta, \nu_2, \Omega](\bar{X}, \vec{n}_\delta, 0)}\ \wedge \colon r$$

and

$$\hat{P}[\delta,\nu,\Omega](\bar{X},\vec{n}_\delta,0) = \hat{P}[\delta,\nu_1,\Omega](\bar{X},\vec{n}_\delta,0) \wedge \hat{P}[\delta,\nu_2,\Omega](\bar{X},\vec{n}_\delta,0),$$

where $\hat{P}[\delta,\nu_i,\Omega](\bar{X},\vec{n}_\delta,0) <_{\hat{\mathscr{P}}} \hat{P}[\delta,\nu,\Omega](\bar{X},\vec{n}_\delta,0)$ for $i \in \{1,2\}$. Note that $S(\nu,\bar{\Omega}) = \Gamma_1,\Gamma_2 \vdash \Delta_1,\Delta_2$.

2. the auxiliary formulas in ξ are ancestors of the complement of $\Omega \cup \Omega'$, denoted by $\bar{\Omega}$. Then we define $\mathrm{Proj}(\pi,\nu,\Omega) =$

$$\dfrac{\dfrac{\begin{array}{cc}(\mathrm{Proj}(\pi,\nu_1,\Omega)) & (\mathrm{Proj}(\pi,\nu_2,\Omega)) \\ \Gamma_1 \vdash \Delta_1, \hat{P}[\delta,\nu_1,\Omega](\bar{X},\vec{n}_\delta,0) & \Gamma_2 \vdash \Delta_2, \hat{P}[\delta,\nu_2,\Omega](\bar{X},\vec{n}_\delta,0)\end{array}}{\Gamma_1',\Gamma_2' \vdash \Delta_1',\Delta_2', \hat{P}[\delta,\nu_1,\Omega](\bar{X},\vec{n}_\delta,0), \hat{P}[\delta,\nu_2,\Omega](\bar{X},\vec{n}_\delta,0)}\,\xi}{\Gamma_1',\Gamma_2' \vdash \Delta_1',\Delta_2', \hat{P}[\delta,\nu_1,\Omega](\bar{X},\vec{n}_\delta,0) \vee \hat{P}[\delta,\nu_2,\Omega](\bar{X},\vec{n}_\delta,0)}\,\vee\colon r$$

and

$$\hat{P}[\delta,\nu,\Omega](\bar{X},\vec{n}_\delta,0) = \hat{P}[\delta,\nu_1,\Omega](\bar{X},\vec{n}_\delta,0) \vee \hat{P}[\delta,\nu_2,\Omega](\bar{X},\vec{n}_\delta,0),$$

where $\hat{P}[\delta,\nu_i,\Omega](\bar{X},\vec{n}_\delta,0) <_{\hat{\mathscr{P}}} \hat{P}[\delta,\nu,\Omega](\bar{X},\vec{n}_\delta,0)$ for $i \in \{1,2\}$. Note that, here, $S(\nu,\bar{\Omega}) = \Gamma_1',\Gamma_2' \vdash \Delta_1',\Delta_2'$.

- If ν_0 is the root node of π we have $C(\pi,\Omega) = \hat{P}[\delta,\nu_0,\Omega](\bar{X},\vec{n}_\delta,0)$, where $\bar{X}$ are the global variables occurring in π and $P(\pi,\Omega) = \rho(\delta_p,\vec{n}_{\delta_p},0) = \mathrm{Proj}(\pi,\nu_0,\Omega)$ is a cut-free proof of $\Gamma \vdash \Delta, \hat{P}[\delta,\nu_0,\Omega](\bar{X},\vec{n}_\delta,0)$ where $\Gamma \vdash \Delta = S(\nu_0,\bar{\Omega})$.

  When we consider the proof $\pi = \rho(\delta,\vec{n}_\delta,m_\delta+1)$ all definitions are the same as for the derivation $\rho(\delta,\vec{n}_\delta,0)$ except for initial sequents of the form $(\delta,\emptyset)\colon S(\delta)$. Let ν be the occurrence of such an initial sequent and Ω^* be the occurrences in $S(\delta)$ which are ancestors of $\Omega \cup \Omega'$ in π. Then we define

$$\begin{array}{rcl}\mathrm{Proj}(\pi,\nu,\Omega) & = & (\delta_p,\emptyset)\colon S(\delta_p), \\ \hat{P}[\delta,\nu,\Omega^*](\bar{X},\vec{n}_\delta,m_\delta+1) & = & \hat{P}[\delta,\nu,\Omega^*](\bar{X},\vec{n}_\delta,m_\delta) = C(\rho(\delta,\vec{n}_\delta,m_\delta),\Omega^*),\end{array}$$

  where the label $(\delta_p,\emptyset)$ when evaluated links to the derivation $\rho(\delta_p,\vec{n}_{\delta_p},m_{\delta_p}) = P(\rho(\delta,\vec{n}_\delta,m_\delta),\Omega^*)$.

  Here again, if ν_0 is the root node of π we have $C(\pi,\Omega) = \hat{P}[\delta,\nu_0,\Omega](\bar{X},\vec{n}_\delta,m_\delta+1)$, where $\bar{X}$ are the global variables occurring in π and $P(\pi,\Omega) = \rho(\delta_p,\vec{n}_{\delta_p},m_{\delta_p}+1) = \mathrm{Proj}(\pi,\nu_0,\Omega)$ is a cut-free proof of $\Gamma \vdash \Delta, \hat{P}[\delta,\nu_0,\Omega](\bar{X},\vec{n}_\delta,m_\delta+1)$, where $\Gamma \vdash \Delta = S(\nu_0,\bar{\Omega})$. The projection schema is defined as $\{(\delta_p,\ \rho(\delta_p,\vec{n}_{\delta_p},0),\rho(\delta_p,\ \vec{n}_{\delta_p},m_{\delta_p}+1))\}$.

(B) δ is not a minimal element in $\mathbf{\Delta}$.
Let $\pi = \rho(\delta,\vec{n}_\delta,0)$. The definition of $\mathrm{Proj}(\pi,\nu,\Omega\cup\Omega')$ is the same as for minimal δ with the exception of leaves $\delta'\colon S(\delta')\Psi$ for a parameter replacement Ψ of $\vec{n}_{\delta'},m_{\delta'}$ with respect

to $\vec{n}_\delta$ and $\delta' < \delta$. Let ν be such a leaf and let Ω^* be all occurrences in $S(\delta')$ which are ancestors of $\Omega \cup \Omega'$ in π. Then we define

$$\begin{aligned} \mathrm{Proj}(\pi, \nu, \Omega^*) &= (\delta'_p, \Psi)\colon S(\delta'_p)\Psi, \\ \hat{P}[\delta, \nu, \Omega^*](\bar{X}, \vec{n}_\delta, 0) &= \hat{P}[\delta', \nu, \Omega^*](\bar{X}, \vec{n}_{\delta'}, m_{\delta'})\Psi = C(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'}), \Omega^*)\Psi. \end{aligned}$$

such that the label (δ'_p, Ψ) when evaluated links to the derivation $\rho(\delta'_p, \vec{n}_{\delta'_p}, m_{\delta'_p})\Psi = P(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'}), \Omega^*)\Psi$, $\delta'_p < \delta_p$ and, as $\delta' < \delta$, $\hat{P}[\delta', \nu, \Omega^*](\bar{X}, \vec{n}_{\delta'}, m_{\delta'}) <_{\hat{\mathscr{P}}} \hat{P}[\delta, \nu, \Omega^*](\bar{X}, \vec{n}_\delta, 0)$. Note that the projection schema is defined as $\{(\delta'_p, \rho(\delta'_p, \vec{n}_{\delta'_p}, 0), \rho(\delta'_p, \vec{n}_{\delta'_p}, m_{\delta'_p} + 1))\}$.

For $\pi = \rho(\delta, \vec{n}_\delta, m_\delta + 1)$ and ν a node in π the definitions of $\mathrm{Proj}(\pi, \nu, \Omega \cup \Omega')$ and of $\hat{P}[\delta, \nu, \Omega^*](\bar{X}, \vec{n}_\delta, m_\delta + 1)$ are covered by the cases of $\rho(\delta', \vec{n}_{\delta'}, m_{\delta'})$ for minimal δ' and by the case $\rho(\delta, \vec{n}_\delta, 0)$ above. Indeed, the leaves which are not axioms are either of the form $\delta\colon S(\delta)$ or $\delta'\colon S(\delta')\Psi$ for a $\delta > \delta'$ and a parameter replacement Ψ.

Let δ_0 be the maximal element in $\mathbf{\Delta}$. Then the term

$$C^*(\delta_0) = C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \emptyset) = \hat{P}[\delta_0, \nu_0, \emptyset](\vec{X}, \vec{n}_{\delta_0}, m_{\delta_0}),$$

where ν_0 is the root node of $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})$, is called the *schematic characteristic term* of $\mathscr{D}$. The term

$$P^*(\delta_0) = P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \emptyset)$$

is called the *schematic projection* of $\mathscr{D}$. The end-sequent of $P^*(\delta_0)$ is defined as the end-sequent of $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \emptyset)$.

Note that in the construction of the schematic characteristic formula we introduce new defined symbols with a corresponding ordering. The same holds for the construction of a projection schema, where the ordering of the proof symbols follows directly the ordering of the proof symbols in the original proof schema.

Proposition 5.1.1. *$C^*(\delta_0)$ and $P^*(\delta_0)$ are well-defined, i.e. $\hat{P}[\delta_0, \nu_0, \Omega] \downarrow \sigma$ and $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}))\downarrow \sigma$ are defined.*

Proof. The constructed formula schemata are of the same structure as in Definition 4.1.22, as we build formulas only using $\neg$, $\wedge$ and $\vee$. Moreover, the recursions for the defined predicate symbols are of the same type as in Definition 4.1.23. Indeed, whenever we have a $\hat{P}$ which is not $<_{\hat{\mathscr{P}}}$-minimal and defined over some formula schema F_B, F_S, we have that for every $\hat{Q}$ occurring in F_B, F_S it holds that $\hat{Q} <_{\hat{\mathscr{P}}} \hat{P}$. In particular by ordering the new defined predicate symbols $\hat{P}[\delta, \nu, \Omega]$ first by the ordering $<$ on proof symbols δ and then by the nodes ν ensures to obtain an order $<_{\hat{\mathscr{P}}}$ that satisfies the conditions of Definition 4.1.23. By Proposition 4.1.10 the occurring formulas are defined. □

The construction of schematic characteristic formula and proof projection will be illustrated in the following examples. The first example is the same as in [LPW17], except that we have adapted it to our new formalism. The second example is based on two parameters n and m.

Example 5.1.1. Let $\mathscr{D} = \{(\delta', \rho(\delta', 0), \rho(\delta', n+1))\} \cup \mathscr{D}_1$, where

$$S(\delta')\colon \forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c)))$$

be the regular proof schema defined in Example 4.2.3, where $\mathscr{D}_1 = \{(\delta, \rho(\delta,0), \rho(\delta, n+1))\}$,

$$S(\delta)\colon \forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(\hat{f}(n,x)))$$

for $\delta < \delta'$ and $\hat{f}(0,x) = x, \hat{f}(s(n),x) = f(\hat{f}(n,x))$. Let $\mathscr{E}$, the set of all defining equations, contain

$$\begin{array}{rcl}
\hat{G}(0) & = & \hat{H}(0) \wedge P(c) \wedge \neg P(\hat{f}(0,c)) \\
\hat{G}(n+1) & = & \hat{H}(n+1) \wedge P(c) \wedge \neg P(\hat{f}(n+1,c)) \\
\hat{H}(0) & = & P(\hat{f}(0, X_1^\delta(0))) \vee \neg P(\hat{f}(0, X_1^\delta(0))) \\
\hat{H}(n+1) & = & \hat{H}(n) \wedge (P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))) \\
 & & \wedge (P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1))))
\end{array}$$

(this schematic NNF defines in fact the schematic characteristic formula). Note that $\hat{G}$ has a uniform definition and hence

$$\begin{array}{rcl}
\hat{G}(0) & = & \hat{H}(0) \wedge P(c) \wedge \neg P(\hat{f}(0,c)) \\
\hat{G}(n+1) & = & \hat{H}(n+1) \wedge P(c) \wedge \neg P(\hat{f}(n+1,c))
\end{array}$$

may be rewritten as

$$\hat{G}(n) \quad = \quad \hat{H}(n) \wedge P(c) \wedge \neg P(\hat{f}(n,c))$$

We construct the projection of $\mathscr{D}$ by constructing $\mathscr{D}_p'\colon \{(\delta_p', \rho(\delta_p', 0), \rho(\delta_p', n+1))\} \cup \mathscr{D}_p$, where $\rho(\delta_p', n) =$

$$\dfrac{(\delta_p, \emptyset)\colon S(\delta_p) \qquad (2)}{\nu'\colon \forall x(P(x) \to P(f(x))) \vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c))), \hat{G}(n)}\ \wedge_r + \mathscr{E}$$

$S(\delta_p)$ is the end-sequent of $\mathscr{D}_p$ (see below), $\hat{P}[\delta', \nu', \emptyset](\bar{X}, n) = \hat{G}(n)$, where $\bar{X}$ is a list of all occurring global variables and (2) is

$$\dfrac{\dfrac{\dfrac{P(c) \vdash P(c) \qquad \dfrac{\dfrac{P(\hat{f}(n,c)) \vdash P(\hat{f}(n,c))}{\vdash P(\hat{f}(n,c)), \neg P(\hat{f}(n,c))}\ \neg_r \qquad P(g(n,c)) \vdash P(g(n,c))}{P(\hat{f}(n,c)) \to P(g(n,c)) \vdash P(g(n,c)), \neg P(\hat{f}(n,c))}\ \to_l}{P(c), P(\hat{f}(n,c)) \to P(g(n,c)) \vdash P(g(n,c)), P(c) \wedge \neg P(\hat{f}(n,c))}\ \wedge_r}{P(\hat{f}(n,c)) \to P(g(n,c)) \vdash P(c) \to P(g(n,c)), P(c) \wedge \neg P(\hat{f}(n,c))}\ \to_r}{\vdash (P(\hat{f}(n,c)) \to P(g(n,c))) \to (P(c) \to P(g(n,c))), P(c) \wedge \neg P(\hat{f}(n,c))}\ \to_r$$

Now we construct $\mathscr{D}_p\colon \{(\delta_p, \rho(\delta_p, 0), \rho(\delta_p, n+1)\}.\ \rho(\delta_p, 0) =$

$$\dfrac{\dfrac{\dfrac{P(\hat{f}(0, X_1^\delta(0))) \vdash P(\hat{f}(0, X_1^\delta(0)))}{\vdash P(\hat{f}(0, X_1^\delta(0))), \neg P(\hat{f}(0, X_1^\delta(0)))}\,\neg_r}{\vdash P(\hat{f}(0, X_1^\delta(0))) \vee \neg P(\hat{f}(0, X_1^\delta(0)))}\,\vee_r}{\nu_0\colon \forall x(P(x) \to P(f(x))) \vdash \hat{H}(0)}\,w_l$$

and $\hat{P}[\delta, \nu_0, \Omega](\bar{X}, 0) = \hat{H}(0)$, where $\bar{X}$ are all the occurring global variables.

$\rho(\delta_p, n+1) =$

$$\dfrac{\nu_{n_1}\colon (\delta_p, \emptyset)\colon S(\delta_p) \qquad \nu_{n_2}\colon (1)}{\nu_{n+1}\colon \forall x(P(x) \to P(f(x))) \vdash \hat{H}(n+1)}\,\wedge_r + \mathscr{E}$$

where $S(\delta_p)$ is the end-sequent of $\rho(\delta_p, m)$ and (1) is

$$\dfrac{\dfrac{\dfrac{\nu_{n_5}\colon P(X_1^\delta(n+1)) \vdash P(X_1^\delta(n+1))}{\nu_{n_4}\colon \vdash P(X_1^\delta(n+1)), \neg P(X_1^\delta(n+1))}\,\neg_r}{\nu_{n_3}\colon \vdash P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))}\,\vee_r \qquad \nu_{n_2'}\colon (2)}{\nu_{n_2}\colon \forall x(P(x) \to P(f(x))) \vdash F_1}\,\wedge_r$$

where $F_1 = (P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))) \wedge (P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1))))$ and (2) is

$$\dfrac{\dfrac{\dfrac{\dfrac{\nu_{n_7'}\colon P(\hat{f}(n, X_1^\delta(n+1))) \vdash P(\hat{f}(n, X_1^\delta(n+1)))}{\nu_{n_5'}\colon \vdash P(\hat{f}(n, X_1^\delta(n+1))), \neg P(\hat{f}(n, X_1^\delta(n+1)))}\,\neg_r \quad \dfrac{\nu_{n_8'}\colon P(\hat{f}(n+1, X_1^\delta(n+1))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}{\nu_{n_6'}\colon P(f(\hat{f}(n, X_1^\delta(n+1)))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}\,\mathscr{E}}{\nu_{n_4'}\colon P(\hat{f}(n, X_1^\delta(n+1))) \to P(f(\hat{f}(n, X_1^\delta(n+1)))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1))), \neg P(\hat{f}(n, X_1^\delta(n+1)))}\,\to_l}{\nu_{n_3'}\colon P(\hat{f}(n, X_1^\delta(n+1))) \to P(f(\hat{f}(n, X_1^\delta(n+1)))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1)))}\,\vee_r}{\nu_{n_2'}\colon \forall x(P(x) \to P(f(x))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1)))}\,\forall_l$$

To construct $\hat{P}[\delta, \nu_{n+1}, \Omega](\bar{X}, n+1)$, which is $\hat{H}(n+1)$, we have to consider the binary inference above ν. As $\wedge_r$ operates on ancestors of Ω, we define $\hat{P}[\delta, \nu_{n+1}, \Omega](\bar{X}, n+1)$ as the conjunction of $\hat{P}[\delta, \nu_{n_1}, \Omega](\bar{X}, n+1)$ and $\hat{P}[\delta, \nu_{n_2}, \Omega](\bar{X}, n+1)$.

$\hat{P}[\delta, \nu_{n_1}, \Omega](\bar{X}, n+1) = \hat{P}[\delta, \nu_{n_1}, \Omega](\bar{X}, n) = \hat{H}(n)$ and

$\hat{P}[\delta, \nu_{n_2}, \Omega](\bar{X}, n+1) = F_1$: indeed, considering the sub-derivations (1) and (2) we have

$\hat{P}[\delta, \nu_i, \Omega](\bar{X}, n+1) = P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1)$ for $i \in \{n_5, n_4, n_3\}$ and

$\hat{P}[\delta, \nu_{n_7'}, \Omega](\bar{X}, n+1) = \hat{P}[\delta, \nu_{n_5'}, \Omega](\bar{X}, n+1) = \neg P(\hat{f}(n, X_1^\delta(n+1)))$,

$\hat{P}[\delta, \nu_{n_8'}, \Omega](\bar{X}, n+1) = \hat{P}[\delta, \nu_{n_6'}, \Omega](\bar{X}, n+1) = P(\hat{f}(n+1, X_1^\delta(n+1)))$,

$\hat{P}[\delta,\nu_j,\Omega](\bar{X},n+1) = \hat{P}[\delta,\nu_{n'_5},\Omega](\bar{X},n+1) \vee \hat{P}[\delta,\nu_{n'_6},\Omega](\bar{X},n+1) = \neg P(\hat{f}(n,X_1^\delta(n+1))) \vee P(\hat{f}(n+1,X_1^\delta(n+1)))$ for $j \in \{n'_4,n'_3,n'_2\}$ and hence,
$\hat{P}[\delta,\nu_{n_2},\Omega](\bar{X},n+1) = \hat{P}[\delta,\nu_{n_3},\Omega](\bar{X},n+1) \wedge \hat{P}[\delta,\nu_{n'_2},\Omega](\bar{X},n+1) = (P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))) \wedge (\neg P(\hat{f}(n,X_1^\delta(n+1))) \vee P(\hat{f}(n+1,X_1^\delta(n+1))))$.
Therefore $\hat{P}[\delta,\nu_{n+1},\Omega](\bar{X},n+1) = \hat{H}(n) \wedge F_1$.

$C^*(\delta')\colon \hat{P}[\delta',\nu',\emptyset](\bar{X},n)$ is the schematic characteristic term and $P^*(\delta')\colon P(\rho(\delta',n),\emptyset)$ is the schematic projection of $\mathscr{D}$.

Example 5.1.2. We construct a proof schema $\mathscr{D}_1\colon \{\delta_1,\rho(\delta_1,0),\rho(\delta_1,n+1)\}$ with end-sequent $S(\delta_1) = \hat{R}(n) \vdash \forall x\exists y P(x,y)$, where

$$\begin{array}{rcl} \hat{R}(0) & = & \forall x P(x,\hat{f}(a,0)) \\ \hat{R}(n+1) & = & \forall x P(x,\hat{f}(a,n+1)) \vee \hat{R}(n) \end{array}$$

and

$$\hat{f}(x,0) = x, \qquad \hat{f}(x,m+1) = g(\hat{f}(x,m)).$$

$\rho(\delta_1,0) =$

$$\dfrac{\dfrac{\dfrac{P(z,\hat{f}(a,0)) \vdash P(z,\hat{f}(a,0))}{\forall x P(x,\hat{f}(a,0)) \vdash P(z,\hat{f}(a,0))}\,\forall_l}{\forall x P(x,\hat{f}(a,0)) \vdash \exists y P(z,y)}\,\exists_r}{\forall x P(x,\hat{f}(a,0)) \vdash \forall x\exists y P(x,y)}\,\forall_r$$

and $\rho(\delta_1,n+1) =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(z,\hat{f}(a,n+1)) \vdash P(z,\hat{f}(a,n+1))}{\forall x P(x,\hat{f}(a,n+1)) \vdash P(z,\hat{f}(a,n+1))}\,\forall_l}{\forall x P(x,\hat{f}(a,n+1)) \vdash \exists y P(z,y)}\,\exists_r}{\forall x P(x,\hat{f}(a,n+1)) \vdash \forall x\exists y P(x,y)}\,\forall_r \qquad (\delta_1,\emptyset)\colon S(\delta_1)}{\forall x P(x,\hat{f}(a,n+1)) \vee \hat{R}(n) \vdash \forall x\exists y P(x,y), \forall x\exists y P(x,y)}\,\vee_l}{\forall x P(x,\hat{f}(a,n+1)) \vee \hat{R}(n) \vdash \forall x\exists y P(x,y)}\,c_r$$

Now we construct the main proof schema $\mathscr{D}_0\colon \{\delta_0,\rho(\delta_0,n,0),\rho(\delta_0,n,m+1)\} \cup \mathscr{D}_1$, where $\delta_0 > \delta_1$ and with end-sequent $S(\delta_0) = \hat{R}(n) \vdash \exists x\exists y P(x,y)$ via

$\rho(\delta_0,n,m) =$

$$\dfrac{(\delta_1,\emptyset)\colon S(\delta_1) \qquad \dfrac{\dfrac{\dfrac{P(\hat{f}(b,m),y_1)\vdash P(\hat{f}(b,m),y_1)}{P(\hat{f}(b,m),y_1)\vdash \exists x\exists y P(x,y)}\exists_r \times 2}{\exists y P(\hat{f}(b,m),y)\vdash \exists x\exists y P(x,y)}\exists_l}{\forall x\exists y P(x,y)\vdash \exists x\exists y P(x,y)}\forall_l}{\hat{R}(n)\vdash \exists x\exists y P(x,y)}\,cut$$

After regularization we obtain for $\mathscr{D}_1\colon \{\delta_1, \rho(\delta_1,0), \rho(\delta_1,n+1)\}$ (note that we have to introduce only one global variable X^{δ_1} and for simplicity we will name it X without superscript δ_1):

$\rho(\delta_1,0) =$

$$\dfrac{\dfrac{\dfrac{P(X(0),\hat{f}(a,0))\vdash P(X(0),\hat{f}(a,0))}{\forall x P(x,\hat{f}(a,0))\vdash P(X(0),\hat{f}(a,0))}\forall_l}{\forall x P(x,\hat{f}(a,0))\vdash \exists y P(X(0),y)}\exists_r}{\forall x P(x,\hat{f}(a,0))\vdash \forall x\exists y P(x,y)}\forall_r$$

and $\rho(\delta_1,n+1) =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(X(n+1),\hat{f}(a,n+1))\vdash P(X(n+1),\hat{f}(a,n+1))}{\forall x P(x,\hat{f}(a,n+1))\vdash P(X(n+1),\hat{f}(a,n+1))}\forall_l}{\forall x P(x,\hat{f}(a,n+1))\vdash \exists y P(X(n+1),y)}\exists_r}{\forall x P(x,\hat{f}(a,n+1))\vdash \forall x\exists y P(x,y)}\forall_r \qquad (\delta_1,\emptyset)\colon S(\delta_1)}{\forall x P(x,\hat{f}(a,n+1))\vee \hat{R}(n)\vdash \forall x\exists y P(x,y), \forall x\exists y P(x,y)}\vee_l}{\forall x P(x,\hat{f}(a,n+1))\vee \hat{R}(n)\vdash \forall x\exists y P(x,y)}\,c_r$$

and for $\mathscr{D}_0\colon \{\delta_0, \rho(\delta_0,n,0), \rho(\delta_0,n,m+1)\}\cup\mathscr{D}_1$ (again, instead of the global variable Y^{δ_0} we will use Y):

$\rho(\delta_0,n,m) =$

$$\dfrac{(\delta_1,\emptyset)\colon S(\delta_1) \qquad \dfrac{\dfrac{\dfrac{P(\hat{f}(b,m),Y(m))\vdash P(\hat{f}(b,m),Y(m))}{P(\hat{f}(b,m),Y(m))\vdash \exists x\exists y P(x,y)}\exists_r \times 2}{\exists y P(\hat{f}(b,m),y)\vdash \exists x\exists y P(x,y)}\exists_l}{\forall x\exists y P(x,y)\vdash \exists x\exists y P(x,y)}\forall_l}{\hat{R}(n)\vdash \exists x\exists y P(x,y)}\,cut$$

We construct the schematic projection $P^*(\delta_0)$ of the regularized proof schema $\mathscr{D}_0$ by constructing $\{(\delta_{0_p}, \rho(\delta_{0_p}, n, 0), \rho(\delta_{0_p}, n, m+1))\} \cup \mathscr{D}_{1_p}$, where $\rho(\delta_{0_p}, n, m) =$

$$\dfrac{(\delta_{1_p}, \emptyset)\colon S(\delta_{1_p}) \qquad \dfrac{\dfrac{P(\hat{f}(b,m), Y(m)) \vdash P(\hat{f}(b,m), Y(m))}{\vdash P(\hat{f}(b,0), Y(m)), \neg P(\hat{f}(b,m), Y(m))}\,\neg_r}{\vdash \exists x \exists y P(x,y), \neg P(\hat{f}(b,m), Y(m))}\,\exists_r \times 2}{\nu_0\colon \hat{R}(n) \vdash \exists x \exists y P(x,y), \neg P(\hat{f}(b,m), Y(m)) \wedge \hat{P}(X,n)}\,\wedge_r$$

where Ω is the set of cut-ancestors, $S(\delta_{1_p})$ is the end-sequent of $\mathscr{D}_{1_p}$ (see below), $\hat{P}[\delta_0, \nu_0, \emptyset](\bar{X}, n, m) = \neg P(\hat{f}(b,m), Y(m)) \wedge \hat{P}(X,n)$ ($\bar{X}$ always denotes all occurring global variables) and $\hat{P}(X,n)$:

$$\begin{aligned} \hat{P}(X,0) &= P(X(0), \hat{f}(a,0)) \\ \hat{P}(X,n+1) &= P(X(n+1), \hat{f}(a,n+1)) \vee \hat{P}(X,n) \end{aligned}$$

comes from the derivation $\rho(\delta_{1_p}, n)$, which will be defined below.

Now we construct $\mathscr{D}_{1_p}\colon \{(\delta_{1_p}, \rho(\delta_{1_p}, 0), \rho(\delta_{1_p}, n+1))\}$, where $\rho(\delta_{1_p}, 0) =$

$$\dfrac{P(X(0), \hat{f}(a,0)) \vdash P(X(0), \hat{f}(a,0))}{\nu_0'\colon \forall x P(x, \hat{f}(a,0)) \vdash P(X(0), \hat{f}(a,0))}\,\forall_l$$

where $\hat{P}[\delta_1, \nu_0', \Omega](\bar{X}, 0) = P(X(0), \hat{f}(a,0))$ and $\rho(\delta_{1_p}, n+1) =$

$$\dfrac{\dfrac{\dfrac{P(X(n+1), \hat{f}(a,n+1)) \vdash P(X(n+1), \hat{f}(a,n+1))}{\forall x P(x, \hat{f}(a,n+1)) \vdash P(X(n+1), \hat{f}(a,n+1))}\,\forall_l \qquad (\delta_{1_p}, \emptyset)\colon S(\delta_{1_p})}{\forall x P(x, \hat{f}(a,n+1)) \vee \hat{R}(n) \vdash P(X(n+1), \hat{f}(a,n+1)), P(X(n), \hat{f}(a,n))}\,\vee_l}{\nu_{n+1}'\colon \forall x P(x, \hat{f}(a,n+1)) \vee \hat{R}(n) \vdash P(X(n+1), \hat{f}(a,n+1)) \vee \hat{P}(X,n)}\,\vee_r$$

where $\hat{P}[\delta_1, \nu_{n+1}', \Omega](\bar{X}, n+1) = \hat{P}(X, n+1)$.

Therefore, the schematic characteristic formula is given by

$$\begin{aligned} \hat{Q}(X,Y,n,0) &= \neg P(\hat{f}(b,0), Y(0)) \wedge \hat{P}(X,n), \\ \hat{Q}(X,Y,n,m+1) &= \neg P(\hat{f}(b,m+1), Y(m+1)) \wedge \hat{P}(X,n), \\ \hat{P}(X,0) &= P(X(0), \hat{f}(a,0)) \\ \hat{P}(X,n+1) &= P(X(n+1), \hat{f}(a,n+1)) \vee \hat{P}(X,n) \end{aligned}$$

The following theorem shows the soundness of characteristic formula schemata and schematic proof projections.

Theorem 5.1.2. *Let σ be a parameter assignment and $\mathscr{D}$ be a regular schematic proof with maximal element δ_0. Then*

1. *the characteristic formula of $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow$ is $C^*(\delta_0)\sigma\downarrow$,*
2. *the projection of $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow$ is $P^*(\delta_0)\sigma\downarrow$.*

Proof.

1. We prove that for all proof symbols δ in $\mathscr{D}$ and all configurations Ω for δ we have that $C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow, \Omega)$ is $C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \Omega)\sigma\downarrow$. The proof is by induction on the number of proof symbols starting with the maximal symbol δ_0 and for every proof symbol we proceed by induction on σ.

   Let us consider the proof symbol δ_0 and the associated derivations $\rho(\delta_0, \vec{n}_{\delta_0}, 0)$ and $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)$. Assume that $\sigma(m_{\delta_0}) > 0$.

   Therefore, $C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \Omega)\sigma\downarrow = C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1), \Omega)\sigma'\downarrow$, where $\sigma'(\alpha) = \sigma(\alpha)$ if $\alpha \neq m_{\delta_0}$ and $\sigma'(m_{\delta_0}) = \sigma(m_{\delta_0} - 1)$. By a straightforward induction on formulas, one can prove that $C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1), \Omega)\sigma'\downarrow$ is obtained from $C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}+1)\sigma'\downarrow, \Omega)$ by replacing each formula $\mathrm{CF}(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}+1)\sigma'\downarrow, \nu, \Omega)$ by $C(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'}), \Omega^*)\Psi\sigma'\downarrow$, where ν is a leaf in the derivation of the form $\delta'\colon S(\delta')\Psi$ for a parameter replacement Ψ of $\vec{n}_{\delta'}, m_{\delta'}$ with respect to $\vec{n}_{\delta_0}$, $\delta > \delta'$ and Ω^* are all occurrences in $S(\delta')$ which are ancestors of $\Omega \cup \Omega'$ in $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}+1)\sigma'\downarrow$.

   By the induction hypothesis we have $C(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'}), \Omega^*)\Psi\sigma'\downarrow = C(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'})\sigma'\downarrow, \Omega^*)\Psi$. Therefore, $C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1), \Omega)\sigma'\downarrow = C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma'\downarrow, \Omega)$ and $C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \Omega)\sigma\downarrow = C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma'\downarrow, \Omega) = C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow$. The proof is similar if $\sigma(m_{\delta_0}) = 0$.

2. We prove that for all proof symbols δ in $\mathscr{D}$ and all configurations Ω for δ we have that $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow, \Omega)$ is $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \Omega)\sigma\downarrow$. The proof is by induction on the number of proof symbols starting with the maximal symbol δ_0 and for every proof symbol we proceed by induction on σ.

   Let us consider the proof symbol δ_0 and the associated derivations $\rho(\delta_0, \vec{n}_{\delta_0}, 0)$ and $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)$. Assume that $\sigma(m_{\delta_0}) > 0$.

   Therefore, $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \Omega)\sigma\downarrow = P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1), \Omega)\sigma'\downarrow$, where $\sigma'(\alpha) = \sigma(\alpha)$ if $\alpha \neq m_{\delta_0}$ and $\sigma'(m_{\delta_0}) = \sigma(m_{\delta_0} - 1)$. By a straightforward induction on derivations, one can prove that $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1), \Omega)\sigma'\downarrow$ is obtained from $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma'\downarrow, \Omega)$ by replacing each $\mathrm{Proj}(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma'\downarrow, \nu, \Omega)$ by $P(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'}), \Omega^*)\Psi\sigma'\downarrow$, where ν is a leaf in the derivation of the form $\delta'\colon S(\delta')\Psi$ for a parameter replacement Ψ of $\vec{n}_{\delta'}, m_{\delta'}$ with respect to $\vec{n}_{\delta_0}$, $\delta > \delta'$ and Ω^* are all occurrences in $S(\delta')$ which are ancestors of $\Omega \cup \Omega'$ in $\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma'\downarrow$.

   By the induction hypothesis we have $P(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'}), \Omega^*)\Psi\sigma'\downarrow = P(\rho(\delta', \vec{n}_{\delta'}, m_{\delta'})\sigma'\downarrow, \Omega^*)\Psi$. Therefore, $P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1), \Omega)\sigma'\downarrow = P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0} + 1)\sigma'\downarrow, \Omega)$ and

$P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \Omega)\sigma\downarrow = P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}+1)\sigma'\downarrow, \Omega) = P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0})\sigma\downarrow$. The proof is similar if $\sigma(m_{\delta_0}) = 0$.

□

The next step of the CERES method is to construct a refutation of the characteristic formula. In the schematic setting, we construct a simple refutation schema of the schematic characteristic formula. The simple refutation schema of the schematic characteristic formula from Example 5.1.1 is $\mathscr{D}$ from Example 4.4.1. In the example below we will construct a simple refutation schema of the schematic characteristic formula from Example 5.1.2.

Example 5.1.3. We construct a simple refutation schema for the schematic characteristic formula of Example 5.1.2:

$$\begin{aligned}
\hat{Q}(X, Y, n, 0) &= \neg P(\hat{f}(b, 0), Y(0)) \wedge \hat{P}(X, n), \\
\hat{Q}(X, Y, n, m+1) &= \neg P(\hat{f}(b, m+1), Y(m+1)) \wedge \hat{P}(X, n), \\
\hat{P}(X, 0) &= P(X(0), \hat{f}(a, 0)) \\
\hat{P}(X, n+1) &= P(X(n+1), \hat{f}(a, n+1)) \vee \hat{P}(X, n)
\end{aligned}$$

The invariant will be $\hat{Q}$ ($\hat{Q}(X, Y, 0, m)$ and $\hat{Q}(X, Y, n+1, m)$) (we do not need to introduce a new defined predicate symbol as in Example 4.4.1). Indeed, $\hat{Q}(X, Y, n, m)$ is derivable from $\hat{Q}(X, Y, n+1, m)$: $\rho(\delta', n, m) =$

$$\dfrac{\dfrac{\dfrac{\vdash \hat{Q}(X, Y, n+1, m)}{\vdash \neg P(\hat{f}(b, m), Y(m)) \wedge \hat{P}(X, n+1)}\, S\hat{Q}r}{\vdash \neg P(\hat{f}(b, m), Y(m))}\, \wedge_{r_1} \qquad \dfrac{(\varphi_1)}{\neg P(\hat{f}(b, m), Y(m)) \vdash \neg P(\hat{f}(b, m), Y(m)) \wedge \hat{P}(X, n)}}{\dfrac{\vdash \neg P(\hat{f}(b, m), Y(m)) \wedge \hat{P}(X, n)}{\vdash \hat{Q}(X, Y, n, m)}\, S\hat{Q}r^+}\, cut$$

where $\varphi_1 =$

$$\dfrac{\dfrac{(\varphi_0)}{\vdash \hat{P}(X, n)} \qquad \dfrac{}{\neg P(\hat{f}(b, m), Y(m)), \hat{P}(X, n) \vdash \neg P(\hat{f}(b, m), Y(m)) \wedge \hat{P}(X, n)}}{\neg P(\hat{f}(b, m), Y(m)) \vdash \neg P(\hat{f}(b, m), Y(m)) \wedge \hat{P}(X, n)}\, cut$$

and $\varphi_0 =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{Q}(X,Y,n+1,m)}{\vdash \neg P(\hat{f}(b,m),Y(m)) \wedge \hat{P}(X,n+1)}\,S\hat{Q}r}{\vdash \neg P(\hat{f}(b,m),Y(m))}\,\wedge_{r_1}}{P(\hat{f}(b,m),Y(m)) \vdash}\,\neg_r \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{Q}(X,Y,n+1,m)}{\vdash \neg P(\hat{f}(b,m),Y(m)) \wedge \hat{P}(X,n+1)}\,S\hat{Q}r}{\vdash \hat{P}(X,n+1)}\,\wedge_{r_2}}{\vdash P(X(n+1),\hat{f}(a,n+1)) \vee \hat{P}(X,n)}\,S\hat{P}r}{\vdash P(X(n+1),\hat{f}(a,n+1)),\hat{P}(X,n)}\,\vee_r}{\vdash \hat{P}(X,n)}\,res\sigma_0(n+1,m)$$

and $\sigma_0(n+1,m) = \{X(n+1) \leftarrow \hat{f}(b,m), Y(m) \leftarrow \hat{f}(a,n+1)\}$.

Now let us construct the simple refutation schema $\mathscr{D}$: $\{(\delta, \rho(\delta,n,0), \rho(\delta,n,m+1))\}$, where $S(\delta) = \vdash$. $\rho(\delta,n,0) =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{Q}(X,Y,n,0)}{\vdash \neg P(\hat{f}(b,0),Y(0)) \wedge \hat{P}(X,n)}\,B\hat{Q}r}{\vdash \neg P(\hat{f}(b,0),Y(0))}\,\wedge_{r_1}}{P(\hat{f}(b,0),Y(0)) \vdash}\,\neg_r \qquad \dfrac{\dfrac{\dfrac{\vdash \hat{Q}(X,Y,0,0)}{\vdash \neg P(\hat{f}(b,0),Y(0)) \wedge \hat{P}(X,0)}\,B\hat{Q}r}{\vdash \hat{P}(X,0)}\,\wedge_{r_2}}{\vdash P(X(0),\hat{f}(a,0))}\,B\hat{P}r}{\vdash}\,res\sigma_1(0)$$

where $\sigma_1(0) = \{X(0) \leftarrow \hat{f}(b,0), Y(0) \leftarrow \hat{f}(a,0)\}$, the initial sequent on the left branch is the maximal symbol $\hat{Q}$ and the initial sequent on the right branch is the invariant. $\rho(\delta,n,m+1) =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{Q}(X,Y,n,m+1)}{\vdash \neg P(\hat{f}(b,m+1),Y(m+1)) \wedge \hat{P}(X,n)}\,S\hat{Q}r}{\vdash \neg P(\hat{f}(b,m+1),Y(m+1))}\,\wedge_{r_1}}{P(\hat{f}(b,m+1),Y(m+1)) \vdash}\,\neg_r \qquad \dfrac{\dfrac{\dfrac{\vdash \hat{Q}(X,Y,0,m+1)}{\vdash \neg P(\hat{f}(b,m+1),Y(m+1)) \wedge \hat{P}(X,0)}\,B\hat{Q}r}{\vdash \hat{P}(X,0)}\,\wedge_{r_2}}{\vdash P(X(0),\hat{f}(a,0))}\,B\hat{P}r}{\vdash}\,res\sigma_1(m)$$

where $\sigma_1(m+1) = \{X(0) \leftarrow \hat{f}(b,m+1), Y(m+1) \leftarrow \hat{f}(a,0)\}$. Here again, the initial sequent $\vdash \hat{Q}(X,Y,0,m+1)$ is the invariant.

Let us construct the simple unification schema of $\mathscr{D}$, which will be denoted as the tuple $\Theta(\delta,n,0), \Theta(\delta,n,m+1))$:

$$\begin{aligned} \Theta(\delta,n,0) &= \sigma_1(0) \cup \Theta(\delta',0,0)\sigma_1(0) \\ \Theta(\delta,n,m+1) &= \sigma_1(m+1) \cup \Theta(\delta',0,m+1)\sigma_1(m+1). \end{aligned}$$

$\Theta(\delta', n, m)$ is given by:

$$\begin{aligned}\Theta(\delta', n, m) &= \sigma_0(n, m) \cup \Theta(\delta', n+1, m)\sigma_0(n, m)\\ \Theta(\delta', n+1, m) &= \sigma_0(n+1, m) = \{X(n+1) \leftarrow \hat{f}(b, m), Y(m) \leftarrow \hat{f}(a, n+1)\}.\end{aligned}$$

Here, for $n = \alpha$ and $m = 0$ we obtain the sequence

$$\{X(\alpha) \leftarrow \hat{f}(b, 0), X(\alpha - 1) \leftarrow \hat{f}(b, 0), \ldots, X(0) \leftarrow \hat{f}(b, 0), Y(0) \leftarrow \hat{f}(a, 0)\}$$

and for $n = 0$ and $m = \alpha$ we obtain the sequence

$$\{X(0) \leftarrow \hat{f}(b, \alpha), Y(\beta) \leftarrow \hat{f}(a, 0), Y(\beta - 1) \leftarrow \hat{f}(a, 0), \ldots, Y(0) \leftarrow \hat{f}(a, 0)\}.$$

Instead of using the simple refutation schema to construct a schematic CERES normal form, we directly proceed with the construction of a schematic Herbrand sequent. Correctness of the schematic Herbrand sequent will then be proved by showing that for every parameter assignment the schematic Herbrand sequent under this particular parameter is equal to the Herbrand sequent of the parametrized input proof.

5.2 Schematic Herbrand Sequent of the Schematic Projection

Our aim is to use schematic CERES for proof analysis as in Section 3.1. Therefore, we will extract a schematic Herbrand sequent from the schematic projection and combine it with the unification schema obtained from the simple refutation schema of the characteristic formula, in order to obtain the schematic Herbrand sequent of the input proof schema. In this section, we will focus on the extraction of the schematic Herbrand sequent from the schematic projection.

When we consider an **LKE**-proof φ of a prenex skolemized end-sequent and the proof projection $\pi\colon P(\varphi)$ then π is a cut-free **LKE**-proof of a skolemized prenex end sequent S; therefore π defines a Herbrand sequent and, for every formula F occurring in S, a set of substitutions $S(F, \pi)$ defining the Herbrand substitutions for the formula F in π. However, when we consider *schematic* **LKE**-proofs the situation changes. The initial sequents of such proofs may correspond to recursive calls and contain quantifiers (and thus are no axioms). To handle this more complex case we consider **LKE**-*derivations*.

We use the notion of thread, trace and yield as in Section 2.2. Note that when we admit quantified formulas in the initial sequents, which is the case for recursively defined proofs, we do not obtain a full but merely a partial Herbrand instance. By construction of a cut-free **LKE**-derivation of a skolemized prenex end-sequent, we observe the following.

Definition 5.2.1. Let π be a cut-free **LKE**-derivation of a skolemized prenex end-sequent S and let μ_0 be an occurrence of a formula $F\colon Qx_1 \ldots Qx_\beta.F'(x_1, \ldots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and F' is quantifier-free. Let $\tau\colon \mu_0, \ldots, \mu_\alpha$ be a trace of μ_0 in π. Then either

(a) $\mathit{formula}(\mu_\alpha) = F$ or

(b) there exists an $i \in \{1, \ldots, \beta - 1\}$ and terms $t_1, \ldots, t_i$ such that

$$\mathit{formula}(\mu_\alpha) = Qx_{i+1} \ldots Qx_\beta . F'(t_1, \ldots, t_i, x_{i+1}, \ldots, x_\beta) \text{ or}$$

(c) there exist terms $t_1, \ldots, t_\beta$ such that $\mathit{formula}(\mu_\alpha)$ is a subformula of $F'(t_1, \ldots, t_\beta)$ and $F'(t_1, \ldots, t_\beta)$ occurs in the yield of τ.

Definition 5.2.1 gives us the means to assign a substitution $hi(\tau)$ to every trace τ in a cut-free proof π of a skolemized prenex end-sequent such that either $hi(\tau)$ is a real Herbrand substitution or just a "partial" one.

Definition 5.2.2. Let π be a cut-free **LKE**-derivation of a skolemized prenex end-sequent S and let μ_0 be an occurrence of a formula $F\colon Qx_1 \ldots Qx_\beta . F'(x_1, \ldots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and F' is quantifier-free. Let $\tau\colon \mu_0, \ldots, \mu_\alpha$ be a trace of μ_0 in π and consider the cases (a), (b) and (c) in Definition 5.2.1. We define

- case (a): $hi(\tau) = \epsilon$ (ϵ is the empty substitution),
- case (b): $hi(\tau) = \{x_1 \leftarrow t_1, \ldots, x_i \leftarrow t_i\}$,
- case (c): $hi(\tau) = \{x_1 \leftarrow t_1, \ldots, x_\beta \leftarrow t_\beta\}$.

We are now ready to define the schematic set of Herbrand substitutions defined by a projection schema.

Definition 5.2.3 (Herbrand schemata for projection schemata)**.** Consider a regular proof schema $\mathscr{D}\colon \{(\delta, \rho(\delta, \vec{n}_\delta, 0), \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ and for some $\delta \in \mathbf{\Delta}$ the projection $\pi\colon P(\rho(\delta, \vec{n}, 0), \Omega)$. Let S be the end-sequent of π.

Let μ_0 be the occurrence of a formula $F(\vec{n}, 0)\colon Qx_1, \ldots, Qx_\beta . F'(x_1, \ldots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and $\tau\colon \mu_0, \ldots, \mu_\alpha$ be a trace of μ_0 in π. We distinguish three cases:

(a) $\mathit{formula}(\mu_\alpha) = F(\vec{n}, 0)$. This is only possible if the sequent occurring at node ν_α (for the corresponding thread $\nu_0, \ldots, \nu_\alpha$) is the end sequent S' of $P(\rho(\delta', \vec{n}, m)\Psi, \Omega')$ for some $\delta' < \delta$, for a parameter replacement Ψ and a configuration Ω' in S'. We define

$$S_S(\pi, \tau) = \bigcup \{S_S(\pi', \tau') \mid \tau' \in T(\mu_\alpha, \pi')\}$$

for $\pi' = P(\rho(\delta', \vec{n}, m)\Psi, \Omega')$

(b) $\mathit{formula}(\mu_\alpha) = Qx_{i+1} \ldots Qx_\beta . F'(t_1, \ldots, t_i, x_{i+1}, \ldots, x_\beta)$ for $i < \beta$. Like in case (a) the sequent occurring at the node ν_α is of the form S' where S' is the end-sequent of $P(\rho(\delta', \vec{n}, m)\Psi, \Omega')$ for some $\delta' < \delta$, for a parameter replacement Ψ and a configuration Ω' in S'. Here we define

$$S_S(\pi, \tau) = \{x_1 \leftarrow t_1, \ldots, x_i \leftarrow t_i\} \bigcup \{S_S(\pi', \tau') \mid \tau' \in T(\mu_\alpha, \pi')\}$$

for $\pi' = P(\rho(\delta', \vec{n}, m)\Psi, \Omega')$

(c) $formula(\mu_\alpha)$ is quantifier-free. We define $S_S(\pi, \tau) = \{hi(\tau)\}$.

In the end we define

$$S_S(\pi, \mu_0) = \bigcup\{S_S(\pi, \tau) \mid \tau \in T(\mu_0, \pi)\}.$$

Next we consider $\pi\colon P(\rho(\delta, \vec{n}, m+1), \Omega)$ and denote by S the end-sequent of π. Let μ_0 be the occurrence of a formula $F(\vec{n}, m+1)\colon Qx_1, \ldots, Qx_\beta.F'(x_1, \ldots, x_\beta)$ in S such that $Q \in \{\forall, \exists\}$ and $\tau\colon \mu_0, \ldots, \mu_\alpha$ be a trace of μ_0 in π. The definition is almost the same as for $\rho(\delta, \vec{n}, 0)$ above with the difference that we obtain an additional case:

- $formula(\mu_\alpha) = F(\vec{n}, m)$. Then μ_α is an occurrence of $F(\vec{n}, m)$ in $\pi'\colon P(\rho(\delta, \vec{n}, m)$. Let S' be the end-sequent of π'. We define

$$S_S(\pi, \tau) = \bigcup(S_S(\pi', \tau') \mid \tau' \in T(\mu_\alpha, \pi').$$

Again we define

$$S_S(\pi, \mu_0) = \bigcup\{S_S(\pi, \tau) \mid \tau \in T(\mu_0, \pi)\}.$$

Let δ_0 be the maximal proof symbol in $\mathbf{\Delta}$. Let μ_0 be an occurrence of a quantified formula in the end-sequents of $\rho(\delta_0, \vec{n}, 0)$ and $\rho(\delta_0, \vec{n}, m+1)$. Then the substitution schema of the projection schema $P(\rho(\delta, \vec{n}, m), \emptyset)$ is the defined as the tuple $S_S(P(\rho(\delta, \vec{n}, 0), \emptyset), \mu_0), S_S(P(\rho(\delta, \vec{n}, m+1), \emptyset), \mu_0)$.

Note that, in case δ is minimal, we obtain for $\rho(\delta, \vec{n}, 0)$ only the case (c) above applies and thus $S_S(\pi, \mu_0)$ is a set of Herbrand substitutions.

Example 5.2.1. Let $P^*(\delta')\colon P(\rho(\delta', n), \emptyset)$ be the schematic projection from Example 5.1.1. Let μ_0 be the occurrence of the formula $\forall x(P(x) \to P(f(x)))$ in the end-sequent of $\pi = P(\rho(\delta', n), \emptyset)$. Then

$$S_S(P(\rho(\delta', 0), \emptyset), \mu_0) = \emptyset \quad \text{and} \quad S_S(P(\rho(\delta', n+1), \emptyset), \mu_0) = \{x \leftarrow \hat{f}(n, X_1^\delta(n+1))\}.$$

Given a parameter assignment σ a substitution schema $S_S(P(\rho(\delta_0, \vec{n}, m), \emptyset), \mu_i)$ can be evaluated by computing $S_S(P(\rho(\delta_0, \vec{\alpha}, \beta), \emptyset), \mu_i)$ for the chosen numerals $\vec{\alpha}$ and β. Moreover, for $\pi = P(\rho(\delta_0, \vec{n}, m), \emptyset)$, $S_S(\pi, \mu_i)\sigma\downarrow = S(\pi\sigma\downarrow, \mu_i)$, where $S(\pi, \mu_i)$ is the set of Herbrand substitutions for the quantified formula occurrence μ_i in the end-sequent of π, see Definition 2.2.4.

Proposition 5.2.1. *Let $\pi = P(\rho(\delta_0, \vec{n}, m), \emptyset)$ be a projection schema, $S_S(\pi, \mu_i)$ a corresponding substitution schema and σ a parameter assignment. Then $S_S(\pi, \mu_i)\sigma\downarrow = S(\pi\sigma\downarrow, \mu_i)$.*

Proof. $S_S(\pi, \mu_i)$ is defined as the union of all $S_S(\pi, \tau)$ for all the traces $\tau\colon \mu_i, \ldots, \mu_\alpha$ of μ_i in π, i.e. $\tau \in T(\mu_i, \pi)$. $S_S(\pi, \mu_i)$ is by construction a set of s-substitutions, as for

some minimal δ only the case (c) in Definition 5.2.3 applies. Depending on whether for a given trace $\tau\colon \mu_i, \ldots, \mu_\alpha$ of μ_i $\mathit{formula}(\mu_\alpha)$ is quantifier-free or not, we take the set of substitutions $hi(\tau)$ or further expand π by evaluating the derivations that are linked to through labelled initial sequents. Computing $S_S(\pi, \mu_i)\sigma\downarrow$ means that we evaluate a set of s-substitutions under σ. As π and $\pi\sigma\downarrow$ have the same end-sequent modulo σ, quantified variables in quantified formulas in the end-sequent remain the same, only the terms that are used in the proof to eliminate the quantifiers change, again only modulo σ. Therefore, evaluating π under σ and then computing $S(\pi\sigma\downarrow, \mu_i)$ results in the same set of substitutions as when we compute $S_S(\pi, \mu_i)\sigma\downarrow$. □

5.3 Schematic Expansion Proof of the Schematic Projection

As demonstrated in Section 5.2 a schematic structure encoding the schematic Herbrand sequent can be extracted from a schematic projection. However, this only works if we consider projection schemata of prenex end-sequents. When we work with projection schemata of end-sequents that are not in prenex form, as for instance the projection schema in Example 5.1.2, in general we have to extract a structure that corresponds to the schematic expansion proof. It is well known that we can transform a proof of a sequent S that is not in prenex form into a derivation of the prenex form of S, however when considering schematic proofs it is not that simple any more. When trying to prenexify a derivation in a schematic setting we might run into several problems, one of which is the introduction of sequents of variable length. Allowing these kind of sequents would be fatal for this work, as the formal definition of such a transformed proof schema is simply out of range. Hence, the extraction of expansion proofs rather than Herbrand sequents is vital for proof analysis of schematic proofs.

In this section we will introduce the basic concepts for constructing a schematic expansion proof and demonstrate how these schematic structures can be extracted from proof schemata.

When we consider an **LKE**-proof φ of a skolemized end-sequent and the proof projection $\pi\colon P(\varphi)$ then π is a cut-free **LKE**-proof of a skolemized end-sequent S; therefore π defines a s-expansion proof. However, when we consider *schematic* **LKE**-proofs the situation changes. The initial sequents of such proofs may correspond to recursive calls and contain quantifiers (and thus are no axioms).

Let us first introduce some basic notions for schematic expansion proofs, by lifting the definitions of Section 2.3 to a schematic setting. Note that most definitions can be extended straight-forwardly, by considering schematic formulas instead of first-order formulas.

Definition 5.3.1. Schematic expansion trees and schematic dual expansion trees are defined inductively as follows (we extend the function *Sh* (shallow), which maps schematic expansion trees to schematic formulas to a schematic setting):

1. If A is a schematic formula and quantifier-free then A is a schematic expansion tree (and a schematic dual expansion tree) for A and $Sh(A) = A$.

2. If E is a schematic expansion tree then $\neg E$ is a schematic dual expansion tree and $Sh(\neg E) = \neg Sh(E)$.

3. If E is a schematic dual expansion tree then $\neg E$ is a schematic expansion tree and $Sh(\neg E) = \neg Sh(E)$.

4. If E_1 and E_2 are schematic (dual) expansion trees, then $E_1 \wedge E_2$, $E_1 \vee E_2$ are schematic (dual) expansion trees and $Sh(E_1 \wedge E_2) = Sh(E_1) \wedge Sh(E_2)$, the same for $\vee$.

5. If E_1 is a schematic dual expansion tree and E_2 is a schematic expansion tree then $E_1 \to E_2$ is a schematic expansion tree and $Sh(E_1 \to E_2) = Sh(E_1) \to Sh(E_2)$.

6. If E_1 is a schematic expansion tree and E_2 is a schematic dual expansion tree then $E_1 \to E_2$ is a schematic dual expansion tree and $Sh(E_1 \to E_2) = Sh(E_1) \to Sh(E_2)$.

7. Let $A(x)$ be a schematic formula and $t_1, ..., t_n$ $(n \geq 1)$ be a list of schematic terms. Let $E_1, ..., E_n$ be schematic expansion trees with $Sh(E_i) = A(t_i)$ for $i = 1, ..., n$; then $\exists x A(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n$ is a schematic expansion tree with $Sh(\exists x A(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n) = \exists x A(x)$.

8. Let $A(x)$ be a schematic formula and $t_1, ..., t_n$ $(n \geq 1)$ be a list of schematic terms. Let $E_1, ..., E_n$ be schematic dual expansion trees with $Sh(E_i) = A(t_i)$ for $i = 1, ..., n$; then $\forall x A(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n$ is a schematic dual expansion tree with $Sh(\forall x A(x) +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n) = \forall x A(x)$.

We extend the function Dp (deep), which maps expansion trees (and dual expansion trees) to quantifier-free formulas to a schematic setting.

Definition 5.3.2. Dp maps a (dual) expansion tree to a formula as follows:

$$\begin{aligned}
Dp(E) &= E \text{ for an atomic schematic expansion tree E,} \\
Dp(\neg E) &= \neg Dp(E), \\
Dp(E_1 \circ E_2) &= Dp(E_1) \circ Dp(E_2) \text{ for } \circ \in \{\wedge, \vee, \to\}, \\
Dp(\exists x A +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n) &= Dp(E_1) \vee ... \vee Dp(E_n), \\
Dp(\forall x A +^{t_1} E_1 +^{t_2} ... +^{t_n} E_n) &= Dp(E_1) \wedge ... \wedge Dp(E_n).
\end{aligned}$$

Now we can introduce the notions of a *schematic expansion proof* and a *schematic s-expansion tree.*

Definition 5.3.3 (schematic expansion proof)**.** Let ET be a schematic expansion tree of a schematic formula A without strong quantifiers. Then ET is called a schematic expansion proof of A from a set of axioms $\mathscr{A}$ if $Sh(ET) = A$ and $\mathscr{A} \models Dp(ET)$.

Definition 5.3.4 (schematic s-expansion tree)**.** The structure $S\colon\ \Gamma \vdash \Delta$ where $\Delta\colon Q_1, \dots Q_s$ is a multiset of schematic expansion trees, $\Gamma\colon P_1, \dots, P_r$ is a multiset of schematic dual expansion trees is called a *schematic s-expansion tree*. If $\neg\Gamma \vee \Delta$ (which stands for $\neg P_1 \vee \ldots \vee \neg P_r \vee Q_1 \vee \ldots \vee Q_s$) is a schematic expansion proof then S is called a *schematic s-expansion proof*. This expansion proof is the expansion proof associated with S; the schematic sequent

$$Seq(S)\colon\ \ Sh(P_1), ..., Sh(P_r) \to Sh(Q_1), ..., Sh(Q_s)$$

is the sequent associated with S.

Definition 5.3.5 (\-operator on schematic s-expansion trees)**.** Let $S\colon\ \Gamma \vdash \Delta, F$ be a schematic s-expansion tree, where $\Delta\colon Q_1, \dots, Q_s$ is a multiset of schematic expansion trees, $\Gamma\colon P_1, \dots, P_r$ is a multiset of schematic dual expansion trees and F a schematic formula without quantifiers. Then we define $S \setminus \ \vdash F = \Gamma \vdash \Delta$. In case $F = P_j$ $S\setminus \ \vdash F = \Gamma \vdash P_1, \dots P_{j-1}, P_{j+1}, \dots, P_r$. $S\setminus F \vdash$ is defined analogously.

In order to show how a schematic expansion proof is extracted from a schematic projection, we first need to extend the Merge operator on expansion trees to schematic expansion trees. The extension is straightforward: two schematic expansion trees T_1 and T_2 can be merged, if $Sh(T_1) = Sh(T_2)$.

Definition 5.3.6 (merge)**.** Let E_1, E_2 be schematic (dual) expansion trees such that $Sh(E_1) = Sh(E_2)$. We define the merge inductively on the complexity of E_1.

- If E_1 is a schematic atom then E_2 is a schematic atom too and $E_1 = E_2$; we define $\mathrm{Merge}_t(E_1,\ E_2) = E_1$.
- If $E_1 = \neg E_1'$. Then $E_2 = \neg E_2'$ for some E_2'. Let $\mathrm{Merge}_t(E_1', E_2') = E_3'$, then $\mathrm{Merge}_t(E_1, E_2) = \neg E_3'$.
- Let $E_1 = E_{11} \circ E_{12}$ for $\circ \in \{\wedge, \vee, \to\}$. Then $E_2 = E_{21} \circ E_{22}$ for some E_{21}, E_{22}. Let $E_1' = \mathrm{Merge}_t(E_{11}, E_{21})$ and $E_2' = \mathrm{Merge}_t(E_{21}, E_{22})$. Then $\mathrm{Merge}_t(E_1, E_2) = E_1' \circ E_2'$.
- Let $E_1 = Qx.A(x) +^{t_1} E_{11} + \cdots +^{t_n} E_{1n}$. Then E_2 is of the form $Qx.A(x) +^{s_1} E_{21} + \cdots +^{s_m} E_{2m}$. Then

$$\mathrm{Merge}_t(E_1, E_2) = Qx.A(x) +^{t_1} E_{11} + \cdots +^{t_n} E_{1n} +^{s_1} E_{21} + \cdots +^{s_m} E_{2m}.$$

The Merge operator can be easily extended to more than two schematic expansion trees. Let $T_1, \dots, T_n$ (for $n \geq 2$) be schematic (dual) expansion trees such that $Sh(T_i) = Sh(T_j)$ for all $i, j \in \{1, \dots, n\}$. Then we define

$$\begin{aligned} \mathrm{merge}_t(T_1, T_2) &= \mathrm{Merge}_t(T_1, T_2), \\ \mathrm{merge}_t(T_1, \dots, T_n) &= \mathrm{Merge}_t(\mathrm{merge}_t(T_1, \dots, T_{n-1}), T_n) \text{ for } n > 2. \end{aligned}$$

As in the first-order case, it is also possible to merge schematic s-expansion trees. As a schematic s-expansion tree is defined via multisets of schematic expansion trees, some schematic expansion trees might occur more than once either on the left or on the right. We restrict the merge of schematic s-expansion trees to normalized ones, where the shallow forms occur only once.

Definition 5.3.7. Let S be a schematic s-expansion tree then S is called normalized if $Seq(S)$ is normalized. Normalization of sequents extends to normalization of schematic sequents in a straight-forward way.

As already noted for the first-order case, restricting the merge to normalized schematic s-expansion proofs does not affect the generality of our approach, as schematic s-expansion trees can be easily transformed into normalized ones.

Now we are ready to define the merging of normalized schematic s-expansion trees.

Definition 5.3.8 (merge of schematic s-expansion trees)**.** Let S_1 and S_2 be two normalized schematic s-expansion trees and $S_1^* = Seq(S_1)$, $S_2^* = Seq(S_2)$. Then, by definition, S_1^*, S_2^* are normalized sequents. We define $\Gamma^* \vdash \Delta^* = S_1^* \cap S_2^*$, $\Pi_1^* \vdash \Lambda_1^* = S_1^* \setminus S_2^*$, $\Pi_2^* \vdash \Lambda_2^* = S_2^* \setminus S_1^*$. Then

$$\begin{aligned} S_1^* &= (\Gamma^* \vdash \Delta^*) \circ (\Pi_1^* \vdash \Lambda_1^*), \\ S_2^* &= (\Gamma^* \vdash \Delta^*) \circ (\Pi_2^* \vdash \Lambda_2^*). \end{aligned}$$

Then there exist schematic s-expansion trees $\Gamma \vdash \Delta$, $\Gamma' \vdash \Delta'$, $\Pi_1 \vdash \Lambda_1$ and $\Pi_2 \vdash \Lambda_2$ such that

$$\begin{aligned} S_1 &= (\Gamma \vdash \Delta) \circ (\Pi_1 \vdash \Lambda_1), \\ S_2 &= (\Gamma' \vdash \Delta') \circ (\Pi_2 \vdash \Lambda_2), \end{aligned}$$

where $Seq(\Gamma \vdash \Delta) = Seq(\Gamma' \vdash \Delta') = \Gamma^* \vdash \Delta^*$, $Seq(\Pi_1 \vdash \Lambda_1) = \Pi_1^* \vdash \Lambda_1^*$ and $Seq(\Pi_2 \vdash \Lambda_2) = \Pi_2^* \vdash \Lambda_2^*$. Note that the concatenation $\circ$ of schematic sequents can be directly extended to schematic s-expansion trees. Then there exist bijective mappings $\pi_l \colon \Gamma \to \Gamma'$ and $\pi_r \colon \Delta \to \Delta'$ with $\pi_l(T) = T'$ iff $Sh(T) = Sh(T')$ (the same for π_r). So assume

$$\begin{aligned} \Gamma \vdash \Delta &= T_1, \ldots, T_n \vdash T_{n+1}, \ldots, T_{n+m} \text{ and therefore} \\ \Gamma' \vdash \Delta' &= \pi_l(T_1), \ldots, \pi_l(T_n) \vdash \pi_r(T_{n+1}), \ldots, \pi_r(T_{n+m}). \end{aligned}$$

Now let $T_i^* = \mathrm{merge}_t(T_i, \pi_l(T_i))$ for $i = 1, \ldots, n$ and $T_i^* = \mathrm{merge}_t(T_i, \pi_r(T_i))$ for $i = n+1, \ldots, n+m$. Then we define

$$\mathrm{Merge}_s(S_1, S_2) = (T_1^*, \ldots, T_n^* \vdash T_{n+1}^*, \ldots, T_{n+m}^*) \circ (\Pi_1 \vdash \Lambda_1) \circ (\Pi_2 \vdash \Lambda_2).$$

Note that, by construction, $\mathrm{Merge}_s(S_1, S_2)$ is a normalized schematic s-expansion tree.

We extend the merging of schematic s-sequents to more than two as follows. Let $n \geq 2$ and $S_1, \ldots, S_n$ be normalized schematic s-expansion trees. Then

$$\begin{aligned} \mathrm{merge}_s(S_1, S_2) &= \mathrm{Merge}_s(S_1, S_2) \text{ for } n = 2, \\ \mathrm{merge}_s(S_1, \ldots, S_n) &= \mathrm{Merge}_s(\mathrm{merge}_s(S_1, \ldots, S_{n-1}), S_n) \text{ for } n > 2. \end{aligned}$$

The schematic s-expansion tree $\mathrm{merge}_s(S_1, \ldots, S_n)$ is also normal which can be verified by an obvious inductive argument.

Frequently we will write $\mathrm{merge}_s\{S_i \mid i = 1, \ldots, n\}$ for $\mathrm{merge}_s(S_1, \ldots, S_n)$. If no confusion arises we will frequently write merge instead of merge_t and merge_s.

Definition 5.3.9 (expansion proof schema from projection schema)**.**
Let $\mathscr{D}\colon \{(\delta, \rho(\delta, \vec{n}_\delta, 0), \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ be a regular proof schema. For some $\delta \in \mathbf{\Delta}$ we consider the projection $\pi\colon P(\rho(\delta, \vec{n}, 0), \Omega)$ and denote by S the end-sequent of π.

We define a transformation ET_S which maps π to a structure which we refer to as schematic expansion tree. We define the transformation inductively but the rules for $\neg_l, \neg_r, \vee_l, \vee_{r_1}, \vee_{r_2}$ are omitted, the transformation of the these rules being obvious.

Base case: π is an axiom. Then $\mathrm{ET}_S(\pi) = \pi$.

π is an initial sequent which is the end sequent S' of $P(\rho(\delta', \vec{n}, m)\Psi, \Omega')$ for some $\delta' < \delta$, for a parameter replacement Ψ and a configuration Ω' in S'. We define $\mathrm{ET}_S(\pi) = \mathrm{ET}_S(\pi')$ for $\pi' = P(\rho(\delta', \vec{n}, m)\Psi, \Omega')$.

If $\pi =$

$$\dfrac{\begin{matrix}(\varphi)\\ A, B, \Gamma \vdash \Delta\end{matrix}}{A \wedge B, \Gamma \vdash \Delta} \wedge_l$$

and $\mathrm{ET}_S(\varphi) = A^*, B^*, \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}_S(\pi) = A^* \wedge B^*, \Gamma^* \vdash \Delta^*$.

If $\pi =$

$$\dfrac{\begin{matrix}(\varphi_1)\\ \Gamma_1 \vdash \Delta_1, A\end{matrix} \qquad \begin{matrix}(\varphi_2)\\ \Gamma_2 \vdash \Delta_2, B\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B} \wedge_r$$

and $\mathrm{ET}_S(\varphi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\mathrm{ET}_S(\varphi_2) = \Gamma_2^* \vdash \Delta_2^*, B^*$, then $\mathrm{ET}_S(\pi) = \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A^* \wedge B^*$.

If $\pi =$

$$\frac{\begin{array}{cc}(\varphi_1) & (\varphi_1)\\ \Gamma_1 \vdash \Delta_1, A & B, \Gamma_2 \vdash \Delta_2\end{array}}{A \to B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \to_l$$

and $\mathrm{ET}_S(\varphi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\mathrm{ET}_S(\varphi_2) = B^*, \Gamma_2^* \vdash \Delta_2^*$, then $\mathrm{ET}_S(\pi) = A^* \to B^*, \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*$.

If $\pi =$

$$\frac{(\varphi)}{A, \Gamma \vdash \Delta, B} \to_r$$

and $\mathrm{ET}_S(\varphi) = A^*, \Gamma^* \vdash \Delta^*, B^*$, then $\mathrm{ET}_S(\pi) = \Gamma^* \vdash \Delta^*, A^* \to B^*$.

If $\pi =$

$$\frac{(\varphi)}{A\{x \leftarrow t\}, \Gamma \vdash \Delta} \forall_l$$

and $\mathrm{ET}_S(\varphi) = A\{x \leftarrow t\}^*, \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}_S(\pi) = \forall x A +^t A\{x \leftarrow t\}^*, \Gamma^* \vdash \Delta^*$.

If $\pi =$

$$\frac{(\varphi)}{\Gamma \vdash \Delta, A\{x \leftarrow t\}} \exists_r$$

and $\mathrm{ET}_S(\varphi) = \Gamma^* \vdash \Delta^*, A\{x \leftarrow t\}^*$, then $\mathrm{ET}_S(\pi) = \Gamma^* \vdash \Delta^*, \exists x A +^t A\{x \leftarrow t\}^*$.

If $\pi =$

$$\frac{(\varphi)}{\Gamma \vdash \Delta} w_l$$

and $\mathrm{ET}_S(\varphi) = \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}_S(\pi) = A, \Gamma^* \vdash \Delta^*$. Similarly for w_r.

If $\pi =$

$$\frac{(\varphi)}{A, A, \Gamma \vdash \Delta} c_l$$

and $\mathrm{ET}_S(\varphi) = A_1^*, A_2^*, \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}_S(\pi) = \mathrm{merge}\{A_1^*, A_2^*\}, \Gamma^* \vdash \Delta^*$. Similarly for c_r.

Next we consider $\pi\colon P(\rho(\delta, \vec{n}, m+1), \Omega)$ and denote by S the end-sequent of π. The definition is almost the same as for $\rho(\delta, \vec{n}, 0)$ above with the difference that we obtain an additional case:

π is an initial sequent which is the end sequent S' of $\pi'\colon P(\rho(\delta, \vec{n}, m), \Omega')$ for a configuration Ω' in S'. We define $\mathrm{ET}_S(\pi) = \mathrm{ET}_S(\pi')$.

Let δ_0 be the maximal proof symbol in $\mathbf{\Delta}$. Then the expansion proof schema of the projection schema $\pi\colon P(\rho(\delta_0, \vec{n}, m), \emptyset)$ is defined by $\mathrm{ET}_S(\pi)$.

$\mathrm{ET}_S(P(\rho(\delta, \vec{n}, m), \Omega))$ can be evaluated under a parameter assignment σ.

Definition 5.3.10. Let $\mathrm{ET}_S(\pi)$ for $\pi = P(\rho(\delta, \vec{n}, m), \Omega)$ be as in Definition 5.3.9 and let σ be a parameter assignment. Then $\mathrm{ET}_S(\pi)\sigma\downarrow$ is defined inductively on the structure of $\mathrm{ET}_S(\pi)$ (we skip some cases as their construction is analogous to the cases presented here):

- $\mathrm{ET}_S(\pi) = A \vdash A$ for atomic A. Then $\mathrm{ET}_S(\pi)\sigma\downarrow = A\sigma\downarrow \vdash A\sigma\downarrow$.
- $\mathrm{ET}_S(\pi) = A \circ B, \Gamma \vdash \Delta$, where $\circ \in \{\wedge, \vee, \to\}$. Then $\mathrm{ET}_S(\pi)\sigma\downarrow = A\sigma\downarrow \circ B\sigma\downarrow, \Gamma\sigma\downarrow \vdash \Delta\sigma\downarrow$.
- $\mathrm{ET}_S(\pi) = \forall x A +^t A\{x \leftarrow t\}, \Gamma \vdash \Delta$. Then $\mathrm{ET}_S(\pi)\sigma\downarrow = \forall x A\sigma\downarrow +^{t\sigma\downarrow} A\sigma\downarrow \{x \leftarrow t\sigma\downarrow\}, \Gamma\sigma\downarrow \vdash \Delta\sigma\downarrow$.
- $\mathrm{ET}_S(\pi) = \Gamma \vdash \Delta, \exists x A +^t A\{x \leftarrow t\}$. Then $\mathrm{ET}_S(\pi)\sigma\downarrow = \Gamma\sigma\downarrow \vdash \Delta\sigma\downarrow, \exists x A\sigma\downarrow +^{t\sigma\downarrow} A\sigma\downarrow \{x \leftarrow t\sigma\downarrow\}$.
- $\mathrm{ET}_S(\pi) = \mathrm{merge}\{A_1, A_2\}, \Gamma \vdash \Delta$. Then $\mathrm{ET}_S(\pi)\sigma\downarrow = \mathrm{merge}\{A_1\sigma\downarrow, A_2\sigma\downarrow\}, \Gamma\sigma\downarrow \vdash \Delta\sigma\downarrow$.

Example 5.3.1. Let $\pi\colon P(\rho(\delta, n, m), \emptyset)$ be the projection schema from Example 5.1.2. $\mathrm{ET}_S(\pi)$ is

$$\begin{aligned} \hat{R}_E(n) \quad \vdash \quad & \exists x \exists y P(x,y) +^{\hat{f}(b,m)} (\exists y P(\hat{f}(b,m), y) +^{Y(m)} P(\hat{f}(b,m), Y(m))), \\ & \neg P(\hat{f}(b,m), Y(m)) \wedge \hat{P}(X, n) \end{aligned}$$

where

$$\begin{aligned} \hat{R}_E(0) &= \forall x P(x,a) +^{X(0)} P(X(0), a) \\ \hat{R}_E(n+1) &= \forall x P(x, \hat{f}(a, n+1)) +^{X(n+1)} P(X(n+1), \hat{f}(a, n+1)) \vee \hat{R}_E(n) \end{aligned}$$

The evaluation under σ, where $\sigma(m) = \sigma(n) = 0$ is given by $\mathrm{ET}_S(P(\rho(\delta, 0, 0), \emptyset)) =$

$$\begin{aligned} \forall x P(x,a) +^{X(0)} P(X(0), a) \quad \vdash \quad & \exists x \exists y P(x,y) +^{b} (\exists y P(b,y) +^{Y(0)} P(b, Y(0))), \\ & \neg P(b, Y(0)) \wedge P(X(0), a) \end{aligned}$$

By evaluating a schematic expansion tree of some projection schema π^* for a parameter assignment σ we obtain the (first-order) expansion tree of the evaluated projection schema $\pi^*\sigma\downarrow$.

Proposition 5.3.1. *Let* $\mathrm{ET}_S(\pi)$ *for* $\pi = P(\rho(\delta, \vec{n}, m), \Omega)$ *be the expansion proof schema of a projection schema* π *and let* σ *be a parameter assignment. Then* $\mathrm{ET}_S(\pi)\sigma\downarrow = \mathrm{ET}(\pi\sigma\downarrow)$*, where* $\mathrm{ET}(\pi\sigma\downarrow)$ *is as in Definition 2.3.10.*

Proof. By construction of $\mathrm{ET}_S(\pi)$. Note that $\mathrm{ET}_S(\pi)$ is computed by evaluating π, i.e. by replacing labelled sequents of the form $S' = S(\delta')\Psi$, where $\delta' \in \mathbf{\Delta}$ with $\delta > \delta'$ and Ψ is a parameter replacement, by their derivations and labelled sequents of the form $S(\delta)\{m \leftarrow \alpha - 1\}$ by their derivations, to construct the schematic expansion proof. Evaluating the schematic expansion proof under σ results in a specific expansion proof for the chosen numerals $\vec{\alpha}$ and β.

However, this expansion proof can be directly obtained from $\pi\sigma\downarrow$, as the evaluation of π under σ is an **LKE**-derivation and thus, we can construct its expansion proof using ET, therefore $\mathrm{ET}_S(P(\rho(\delta_0, \vec{n}, \alpha), \emptyset))\sigma\downarrow = \mathrm{ET}(P(\rho(\delta_0, \vec{n}, \alpha), \emptyset)\sigma\downarrow)$. □

5.4 Extraction of Schematic Herbrand Sequents from Non-Normalized Proof Schemata

Analogous to the first-order case as described in Section 3.3 the construction of a schematic CERES normal form is superfluous for the extraction of schematic Herbrand sequents. We can extract a schematic Herbrand sequent from the projection schema and the simple refutation schema of the schematic characteristic formula.

To this aim, the Herbrand substitutions obtained from the schematic projection will be substituted with the unification schema $\Theta(\delta', \vec{n}, m)$ obtained from the simple refutation schema, constructing a schematic structure $\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))$ for a proof schema $\mathscr{D}$ and quantified formula occurrence μ_i. It can be shown that this schematic structure is a schematic representation of the Herbrand instances for μ_i of the normal form of the schematic proof $\mathscr{D}$. In fact, for any parameter assignment σ $\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow$ evaluates to the Herbrand instances for μ_i of the CERES normal form of $\mathscr{D}\sigma\downarrow$.

Definition 5.4.1. Let $\mathscr{D}\colon \{(\delta,\ \rho(\delta, \vec{n}_\delta, 0),\ \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ be a regular proof schema of a skolemized prenex end-sequent schema with schematic characteristic formula $C^*(\delta_0) = C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \emptyset)$ and projection schema $P^*(\delta_0) = P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \emptyset)$. Let $\mathscr{D}'\colon \{(\delta', \rho(\delta', \vec{n}_{\delta'}, 0), \rho(\delta', \vec{n}_{\delta'}, m_{\delta'} + 1)) \mid \delta' \in \mathbf{\Delta}'\}$ be the simple refutation schema of $C^*(\delta_0)$.

For each quantified formula occurrence μ_i in the end-sequents of the derivations $\rho(\delta_0, \vec{n}, 0)$ and $\rho(\delta_0, \vec{n}, m + 1)$ of $P^*(\delta_0)$ let $S_S(P^*(\delta_0), \mu_i)$ be the substitution schema of $P^*(\delta_0)$ for

μ_i. Then we define a set

$$\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m)) = \bigcup_{j=1}^{\alpha} S_S(P^*(\delta_0), \mu_i)\theta_j$$

for $\theta_1, \ldots, \theta_\alpha \in \Theta(\delta', \vec{n}, m)$ and $\Theta(\delta', \vec{n}, m)$ the unification schema of $\mathscr{D}'$, where δ' is the maximal symbol.

$\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))$ can be evaluated under any parameter assignment σ:

$$\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \bigcup_{j=1}^{\alpha} S_S(P^*(\delta_0), \mu_i)\sigma\downarrow\, \theta_j$$

for $\theta_1, \ldots, \theta_\alpha \in \Theta(\delta', \vec{n}, m)\sigma\downarrow$ and $\Theta(\delta', \vec{n}, m)$ the unification schema $\mathscr{D}'$.

The schematic Herbrand instances for μ_i of a proof schema $\mathscr{D}$ evaluate under all parameter assignments σ to the Herbrand instances for μ_i of $\mathscr{D}\sigma\downarrow$.

Theorem 5.4.1. *Let $\mathscr{D}$: $\{(\delta,\ \rho(\delta, \vec{n}_\delta, 0),\ \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ be a regular proof schema of a skolemized prenex end-sequent and σ a parameter assignment. Then $\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \overline{H}(\mathscr{D}\sigma\downarrow, \mu_i, \rho)$, where $\overline{H}(\mathscr{D}\sigma\downarrow, \mu_i, \rho)$ is as in Definition 3.3.1.*

Proof. Let δ_0 be the $<$-maximal symbol in $\mathscr{D}$. By definition

$$\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \bigcup_{j=1}^{\alpha} S_S(P^*(\delta_0), \mu_i)\sigma\downarrow\, \theta_j$$

for $\theta_1, \ldots, \theta_k \in \Theta(\delta', \vec{n}, m)\sigma\downarrow$, where $\Theta(\delta', \vec{n}, m)$ is the unification schema of the simple refutation schema $\mathscr{D}'$ of the schematic characteristic formula $C^*(\delta_0)$.

$\mathscr{D}'\sigma\downarrow$ is a RPL_0^{Ψ} refutation of the instantiated (schematic) characteristic formula $C^*(\delta_0)\sigma\downarrow$, which by Theorem 5.1.2 is the characteristic formula of $\mathscr{D}\sigma\downarrow$. Therefore the set $\{\theta_1, \ldots, \theta_\alpha\}$ for $\theta_1, \ldots, \theta_\alpha \in \Theta(\delta', \vec{n}, m)\sigma\downarrow$ is equal to a set $\{\theta_1, \ldots, \theta_\alpha\}$ for $\theta_1, \ldots, \theta_\alpha \in \Sigma(\rho)$, where ρ is the refutation $\mathscr{D}'\sigma\downarrow$. Thus,

$$\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \bigcup_{j=1}^{\alpha} S_S(P^*(\delta_0), \mu_i)\sigma\downarrow\, \theta_j$$

for $\theta_1, \ldots, \theta_\alpha \in \Sigma(\rho)$.

By Proposition 5.2.1 $S_S(P^*(\delta_0), \mu_i)\sigma\downarrow = S(P^*(\delta_0)\sigma\downarrow, \mu_i)$ and thus

$$\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \bigcup_{j=1}^{\alpha} S(P^*(\delta_0)\sigma\downarrow, \mu_i)\theta_j$$

for $\theta_1, \ldots, \theta_\alpha \in \Sigma(\rho)$.

By Theorem 5.1.2 the proof projection of $\mathscr{D}\sigma\downarrow$ is $\pi^* = P^*(\delta_0)\sigma\downarrow$, thus we have that

$$\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \bigcup_{j=1}^{\alpha} S(\pi^*, \mu_i)\theta_j$$

for $\theta_1, \ldots, \theta_\alpha \in \Sigma(\rho)$ and then, by definition of $\overline{H}(\mathscr{D}\sigma\downarrow, \mu_i, \rho)$, $\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \overline{H}(\mathscr{D}\sigma\downarrow, \mu_i, \rho)$. □

We have shown that for any parameter assignment σ the schematic Herbrand instance evaluated under σ is equal to the structure $\overline{H}(\mathscr{D}\sigma\downarrow, \mu_i, \rho)$. Therefore, the evaluated schematic Herbrand instance is equal to the Herbrand instance extracted from the evaluated CERES normal form.

Theorem 5.4.2. *Let* $\mathscr{D}$: $\{(\delta,\ \rho(\delta, \vec{n}_\delta, 0),\ \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ *be a regular proof schema of a skolemized prenex end-sequent,* σ *a parameter assignment and* π *the* CERES *normal form of* $\mathscr{D}\sigma\downarrow$. *Then* $\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = S(\pi, \mu_i)$.

Proof. By Theorem 5.4.1 we have that $\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \overline{H}(\mathscr{D}\sigma\downarrow, \mu_i, \rho)$. $\mathscr{D}\sigma\downarrow$ is an **LKE**-proof and allows the construction of a CERES normal form π of $\mathscr{D}\sigma\downarrow$. By Theorem 3.3.1 $\overline{H}(\mathscr{D}\sigma\downarrow, \mu_i, \rho) = S(\pi, \mu_i)$. Therefore, $\overline{H_S}(\mathscr{D}, \mu_i, \Theta(\delta', \vec{n}, m))\sigma\downarrow = S(\pi, \mu_i)$. □

Example 5.4.1. Let $P^*(\delta')$: $P(\rho(\delta', n), \emptyset)$ be the schematic projection from Example 5.1.1. We know from Example 5.2.1 that

$$S_S(P(\rho(\delta', 0), \emptyset), \mu_0) = \emptyset \quad \text{and} \quad S_S(P(\rho(\delta', n+1), \emptyset), \mu_0) = \{x \leftarrow \hat{f}(n, X_1^\delta(n+1))\}$$

for the occurrence μ_0 of the formula $\forall x(P(x) \to P(f(x)))$ in the end-sequent of $\pi = P(\rho(\delta', n), \emptyset)$.

From Example 4.4.2 we know that the unification schema is given by the tuple $(\Theta(\delta_0, 0), \Theta(\delta_0, n+1))$, where

$$\begin{aligned} \Theta(\delta_0, 0) &= \{\} \\ \Theta(\delta_0, n+1) &= \{X_1^\delta(n+1) \leftarrow c\} \cup \Theta(\delta, n), \end{aligned}$$

Thus, $\overline{H_S}(\mathscr{D}, \mu_0, \Theta(\delta_0, n)) = S_S(P^*(\delta_0), \mu_0)\theta_1$ for $\theta_1 \in \Theta(\delta_0, n)$ and $\theta_1 = \{X_1^\delta(n) \leftarrow c\}$ and hence we obtain for some numeral α the instances:

$$c, f(c), f(f(c)), \ldots, f^{\alpha-1}(c).$$

5.5 Extraction of Schematic Expansion Proofs from Non-Normalized Proof Schemata

Analogous to the first-order case as described in Section 3.4 the construction of a schematic CERES normal form is superfluous for the extraction of schematic expansion

proofs. Instead, we extract a schematic expansion proof from the projection schema and the simple refutation schema of the schematic characteristic formula.

To this aim, the schematic expansion proof will be composed of the schematic expansion proof of the projection schema after removal of the schematic expansion tree of the characteristic formula schema and then substituted with the substitutions given by the simple refutation schema. These substituted schematic expansion proofs can be merged, resulting in the expansion proof of the schematic CERES normal form.

Below we define a schematic expansion tree $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))$. It can be shown that this schematic structure is a schematic representation of the expansion proof of the proof schema $\mathscr{D}$. In fact, for any parameter assignment σ $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow$ evaluates to the expansion proof of $\mathscr{D}\sigma\downarrow$.

Definition 5.5.1. Let $\mathscr{D}$: $\{(\delta,\ \rho(\delta, \vec{n}_\delta, 0),\ \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ be a regular proof schema of a skolemized end-sequent with schematic characteristic formula $C^*(\delta_0) = C(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \emptyset)$ and projection schema $P^*(\delta_0) = P(\rho(\delta_0, \vec{n}_{\delta_0}, m_{\delta_0}), \emptyset)$. Let $\mathscr{D}'$ be the simple refutation schema of $C^*(\delta_0)$. Then we define

$$\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m)) = \text{merge}((\text{ET}_S(P^*(\delta_0))\setminus\vdash C^*(\delta_0))\theta_1, \ldots, (\text{ET}_S(P^*(\delta_0))\setminus\vdash C^*(\delta_0))\theta_\alpha)$$

for $\theta_1, \ldots, \theta_\alpha \in \Theta(\delta', \vec{n}, m)$ and $\Theta(\delta', \vec{n}, m)$ the unification schema of $\mathscr{D}'$ with maximal symbol δ'.

$\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))$ can be evaluated under any parameter assignment σ: $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \text{merge}((\text{ET}_S(\ P^*(\delta_0))\sigma\downarrow \setminus\vdash C^*(\delta_0)\sigma\downarrow)\theta_1, \ldots, (\text{ET}_S(P^*(\delta_0))\sigma\downarrow \setminus\vdash C^*(\delta_0)\sigma\downarrow)\theta_\alpha)$ for $\theta_1, \ldots, \theta_\alpha \in \Theta(\delta', \vec{n}, m)\sigma\downarrow$ and $\Theta(\delta', \vec{n}, m)$ the unification schema of $\mathscr{D}'$.

The schematic expansion proof of a proof schema $\mathscr{D}$ evaluates under all parameter assignments σ to the expansion proof of $\mathscr{D}\sigma\downarrow$.

Theorem 5.5.1. *Let $\mathscr{D}$: $\{(\delta,\ \rho(\delta, \vec{n}_\delta, 0),\ \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ be a regular proof schema of a skolemized end-sequent and σ a parameter assignment. Then $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \overline{E}(\mathscr{D}\sigma\downarrow, \rho)$, where $\overline{E}(\varphi, \rho)$ is as in Definition 3.4.1.*

Proof. Let δ_0 be the $<$-maximal symbol in $\mathscr{D}$. By definition $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow =$

$$\text{merge}((\text{ET}_S(P^*(\delta_0))\sigma\downarrow \setminus\vdash C^*(\delta_0)\sigma\downarrow)\theta_1, \ldots, (\text{ET}_S(P^*(\delta_0))\sigma\downarrow \setminus\vdash C^*(\delta_0)\sigma\downarrow)\theta_\alpha)$$

for $\theta_1, \ldots, \theta_\alpha \in \Theta(\delta', \vec{n}, m)\sigma\downarrow$.

$\mathscr{D}'\sigma\downarrow$ is a RPL_0^Ψ refutation of the instantiated characteristic formula $C^*(\delta_0)\sigma\downarrow$ and therefore the set $\{\theta_1, \ldots, \theta_\alpha\}$ for $\theta_1, \ldots, \theta_\alpha \in \Theta(\delta', \vec{n}, m)\sigma\downarrow$ is equal to a set $\{\theta_1, \ldots, \theta_\alpha\}$ for $\theta_1, \ldots, \theta_\alpha \in \Sigma(\rho)$, where ρ is the refutation $\mathscr{D}'\sigma\downarrow$. Thus, $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow =$

$$\text{merge}((\text{ET}_S(P^*(\delta_0))\sigma\downarrow \setminus\vdash C^*(\delta_0)\sigma\downarrow)\theta_1, \ldots, (\text{ET}_S(P^*(\delta_0))\sigma\downarrow \setminus\vdash C^*(\delta_0)\sigma\downarrow)\theta_\alpha)$$

for $\theta_1, \ldots, \theta_\alpha \in \Sigma(\rho)$.

By Proposition 5.3.1 $\mathrm{ET}_S(P^*(\delta_0))\sigma\downarrow = \mathrm{ET}(P^*(\delta_0)\sigma\downarrow)$ and thus $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow =$

$$\mathrm{merge}((\mathrm{ET}(\pi^*)\backslash\vdash C^*(\delta_0)\sigma\downarrow)\theta_1, \ldots, (\mathrm{ET}(\pi^*)\backslash\vdash C^*(\delta_0)\sigma\downarrow)\theta_\alpha)$$

for $\pi^* = P^*(\delta_0)\sigma\downarrow$ and $\theta_1, \ldots, \theta_\alpha \in \Sigma(\rho)$. As by Theorem 5.1.2 the proof projection of $\mathscr{D}\sigma\downarrow$ is π^* and the characteristic formula is $C^*(\delta_0)\sigma\downarrow$, we finally obtain by definition of $\overline{E}$ that $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \overline{E}(\mathscr{D}\sigma\downarrow, \rho)$. □

We have shown that for any parameter assignment σ the schematic expansion proof evaluated under σ is equal to the structure $\overline{E}(\mathscr{D}\sigma\downarrow, \rho)$. Therefore, the evaluated schematic expansion proof is equal to the expansion proof extracted from the evaluated CERES normal form.

Theorem 5.5.2. *Let* $\mathscr{D}\colon \{(\delta,\ \rho(\delta, \vec{n}_\delta, 0),\ \rho(\delta, \vec{n}_\delta, m_\delta + 1)) \mid \delta \in \mathbf{\Delta}\}$ *be a regular proof schema of a skolemized end-sequent,* σ *a parameter assignment and* π *the* CERES *normal form of* $\mathscr{D}\sigma\downarrow$*. Then* $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \mathrm{ET}(\pi)$.

Proof. By Theorem 5.5.1 we have that $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \overline{E}(\mathscr{D}\sigma\downarrow, \rho)$. $\mathscr{D}\sigma\downarrow$ is an **LKE**-proof and allows the construction of a CERES normal form π of $\mathscr{D}\sigma\downarrow$. By Theorem 3.4.2 $\overline{E}(\mathscr{D}\sigma\downarrow, \rho) = \mathrm{ET}(\pi)$. Therefore, $\overline{E_S}(\mathscr{D}, \Theta(\delta', \vec{n}, m))\sigma\downarrow = \mathrm{ET}(\pi)$. □

Example 5.5.1. Let $\pi\colon P(\rho(\delta, n, m), \emptyset)$ be the projection schema from Example 5.1.2. We know from Example 5.3.1 that $\mathrm{ET}_S(\pi) =$

$$\begin{aligned}\hat{R}_E(n) \quad \vdash \quad & \exists x \exists y P(x,y) +^{\hat{f}(b,m)} (\exists y P(\hat{f}(b,m), y) +^{Y(m)} P(\hat{f}(b,m), Y(m))),\\ & \neg P(\hat{f}(b,m), Y(m)) \wedge \hat{P}(X, n)\end{aligned}$$

where

$$\begin{aligned}\hat{R}_E(0) &= \forall x P(x,a) +^{X(0)} P(X(0), a)\\ \hat{R}_E(n+1) &= \forall x P(x, \hat{f}(a, n+1)) +^{X(n+1)} P(X(n+1), \hat{f}(a, n+1)) \vee \hat{R}_E(n)\end{aligned}$$

From Example 5.1.3 we know that the simple unification schema is given by the tuple $\Theta(\delta, n, 0), \Theta(\delta, n, m+1))$:

$$\begin{aligned}\Theta(\delta, n, 0) &= \sigma_1(0) \cup \Theta(\delta', 0, 0)\sigma_1(0)\\ \Theta(\delta, n, m+1) &= \sigma_1(m+1) \cup \Theta(\delta', 0, m+1)\sigma_1(m+1).\end{aligned}$$

where $\sigma_1(m) = \{X(0) \leftarrow \hat{f}(b,m), Y(m) \leftarrow \hat{f}(a,0)\}$ and $\Theta(\delta', n, m)$ is given by:

$$\begin{aligned}\Theta(\delta', n, m) &= \sigma_0(n,m) \cup \Theta(\delta', n+1, m)\sigma_0(n,m)\\ \Theta(\delta', n+1, m) &= \sigma_0(n+1, m) = \{X(n+1) \leftarrow \hat{f}(b,m), Y(m) \leftarrow \hat{f}(a, n+1)\}.\end{aligned}$$

Now we can construct

$$\overline{E_S}(\mathscr{D}, \Theta(\delta, n, m)) = (\mathrm{ET}_S(\pi)\backslash \vdash C^*(\delta_0))\sigma,$$

where $\sigma = \{X(n) \leftarrow \hat{f}(b,m), Y(m) \leftarrow \hat{f}(a,n)\}$.

$\mathrm{ET}_S(\pi)\backslash \vdash C^*(\delta_0) =$

$$\hat{R}_E(n) \quad \vdash \quad \exists x \exists y P(x,y) +^{\hat{f}(b,m)} (\exists y P(\hat{f}(b,m), y) +^{Y(m)} P(\hat{f}(b,m), Y(m)))$$

and hence we obtain $\overline{E_S}(\mathscr{D}, \Theta(\delta, n, m)) =$

$$\hat{R}'_E(n) \quad \vdash \quad \exists x \exists y P(x,y) +^{\hat{f}(b,m)} (\exists y P(\hat{f}(b,m), y) +^{\hat{f}(a,n)} P(\hat{f}(b,m), \hat{f}(a,n)))$$

where

$$\begin{aligned} \hat{R}'_E(0) &= \forall x P(x,a) +^{\hat{f}(b,m)} P(\hat{f}(b,m), a) \\ \hat{R}'_E(n+1) &= \forall x P(x, \hat{f}(a,n+1)) +^{\hat{f}(b,m)} P(\hat{f}(b,m), \hat{f}(a,n+1)) \vee \hat{R}'_E(n) \end{aligned}$$

Let σ be defined as $\sigma(n) = \sigma(m) = 0$, then $\overline{E_S}(\mathscr{D}, \Theta(\delta, n, m))\sigma\downarrow$ is given by

$$\forall x P(x,a) +^b P(b,a) \quad \vdash \quad \exists x \exists y P(x,y) +^b (\exists y P(b,y) +^a P(b,a))$$

Note that we can construct the deep function of the expansion trees above, which results in the Herbrand sequent

$$P(b,a) \vdash P(b,a).$$

5.6 A Note on the Extension to Equality

As in the first-order setting, handling equality rules is crucial to any proof analysis method. In fact, to be able to analyse Fürstenberg's proof we will need to add equality to our formalism. However, handling equality rules schematically can be a non-trivial task. The difficulty lies in allowing equality over terms, which leads to paramodulation rules that operate on any specific occurrence in an expression. This research line is worth investigating and left as future work. A different approach to the problem of equality in a schematic setting is the addition of equality axioms and can be performed in our formalism straightforwardly. Those new equality axioms can then be used to simulate paramodulation.

First of all it should be noted that the calculus **LKE** that is used in the construction of a proof schema already allows an equational theory $\mathscr{E}$, see Definition 4.2.3. To extend Definition 4.2.6 of a proof schema to a proof schema with equality, the only thing that needs to change is that the axiom set $\mathscr{A}_S$ is extended to an axiom set with equality axioms. The extraction of a schematic characteristic formula and the construction of a projection schema can be then performed straightforwardly.

The next interesting part is the construction of a simple resolution refutation schema. Here again, the calculus RPL_0^Ψ, see Definition 4.3.3, uses inference rules for the elimination and introduction of defined predicate symbols that resemble inferences of an equational theory.

For dealing with equality by resolution we can specify it with the following congruence axioms $\mathscr{E}_=$:

$$\begin{array}{rcll} & \vdash & x = x & \text{(reflexivity)} \\ x = y & \vdash & y = x & \text{(symmetry)} \\ x_1 = y_1 \wedge \ldots \wedge x_n = y_n & \vdash & f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) & \text{(monotonicity I)} \\ x_1 = y_1 \wedge \ldots \wedge x_n = y_n & \vdash & P(x_1, \ldots, x_n) = P(y_1, \ldots, y_n) & \text{(monotonicity II)} \end{array}$$

The monotonicity axioms are axiom schemata and we require one monotonicity I axiom for each non-constant n-ary (defined) function symbol f ($\hat{f}$) and one monotonicity II axiom for each (defined) predicate symbol P ($\hat{P}$). To handle the above axioms in simple refutation schemata, we need to integrate them into RPL_0^Ψ derivations. This can be done easily, by transforming e.g. monotonicity II first to

$$x_1 = y_1 \wedge \ldots \wedge x_n = y_n \vdash P(x_1, \ldots, x_n) \to P(y_1, \ldots, y_n)$$

and then to

$$x_1 = y_1 \wedge \ldots \wedge x_n = y_n \vdash \neg P(x_1, \ldots, x_n) \vee P(y_1, \ldots, y_n)$$

(analogously for the other axioms). Example 5.6.1 illustrates how such a simple refutation schema with equality rules might look like.

Example 5.6.1. We construct a simple refutation schema of Ψ:

$$\begin{array}{rcl} \hat{G}(0) & = & \hat{H}(0) \wedge a = c \wedge P(a) \wedge \neg P(\hat{f}(0, c)) \\ \hat{G}(n+1) & = & \hat{H}(n+1) \wedge \wedge a = c \wedge P(a) \wedge \neg P(\hat{f}(n+1, c)) \\ \hat{H}(0) & = & P(\hat{f}(0, X_1^\delta(0))) \vee \neg P(\hat{f}(0, X_1^\delta(0))) \\ \hat{H}(n+1) & = & \hat{H}(n) \wedge (P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))) \\ & & \wedge (P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1)))) \end{array}$$

We introduce $\hat{J}$, which will be used as an induction invariant (compare to $\hat{P}_i$ in case 4. of Definition 4.4.1), hence we obtain

$$\begin{array}{rcl} \hat{G}(0) & = & \hat{J}(0) \wedge \neg P(\hat{f}(0, c)) \\ \hat{G}(n+1) & = & \hat{J}(n+1) \wedge \neg P(\hat{f}(n+1, c)) \\ \hat{J}(0) & = & \hat{H}(0) \wedge a = c \wedge P(a) \\ \hat{J}(n+1) & = & \hat{H}(n+1) \wedge a = c \wedge P(a) \\ \hat{H}(0) & = & P(\hat{f}(0, X_1^\delta(0))) \vee \neg P(\hat{f}(0, X_1^\delta(0))) \\ \hat{H}(n+1) & = & \hat{H}(n) \wedge (P(X_1^\delta(n+1)) \vee \neg P(X_1^\delta(n+1))) \\ & & \wedge (P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1)))) \end{array}$$

First, we construct a resolution schema $\mathscr{D}_1$: $\{(\delta_1, \rho(\delta_1, 0), \rho(\delta_1, n+1)\} \cup \mathscr{D}_2$ with end-sequent schema $S(\delta_1) = \vdash P(\hat{f}(n, c))$. $\rho(\delta_1, 0) =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\delta_2, \{n \leftarrow 0\})\colon \vdash \hat{J}(0)}{\vdash \hat{H}(0) \wedge a = c \wedge P(a)}\, B\hat{J}r}{\vdash a = c \wedge P(a)}\, \wedge_{r_2}}{\vdash a = c}\, \wedge_{r_1}}{x = y \vdash \neg P(x) \vee P(y)}\, res\sigma \qquad \dfrac{\dfrac{\dfrac{(\delta_2, \{n \leftarrow 0\})\colon \vdash \hat{J}(0)}{\vdash \hat{H}(0) \wedge a = c \wedge P(a)}\, B\hat{J}r}{\vdash a = c \wedge P(a)}\, \wedge_{r_2}}{\vdash P(a)}\, \wedge_{r_2}}{\dfrac{\dfrac{\vdash \neg P(a) \vee P(c)}{\vdash \neg P(a), P(c)}\, \vee_r}{P(a) \vdash P(c)}\, \neg_r}$$
$$\vdash P(c) (= P(\hat{f}(0, c)))\; res$$

where $\sigma = \{x \leftarrow a, y \leftarrow c\}$ and $\rho(\delta_1, n+1) =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\delta_2, \{n \leftarrow n+1\})\colon \vdash \hat{J}(n+1)}{\vdash \hat{H}(n+1) \wedge a = c \wedge P(a)}\, S\hat{J}r}{\vdash \hat{H}(n+1)}\, \wedge_{r_1}}{\vdash P(\hat{f}(n+1, X_1^\delta(n+1))) \vee \neg P(\hat{f}(n, X_1^\delta(n+1)))}\, S\hat{H}r + \wedge_{r_2} \times 2}{\vdash P(\hat{f}(n+1, X_1^\delta(n+1))), \neg P(\hat{f}(n, X_1^\delta(n+1))))}\, \vee_r}{P(\hat{f}(n, X_1^\delta(n+1)))) \vdash P(\hat{f}(n+1, X_1^\delta(n+1)))}\, \neg_r}{(\delta_1, \{n \leftarrow n\})\colon \vdash P(\hat{f}(n, c)) \qquad \vdash P(\hat{f}(n+1, c))}\, res\{X_1^\delta(n+1) \leftarrow c\}$$

for $\delta_1 > \delta_2$.

Now we construct $\mathscr{D}_2 = \{(\delta_2, \rho(\delta_2, 0), \rho(\delta_2, n+1)\}$, where $\rho(\delta_2, 0) =$

$$\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{J}(0) \wedge \neg P(\hat{f}(0, c))}\, B\hat{G}r}{\vdash \hat{J}(0)}\, \wedge_{r_1}$$

and $\rho(\delta_2, n+1) =$

$$\dfrac{\dfrac{\vdash \hat{G}(n+1)}{\vdash \hat{J}(n+1) \wedge \neg P(\hat{f}(n+1, c))}\, B\hat{G}r}{\vdash \hat{J}(n+1)}\, \wedge_{r_1}$$

Now we are ready to define $\mathscr{D}$: $\{(\delta_0, \rho(\delta_0, 0), \rho(\delta_0, n{+}1)\} \cup \mathscr{D}_1$, where $S(\delta_0) = \vdash$: $\rho(\delta_0, 0) =$

$$\dfrac{(\delta_1, \{n \leftarrow 0\})\colon \vdash P(\hat{f}(0,c)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(0)}{\vdash \hat{H}(0) \wedge P(c) \wedge \neg P(\hat{f}(0,c))}\, B\hat{G}r}{\vdash P(c) \wedge \neg P(\hat{f}(0,c))}\, \wedge_{r_2}}{\vdash \neg P(\hat{f}(0,c))}\, \wedge_{r_2}}{P(\hat{f}(0,c)) \vdash}\, \neg_r}{\vdash}\, res$$

$\rho(\delta_0, n+1) =$

$$\dfrac{(\delta_1, \{n \leftarrow n+1\})\colon \vdash P(\hat{f}(n+1,c)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{G}(n+1)}{\vdash \hat{J}(n+1) \wedge \neg P(\hat{f}(n+1,c))}\, S\hat{G}r}{\vdash \neg P(\hat{f}(n+1,c))}\, \wedge_{r_2}}{P(\hat{f}(n+1,c)) \vdash}\, \neg_r}{\vdash}\, res$$

As a last step we have to show that $\hat{J}(n)$ is derivable from $\hat{J}(n+1)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{J}(n+1)}{\vdash \hat{H}(n+1) \wedge P(c)}\, S\hat{J}r}{\vdash \hat{H}(n+1)}\, \wedge_{r_1}}{\vdash \hat{H}(n)}\, S\hat{H}r + \wedge_{r_1} \qquad \dfrac{\dfrac{\vdash \hat{J}(n+1)}{\vdash \hat{H}(n+1) \wedge P(c)}\, S\hat{J}r}{\vdash P(c)}\, \wedge_{r_4}}{\vdash \hat{H}(n) \wedge P(c)}\, \wedge_r}{\vdash \hat{J}(n)}\, S\hat{J}^+r$$

Resolution with $\mathscr{E}_=$ might however cause too many and often unnecessary new clauses. Overcoming this problem of non-efficiency is only possible by using a schematic version of the paramodulation rule.

CHAPTER 6

Implementation and Experiments in Gapt

In this chapter we will describe some of our implementations and experiments with mathematical proofs. The underlying system of all our implementations is Gapt[1] (General Architecture for Proof Theory) [EHR+16], which is a framework for implementing proof transformations and provides numerous algorithms for the analysis, transformation, and construction of proofs in various formal calculi. Gapt is implemented in Scala and licensed under the GNU General Public License. The software is available under `https://logic.at/gapt`. Gapt initially started as an implementation of the CERES method. The system which provided the foundational architecture for the current version of Gapt was developed for the analysis of Fürstenberg's proof of the infinitude of primes [BHL+08]. Gapt also provides an interface for importing proofs from most major theorem provers and exporting proofs and other structures in TPTP format.

For information on how to install and use the system Gapt we refer to the Gapt User Manual[2]. Gapt opens in a Scala interactive shell (scala>) which can be used to run all the commands provided by the system.

The implementation of this work in Gapt can be split in two parts. The first implementation is concerned with proof analysis with CERES for first-order logic with equality, as introduced in Section 3.5. To this aim, the method for Herbrand sequent or expansion proof extraction from the proof projections and the resolution refutation, as described by the algorithm EXP_{new} (see Definition 3.5.15) has been implemented. The second part is concerned with the implementation of the schematic proof analysis method based on schematic CERES. In a first step towards a full implementation we have extended Gapt with a schematic formalism that allows for the construction of formalized schematic

[1] http://www.logic.at/gapt/

[2] http://www.logic.at/gapt/downloads/gapt-user-manual.pdf

proofs. Moreover, it is possible to analyse schematic proofs by constructing a schematic characteristic formula, see Definition 5.1.2, or a schematic characteristic clause set. With these implementations it is already possible to analyse interesting schematic proofs, however a full implementation of schematic CERES, in particular the construction of a proof projection and the refutation, are still missing and left for future work.

6.1 Experiments with Clausal CERES + Equality

This section explains how to run the algorithm EXP_{new} (Definition 3.5.15) followed by a discussion of results obtained by experiments with formal proofs.

Under "Proof Examples" in the Gapt system there is a set of functions that generate proofs of some end-sequent. In the following demonstration we will use the proof from Example 3.5.1, which is referred to as CERESExpansionExampleProof.proof in Gapt. The sequence of commands

```
scala> val p = CERESExpansionExampleProof.proof
scala> val p1 = CERES( p )
scala> prooftool( p1 )
```

instantiates the proof from Example 3.5.1 and stores it in the variable p, which is used as input for the method CERES. The variable p1 stores the generated CERES normal form. Note that the outputs stored in variables p and p1 are strings representing the proofs. To obtain a proof in a tree-like structure prooftool can be used, which is a viewer for proofs and other elements also implemented in Gapt [DLL+13]. The algorithm EXP, which extracts expansion proofs from CERES normal forms is implemented in Gapt as the method LKToExpansionProof. Note that this method extracts an expansion proof from any **LK**-proof and not only from CERES normal forms. More information on expansion trees in Gapt can be found in [HLRR13] and [Rei15]. The following demonstration shows how to obtain expansion proofs and Herbrand sequents (corresponding to the deep function of an expansion proof) from the CERES normal form p1 (defined in the demonstration above):

```
scala> val exp = LKToExpansionProof( p1 )
scala> prooftool( exp )
scala> val dp = exp.deep
scala> prooftool( dp )
```

The output stored in exp is a string representing the expansion proof of p1; using prooftool a better representation can be obtained. Note that also the deep function can be displayed in prooftool. The next demonstration shows how to use the algorithm EXP_{new}, which is implemented as the method CERESExpansionProof (within the CERES implementation)

```
scala> val p = CERESExpansionExampleProof.proof
scala> val exp = CERES.CERESExpansionProof( p, Escargot )
scala> val dp = exp.deep
```

The output stored in variable dp is a string representing the deep function of the expansion proof extracted from the proof projections and the corresponding ground PR refutation. Note that we use a simple built-in prover called Escargot. Instead of using Escargot, any other resolution prover supported by Gapt may be used (Gapt includes interfaces to several first-order theorem provers, such as Prover9, E Prover and LeanCoP, for more details we refer to the Gapt User Manual).

To measure the complexity of algorithms we use the command `time`, provided by the Gapt system. This command measures the time in ms that is needed on the system in use to perform a method. We performed several experiments with proofs containing cuts and measured the speed-up in time via

```
scala> time{ LKToExpansionProof( CERES( p ) ) }
scala> time{ CERES.CERESExpansionProof( p ) }
```

Our experiments have shown that there is a speed-up in the computation of expansion proofs with the algorithm EXP_{new} compared to the algorithm EXP already for small and simple proofs like in Example 3.5.1: our best result for the algorithm EXP is $47ms$, on the other hand, with the algorithm EXP_{new} we obtained $18ms$. Since even for a small and simple proof like the proof in Example 3.5.1 a speed-up is obtained, it is clear that we can increase the speed-up when we consider more complex and longer proofs. Therefore we analyzed more complicated proofs provided by Gapt:

- `lattice.proof` and
- `tape.proof`.

The first proof emerges from a simple example in lattice theory. It is a proof of one direction of the equivalence of different definitions of the concept of a lattice. Indeed, there are several different definitions of a lattice, but they are all equivalent. In particular we will focus on three definitions that we will refer to as $L1$-, $L2$- and $L3$-lattice. They are all based on the notion of semi-lattice.

Definition 6.1.1. A semi-lattice is a set L together with an operation $\circ$ which is

- commutative: $\forall x \forall y (x \circ y = y \circ x)$,
- associative: $\forall x \forall y \forall z ((x \circ y) \circ z = x \circ (y \circ z))$,
- idempotent: $\forall x (x \circ x = x)$.

Definition 6.1.2 (Lattice: definition 1)**.** A $L1$-lattice is a set L together with operations $\cap$ (meet) and $\cup$ (join) such that both $\langle L, \cap \rangle$ and $\langle L, \cup \rangle$ are semi-lattices and $\cap$ and $\cup$ are inverse in the sense that $\forall x \forall y (x \cap y = x \leftrightarrow x \cup y = y)$.

Definition 6.1.3 (Lattice: definition 2)**.** A $L2$-lattice is a set L together with operations $\cap$ and $\cup$ such that both $\langle L, \cap \rangle$ and $\langle L, \cup \rangle$ are semi-lattices and the absorption laws $\forall x \forall y ((x \cap y) \cup x = x)$ and $\forall x \forall y ((x \cup y) \cap x = x)$ hold.

Definition 6.1.4 (Lattice: definition 3)**.** A $L3$-lattice is a partially ordered set[1] $\langle S, \leq \rangle$ such that for each two elements x, y of S there exists

- a greatest lower bound (GLB) $glb(x, y)$, i.e. $\forall x \forall y (glb(x, y) \leq x \wedge glb(x, y) \leq y \wedge \forall z ((z \leq x \wedge z \leq y) \rightarrow z \leq glb(x, y)))$,
- a least upper bound (LUB) $lub(x, y)$, i.e. $\forall x \forall y (x \leq lub(x, y) \wedge y \leq lub(x, y) \wedge \forall z ((x \leq z \wedge y \leq z) \rightarrow lub(x, y) \leq z))$.

Usually, one proves the equivalence of several different definitions or statements by a cycle of implications. This reduces the size of the proof, but on the other hand does not provide direct proofs between the statements. More precisely, the following two propositions imply that $L1$-lattices are $L2$-lattices.

Proposition 6.1.1. *$L1$-lattices are $L3$-lattices.*

Proposition 6.1.2. *$L3$-lattices are $L2$-lattices.*

The proof of the statement "$L1$-lattices are $L2$-lattices" is however not a direct one and uses the notion of partially ordered sets. This notion occurs neither in $L1$ nor in $L2$. Using CERES we can automatically obtain a direct formal proof of the desired statement. The analysis of the lattice proof as performed in CERES follows the steps below (see also in [HLWP08]):

1. Formalization of the lattice proof in a sequent calculus.
2. Cut-elimination of the formalized lattice proof using CERES.
3. Extraction of the Herbrand sequent of the CERES normal form.
4. Use of the Herbrand sequent to interpret the resulting proof and obtain a new direct informal proof.

[1] We use the general definition of a partial order $\leq$ on a set S, which is reflexive, anti-symmetric and transitive.

After extracting a Herbrand sequent form the CERES normal form we can construct an informal analytic proof of the desired theorem. This proof is based on the CERES normal form but uses only the information about the variable instantiations contained in its extracted Herbrand sequent. In the analysis the formulas from the Herbrand sequent are used as a guide to construct an analytic mathematical proof.

Another interesting proof for analyzing and comparing the implemented methods is tape.proof, which is a proof of the statement "Given an infinite tape labelled by zeros and ones there are two cells with the same value.". The tape proof is from [Urb00] and was formalized in **LK** and analyzed by CERES in [BHL+05, BHL+06].

The proof proceeds by two lemmas:

1. There are infinitely many cells labelled by 0.
2. There are infinitely many cells labelled by 1.

These lemmas are eliminated by CERES a a more direct argument is obtained in the resulting proof.

The tape proof is a subcase of the proof of the unbounded pigeonhole principle, for more information we refer to Section 4.2 of [Urb00] and Section 3 of [BHL+05]. Note that the formalization of the tape proof as described in [Urb00] is realized in Gapt as tabeUrban.proof.

Figure 6.1 shows our results on experiments with the proofs lattice.proof, tape.proof and tapeUrban.proof. In all three cases, the method EXP_{new} outperforms the method EXP.

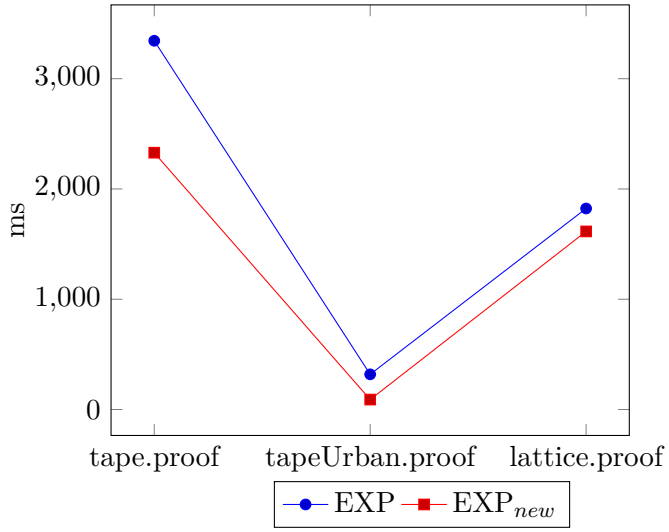

Figure 6.1: Comparison of the methods EXP and EXP_{new} for the proofs: tape.proof, tapeUrban.proof, lattice.proof.

Furthermore, we analyzed the two methods based on CERES in comparison to reductive cut-elimination methods. The Gapt system contains an implementation of Gentzen-style reductive cut-elimination, which can be used by calling the method ReductiveCutElimination. For this analysis we used several proofs provided by Gapt, as for instance simple proofs containing cuts (fol1.proof and fol2.proof), formalizations of the so-called poset proof (poset.proof), formalizations of the pigeonhole principle (Pi2Pigeonhole.proof and Pi3Pigeonhole.proof) and some "artificial" proofs in the sense that we introduced cuts to originally cut-free proofs. Indeed, Gapt provides a cut-introduction procedure called CutIntroduction(p), which in some cases compresses the cut-free proof p by adding cuts. We use this cut-introduction method on sequences of cut-free proofs, as for instance the sequence of proofs LinearExampleProof(n), constructing cut-free proofs of sequents $P(0), \forall x(P(x) \to P(s(x))) \vdash P(s^n(0))$, where $n \geq 0$, the sequence of proofs LinearEqExampleProof(n), constructing cut-free proofs of sequents Refl, Trans, $\forall x(f(x) = x) \vdash f^n(a) = a$ and the sequence of proofs FactorialFunctionEqualityExampleProof(n), constructing proofs of $f(n) = g(n, 1)$, where f is the head recursive and g the tail recursive formulation of the factorial function. Introducing cuts to these three sequences of proofs results in sequences of shorter proofs. Table 6.1 summarizes our results and shows that not only EXP_{new} is faster that EXP, but both CERES based methods clearly outperform the method based on Gentzen-style cut-elimination.

Table 6.1: Comparison of the three different methods for the extraction of expansion proofs: based on Gentzen's reductive method, methods EXP and EXP_{new}.

proof	reductive	EXP	EXP_{new}
fol1.proof	68 ms	27 ms	17 ms
fol2.proof	53 ms	30 ms	21 ms
Pi2Pigeonhole.proof	1350 ms	680 ms	170 ms
Pi3Pigeonhole.proof	842 ms	552 ms	181 ms
poset.proof.cycleImpliesEqual3	1621 ms	360 ms	147 ms
poset.proof.cycleImpliesEqual4	6877 ms	850 ms	250 ms
CutIntroduction(LinearExampleProof(4))	138 ms	70 ms	26 ms
CutIntroduction(LinearExampleProof(8))	294 ms	42 ms	31 ms
CutIntroduction(LinearExampleProof(10))	577 ms	32 ms	28 ms
CutIntroduction(LinearExampleProof(15))	890 ms	36 ms	26 ms
CutIntroduction(LinearExampleProof(18))	1329 ms	54 ms	25 ms
CutIntroduction(LinearExampleProof(19))	1205 ms	85 ms	51 ms
CutIntroduction(LinearEqExampleProof(2))	190 ms	72 ms	32 ms
CutIntroduction(LinearEqExampleProof(5))	924 ms	98 ms	34 ms
CutIntroduction(LinearEqExampleProof(10))	2640 ms	113 ms	43 ms
CutIntroduction(LinearEqExampleProof(15))	6510 ms	134 ms	46 ms
CutIntroduction(LinearEqExampleProof(16))	10875 ms	163 ms	53 ms
CutIntroduction(LinearEqExampleProof(18))	12423 ms	455 ms	97 ms
CutIntroduction(FactorialFunctionEqualityExampleProof(3))	3525 ms	500 ms	360 ms
CutIntroduction(FactorialFunctionEqualityExampleProof(4))	8473 ms	795 ms	590 ms
CutIntroduction(FactorialFunctionEqualityExampleProof(5))	20006 ms	1430 ms	930 ms

Another interesting sequence of cut-free proofs provided by Gapt is formalized in SquareDiagonalExampleProof(n), which constructs a sequence of cut-free proofs of

$$P(0,0), \forall x \forall y(P(x,y) \to P(s(x),y)), \forall x \forall y(P(x,y) \to P(x,s(y))) \vdash P(s^n(0), s^n(0)),$$

where $n \geq 0$. For every n the constructed proof goes along the diagonal of P, i.e. one x-step, then one y-step, etc. Cuts can be introduced to this sequence of proofs, however the proof obtained by CutIntroduction(SquareDiagonalExampleProof(n)) is not necessarily longer the higher the value for n is. More precisely, the proof SquareDiagonalExampleProof(n) might not get as much compressed as the proof SquareDiagonalExampleProof(n+1) by introducing cuts, leading to a shorter proof for $n+1$. Therefore, the method based on reductive cut-elimination is not always slower for higher values of n. In fact, as can be observed in Figure 6.2,

the computing time for the method based on reductive cut-elimination on the proof CutIntroduction(SquareDiagonalExampleProof(7)) is 6235ms, while on the proof CutIntroduction(SquareDiagonalExampleProof(8)) it is 1278ms. The analysis in Figure 6.2 is performed for values $2 \leq n \leq 18$, as for higher values the constructed proof is too long to be analyzed on the operating system in use.

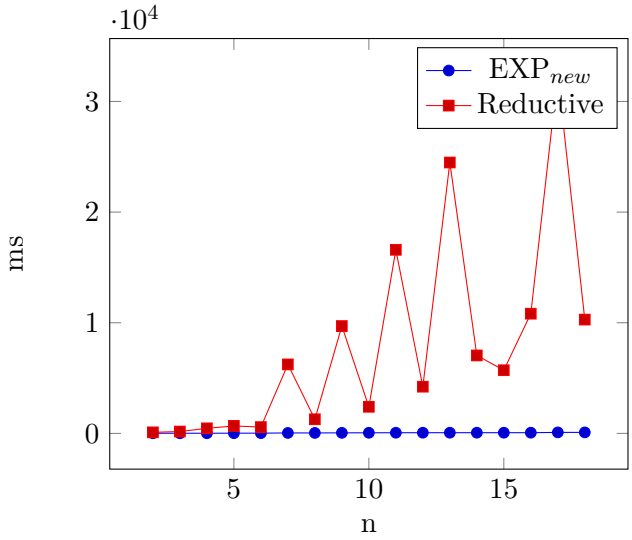

Figure 6.2: Comparison of the method EXP_{new} and the method Reductive for the proof CutIntroduction(SquareDiagonalExampleProof(n)) for $2 \leq n \leq 18$.

Note that also for SquareDiagonalExampleProof(n) the method EXP_{new} outperforms the method EXP, as can be seen in Figure 6.3.

We want to remark that the obtained results depend to a great degree on the operating system in use. Therefore, it is possible that the obtained results fluctuate. Nevertheless, the speed-up of the method EXP_{new} compared to EXP is given and can be clearly recognized.

6.2 Schematic Proof Construction

Gapt has been extended with a schematic formalism in order to perform a proof analysis with schematic CERES. The first step is to construct formalized schematic proofs and will be handled in this section.

First of all we like to note that the existing proof construction mechanisms of Gapt still rely on the syntactic constructions of Scala, i.e. Gapt does not have a dedicated input format. Nonetheless, Gapt contains a simple tactics language called gaptic for the construction of proofs providing a comfortable bottom-up development of proofs similar to the tactic languages found in proof assistants such as Coq and Isabelle.

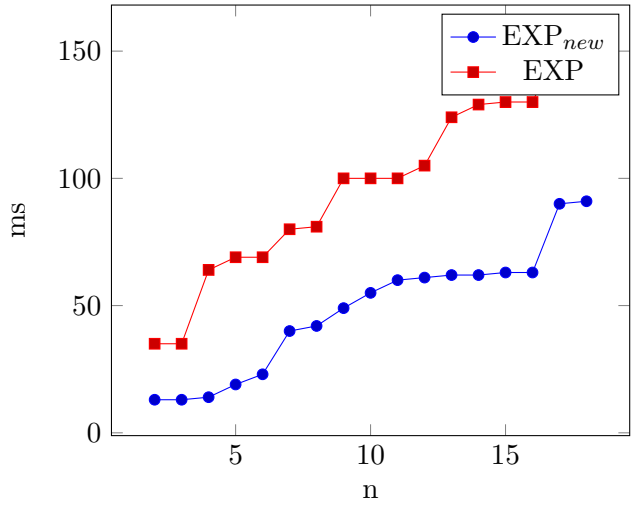

Figure 6.3: Comparison of the methods EXP and EXP$_{new}$ for the proof (containing cuts) CutIntroduction(SquareDiagonalExampleProof(n)) for $2 \leq n \leq 18$.

We will demonstrate how to formalize a simple example proof, which can be found in the examples directory of Gapt as FunctionIterationSchema.scala, using the Gapt system. More complex examples can also be found in this directory, as for instance the proof schema discussed in [CL17]. From the Scala interactive shell one can load these examples by importing the objects. For instance, the FunctionIterationSchema can be imported with the following command

```
scala> import examples.FunctionIterationSchema
```

Only after importing the objects we can access the proof schemata. For a general overview of gaptic we refer the reader to the Gapt User Manual.

The following proof schema is our canonical example for illustrating the schematic features of the Gapt system.

Example 6.2.1. We construct a proof schema

$$\mathscr{D}\colon \{(\delta, \rho(\delta, \vec{n}, 0), \rho(\delta, \vec{n}, n+1))\},$$

where the end-sequent is

$$S(\delta) = P(a), \forall x(P(x) \to P(f(x))) \vdash P(i\hat{f}(n))$$

using

$$\mathscr{E} = \begin{cases} i\hat{f}(0) = a \\ i\hat{f}(n+1) = f(i\hat{f}(n)). \end{cases}$$

$\rho(\delta, \vec{n}, 0)$ is an **LKE**-deduction of $S(\delta)\{n \leftarrow 0\}$:

$$\dfrac{\dfrac{P(a) \vdash P(a)}{P(a), \forall x(P(x) \to P(f(x))) \vdash P(a)}\, w : l}{P(a), \forall x(P(x) \to P(f(x))) \vdash P(\hat{if}(0))}\, \mathscr{E}$$

and $\rho(\delta, \vec{n}, n+1)$ is an **LKE**-deduction of $S(\delta)\{n \leftarrow n+1\}$:

$$\dfrac{\dfrac{\dfrac{(\nu_1)}{\forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(f(x)))} \qquad (\nu_2)}{P(0), \bigwedge_{i=0}^{n}(P(i) \to P(i+1)), P(n+1) \to P(n+2) \vdash P(n+2)}\, cut}{P(a), \forall x(P(x) \to P(f(x))), \forall x(P(x) \to P(f(x))) \vdash P(\hat{if}(n+1))}$$
$$\dfrac{}{P(a), \forall x(P(x) \to P(f(x))) \vdash P(\hat{if}(n+1))}\, c : l$$

where ν_1 is

$$\dfrac{\dfrac{\dfrac{\dfrac{P(\alpha) \vdash P(\alpha) \qquad P(f(\alpha)) \vdash P(f(\alpha)))}{P(\alpha) \to P(f(\alpha)), P(\alpha) \vdash P(f(\alpha)))}\, \to : l}{P(\alpha) \to P(f(\alpha)) \vdash P(\alpha) \to P(f(\alpha)))}\, \to : r}{\forall x(P(x) \to P(f(x))) \vdash P(\alpha) \to P(f(\alpha)))}\, \forall : l}{\forall x(P(x) \to P(f(x))) \vdash \forall x(P(x) \to P(f(x)))}\, \forall : r$$

and ν_2 is

$$\dfrac{\dfrac{\dfrac{S(\delta)\colon P(a), \forall x(P(x) \to P(f(x))) \vdash P(\hat{if}(n)) \qquad P(f(\hat{if}(n))) \vdash P(f(\hat{if}(n)))}{P(a), P(\hat{if}(n)) \to P(f(\hat{if}(n))), \forall x(P(x) \to P(f(x))) \vdash P(f(\hat{if}(n)))}\, \to : l}{P(a), P(\hat{if}(n)) \to P(f(\hat{if}(n))), \forall x(P(x) \to P(f(x))) \vdash P(\hat{if}(n))}\, \mathscr{E}}{P(a), \forall x(P(x) \to P(f(x))), \forall x(P(x) \to P(f(x))) \vdash P(\hat{if}(n+1))}\, \forall : l$$

First of all note that schematic proofs are dependent on several core packages of Gapt:

```
import gapt.expr._
import gapt.proofs.Context._
import gapt.proofs.gaptic._
import gapt.proofs.Context
import gapt.proofs.Sequent
```

These are the minimum imports needed for schematic proof construction. The *context* stores all information needed for the construction of a proof (as for instance the proof names, end-sequents of proofs, recursive predicates, logical sorts, and theory axioms). A schematic proof is an ordered set of tuples of proofs of the same end-sequent. The schematic extension of Gapt generalizes this concept to proof tuples with the same end-sequent, allowing the use of inductive definitions more complex than the natural

numbers. To enforce that base and step case proof share the same end-sequent the system allows for *proof name declarations* which each derivation in a proof schema for the same proof symbol must match. For the FunctionIterationSchema the following symbols and definitions have to be added to the context within the proof object:

```
object FunctionIterationSchema extends TacticsProof {

  ctx += Context.Sort( "i" )
  ctx += Context.InductiveType( "nat", hoc"0 : nat",
         hoc"s : nat>nat" )

  ctx += hoc"f:i>i"
  ctx += hoc"a:i"
  ctx += hoc"P: i>o"

  ctx += PrimRecFun( hoc"if:nat>i>i", "if 0 x = x", "if (s y)
         x = (f (if y x))" )

  ctx += hoc"phi: nat>nat"
  val esPhi = Sequent(Seq( hof"!x (-P(x) | P(f(x)))", hof"P(a)"),
                      Seq(hof"P(if(n,a))" ) )

  ctx += Context.ProofNameDeclaration( le"phi n", esPhi )
         . . .
  }
```

The variable ctx denotes the context. Its first extension is needed to handle the individual sort and the second extension to handle the numeric sort. Note that the distinction between the inductive sort Context.InductiveType("nat", hoc"0:nat", hoc"s:nat>nat") and the first-order term sort Context.Sort("i") must be made explicit, as well as the constants associated with the proof. f, a and P are simple symbol definitions used in FunctionIterationSchema. What follows is an extension of ctx with an inductive symbol definition using PrimRecFun. In fact, the definition of $\hat{\text{if}}(n+1, a)$ from Example 6.2.1 is added as a primitive recursive function PrimRecFun(hoc"if:nat>i>i", "if 0 x = x", "if (s y) x = (f (if y x))") to the context. We define a proof name phi with end-sequent esPhi, and add them using ProofNameDeclaration to our context.

Now we can construct the base and step case proof associated with the proof definition of phi:

```
  val esPhiSc = Sequent( Seq("Ant_1" -> hof"!x (-P(x) | P(f(x)))",
                "Ant_0" -> hof"P(a)" ), Seq( "Suc_0" -> hof"P(if(
```

```
                s(n),a))" ) )

  val phiSc = Lemma( esPhiSc ) {
    cut( "cut", hof"!x (-P(x) | P(f(x)))" )
    allR( "cut", fov"A" )
                . . .
    ref( "phi" )
    unfold( "if" ) atMost 1 in "Suc_0"
    trivial
  }

  ctx += Context.ProofDefinitionDeclaration( le"phi (s n)", phiSc )

  val esPhiBc = Sequent( Seq("Ant_1" -> hof"!x (-P(x) | P(f(x)))",
                "Ant_0" -> hof"P(a)" ), Seq( "Suc_0" -> hof"P(if(
                0,a))" ) )

  val phiBc = Lemma( esPhiBc ) {
    unfold( "if" ) atMost 1 in "Suc_0"
    trivial
  }

  ctx += Context.ProofDefinitionDeclaration(le"phi 0", phiBc)
}
```

Note that the proof name has a function type of ‘‘nat > nat’’. While it acts as a place holding constant, for proper integration into the system it has been implemented as a lambda term and thus, takes arguments and has a return type.

The base and step case proof associated with phi are referred to as phiSc and phiBc. We add these proofs to the context using ProofDefinitionDeclaration. Taking a close look at the above code, one will notice that phiBc is added to the context with a lambda expression le"phi 0" and phiSc with le"phi (s n)". These are the cases of the inductive definition handled by the proof. The cases do not need to match the inductive definition, moreover not all the cases have to be covered. An example of such a schema would be the NdiffSchema, which can be found the the examples directory of Gapt under schema/NdiffSchema.scala.

Notice the additional tactic added to the gaptic language, i.e. the reference tactic. The reference tactic ref is used to reference a proof and corresponds to adding end-sequents of previously defined derivations as initial sequents to new derivations. In the example code above, in phiSc it is used to reference to phi, i.e. we use ref("phi"). Note that using ref in a proof allows for referencing the same proof as well (self reference). The system checks the reference for validity by checking whether there exists a proof name matching the goal the reference is called on.

In our example we extensively use the unfold tactic. The unfold tactic, while not particular to schema, is used for unfolding the primitive recursive definitions such as $\hat{\text{if}}(n+1, a)$. Essentially, it implements the equational theory $\mathscr{E}$.

To display the base and the step case proof of FunctionIterationSchema, we have to access phiBc and phiSc. The sequence of commands

```
scala> import examples.FunctionIterationSchema
scala> val p = FunctionIterationSchema.phiBc
scala> prooftool( p )
```

first imports FunctionIterationSchema, stores the base case proof phiBc in p and outputs the derivation in prooftool, see Figure 6.4.

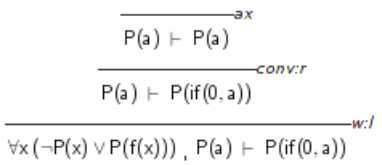

Figure 6.4: Prooftool output of the base case proof.

The following sequence of commands is needed to output the step case proof in prooftool.

```
scala> import examples.FunctionIterationSchema
scala> val p = FunctionIterationSchema.phiSc
scala> prooftool( p )
```

Of course, FunctionIterationSchema does not have to be imported again if we have already imported it for displaying the base case proof. Figure 6.5 illustrates the output of prooftool.

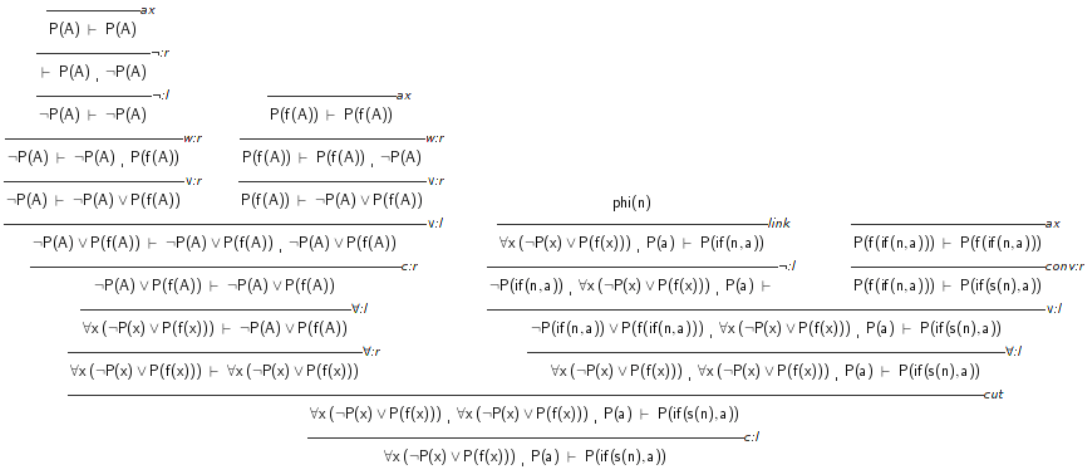

Figure 6.5: Prooftool output of the step case proof.

Also, we can instantiate the proof schema FunctionIterationSchema with an arbitrary value. The proof schema can be instantiated with the value 2 with the commands

```
scala> import examples.FunctionIterationSchema
scala> import examples.FunctionIterationSchema.ctx
scala> val p = instantiateProof.Instantiate( le"phi (s (s 0))" )
scala> prooftool( p )
```

It is important to import the context ctx of FunctionIterationSchema as well!

The output in prooftool is hardly readable and this problem becomes even worse for bigger proofs. The solution to this problem is to output the proof in a different representation. Indeed, with prooftool we can display proofs not only in a tree-like structure (i.e. as usual sequent calculus proofs) but also in a so-called sunburst view (accessible via the *View* menu in prooftool), see Figure 6.6. The sunburst view was introduced to obtain a means of displaying very large proofs. It can be interpreted as a structure which can be unrolled to a proof in tree-like structure. Indeed, the point in the middle of the sunburst corresponds to the end-sequent of a proof in tree-like structure. Cut rules are displayed in green, while structural rules and axioms are displayed in gray. The orange parts correspond to unary logical rules, the yellow ones to binary logical rules, strong quantifier rules are displayed in red and weak quantifier rules in blue.

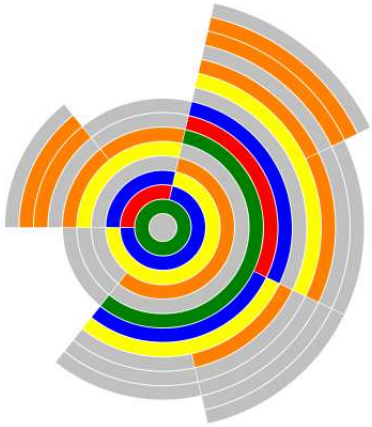

Figure 6.6: Prooftool output in sunburst view of FunctionIterationSchema for parameter 2.

6.3 Experiments with Schematic CERES

Instantiated proof schemata are essentially **LK**-proofs extended with an equational theory and thus, any of the proof analytic tools and methods of Gapt can be applied. However, there are also tools specifically designed for uninstantiated proof schemata, especially those which contain cuts. In this section we will provide an introduction to the schematic proof analysis capabilities of Gapt. First, we will explain in detail how some of the features are implemented in Gapt using the FunctionIterationSchema. Then we will analyse a more complicated example.

6.3.1 The Function Iteration Schema

In the previous section we provided a short introduction to proof schema construction in Gapt using FunctionIterationSchema. In this section we will demonstrate an analysis of this proof schema.

First of all note that we can analyse any instantiation of FunctionIterationSchema. For instance, we can construct the characteristic clause set of an instance of a proof schema from an intermediate representation refereed to as the *struct* of the proof. In the code below we extract the characteristic clause set of instance 3.

```
scala> import examples.FunctionIterationSchema
scala> import examples.FunctionIterationSchema.ctx
scala> val p = instantiateProof.Instantiate( le"phi (s (s (s 0)
                ))" )
scala> val struct = StructCreators.extract( p )
scala> val cs = CharacteristicClauseSet( struct )
```

The output of prooftool is illustrated in Figure 6.7.

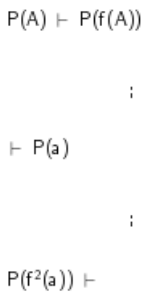

Figure 6.7: Characteristic clause set of instance 3 of the FunctionIteration schema.

However, we can also analyse schematic proofs and one of the most interesting benefits of the interactive features which have been added to Gapt is the representation of the cut-structure of a proof schema as an inductive definition. This feature corresponds to the construction of a schematic characteristic formula, see Definition 5.1.2. In fact, the characteristic formula schema as well as the schematic characteristic clause set can be constructed from the struct of the proof. The commands

```
scala> val SCS = SchematicStruct( "phi" ).getOrElse( Map() )
```

stores the SchematicStruct in SCS. Given its type it is not possible to display it in prooftool, however in the Scala interactive shell one can check that it corresponds to a schematic characteristic formula. With the code

```
val SCS: Map[CLS, ( Struct, Set[Var] )] = SchematicStruct(
        "phi" ).getOrElse( Map() )

val CFPRP = CharFormPRP( SCS )

CharFormPRP.PR( CFPRP )
```

we store the schematic characteristic formula in CFPRP and construct the primitive recursive definitions of the characteristic formula with CharFormPRP.PR(CFPRP).

To construct a schematic refutation of the schematic characteristic formula, we have to construct a proof schema which proves the negation of the schematic characteristic formula. The schematic refutation of FunctionIterationSchema can be found in the directory /gapt/examples/schema/FunctionIteration Refutation.

```
object FunctionIterationRefutation extends TacticsProof(
       FunctionIterationSchema.ctx ) {

  val SCS: Map[CLS, ( Struct, Set[Var] )] = SchematicStruct(
          "phi" ).getOrElse( Map() )

  val CFPRN = CharFormPRN( SCS )

  CharFormPRN.PR( CFPRN )
        .  .  .
}
```

In the code above, we first import the context `ctx` from `FunctionIterationSchema`. Then the schematic struct is produced from the context, which is referred to as SCS. The negation of the schematic characteristic formula is stored in CFPRN and the primitive recursive definition of the negation of the characteristic formula is constructed with CharFormPRN.PR(CFPRN). Other than these few extra commands at the beginning of the object file, the construction of the refutation of the schematic characteristic formula is quite similar to the construction of a proof schema. Essentially, $F \vdash$, where F is the schematic characteristic formula, is proved. Once the refutation schema from the characteristic formula is constructed an expansion proof or a Herbrand sequent for any instance can be produced. In Figure 6.8 we illustrate the expansion proof of `FunctionIterationRefutation` for parameter 7, which can be obtained with the sequence of commands below (note that just like the sunburst view the expansion proof is accessible via the *View* menu in prooftool).

```
scala> import examples.FunctionIterationRefutation
scala> import examples.FunctionIterationRefutation.ctx
scala> val proof = instantiateProof.Instantiate( le"Top (s (s (s
                    (s (s (s (s 0)))))))) " )
scala> prooftool( proof )
```

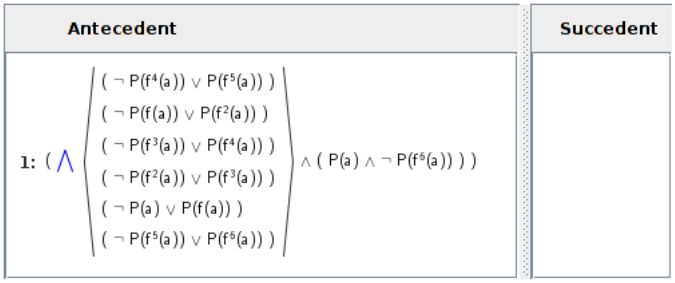

Figure 6.8: Expansion proof of instance 7 of the FunctionIterationRefutation schema.

6.3.2 The Eventually Constant Schema

Another interesting example is the analysis of the so-called Eventually Constant Schema (ECS for short). In [CL16] the ECS was analysed using the method of [DLRW13] by manually producing instances of the proof and its cut-structure using various theorem provers. This interactive process has been streamlined using Gapt and schemata of both the proof (EventuallyConstantSchema) and the refutation of the characteristic formula (EventuallyConstantRefutation) can be found in gapt/examples/schema/Schema. The ECS is a formal proof of the following statement.

Proposition 6.3.1. *Given a total monotonically decreasing function $f : \mathbb{N} \to \{0, \cdots, n\}$, for $n \in \mathbb{N}$, there exists an $x \in \mathbb{N}$ such that for all $y \in \mathbb{N}$, where $x \leq y$, it is the case that $f(x) = f(y)$.*

Proof. In case the range contains only 0 the theorem trivially holds. Let us assume as induction hypothesis that for a co-domain with n elements the proposition holds and show that it holds for a co-domain with $n + 1$ elements. If for all positions x, $f(x) = n$ then the theorem holds, else if at some y, $f(y) \neq n$ then from that point on f cannot map to n, as the function is monotonically decreasing. Therefore, f will only have n elements in its co-domain and by the induction hypothesis the proposition holds. □

The following sequence of commands outputs the instantiation of the proof schema EventuallyConstantSchema with 2, see Figure 6.9. In fact, displaying the proof in sunburst view is essential, as the proof would be to big to be illustrated here.

```
scala> import examples.EventuallyConstantSchema
scala> import examples.EventuallyConstantSchema.ctx
scala> val p = instantiateProof.Instantiate( le"omega (s
                (s 0))" )
scala> prooftool( p )
```

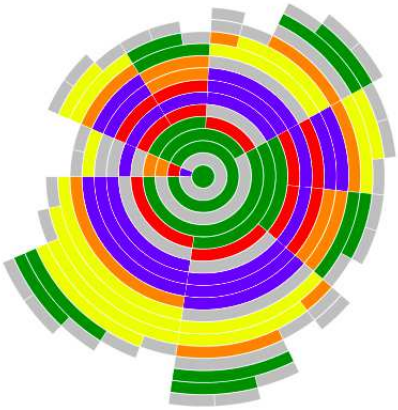

Figure 6.9: Sunburst view of the EventuallyConstantSchema instantiated with 2. Cuts are displayed in green.

The cut formula corresponds to the case distinction made in the step case of our mathematical proof. In the **LKE**-calculus it has the form:

$$\exists x \forall y \left(((x \leq y) \rightarrow n+1 = f(y)) \vee f(y) < n+1 \right).$$

The formal statement of the end-sequent in the **LKE**-calculus has a $\exists\forall$ quantifier prefix like the cut formula:

$$\forall x (\bigvee_{i=0}^{n+1} i = f(x)), \forall x \forall y \Big(x \leq y \rightarrow f(y) \leq f(x) \Big) \vdash \exists x \forall y (x \leq y \rightarrow f(x) = f(y)).$$

Both the CERES method and schematic CERES method were designed for proofs without strong quantifiers in the end-sequent. Therefore the proofs have to be skolemized, introducing the Skolem function $g(\cdot)$ in the formal proof entered in the Gapt system.

The schematic characteristic formula of the ECS is very complex and therefore we present it here as a recursively defined clause set. More details can be found in [CL16]. $C4(x, y, i, k)$ and $C4'(x, y, i, k)$ formalize the case distinction occurring in the cut-formula.

$$\begin{array}{lcl}
C1(x,k) & \equiv & \vdash x(k) \leq x(k) \\
C2(x,k) & \equiv & \vdash x(k) \leq g(x(k)) \\
C3(x,i,k) & \equiv & i = f(x(k)), i = f(g(x(k))) \vdash \\
C4(x,y,i,k) & \equiv & y(k) \leq x(k), f(y(k)) < i+1 \vdash \\
 & & f(x(k)) < i, i = f(x(k)) \\
C4'(x,y,i,k) & \equiv & y(k) \leq x(k+1), f(y(k)) < i+1 \vdash \\
 & & f(x(k+1)) < i, i = f(x(k+1)) \\
C5(x,k) & \equiv & f(x(k)) < 0 \vdash \\
C6(x,k) & \equiv & f(g(x(k))) < 0 \vdash \\
C7(x,k) & \equiv & 0 \leq x(k) \vdash f(x(k)) < n, f(x(k)) = n
\end{array}$$

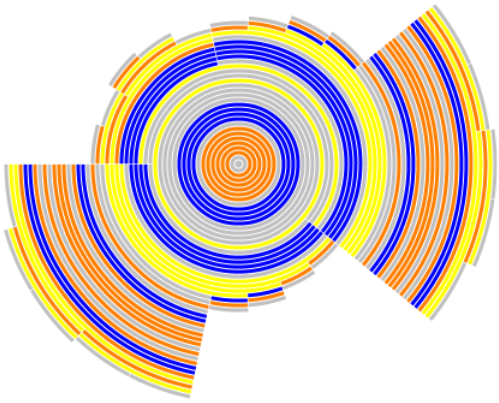

Figure 6.10: Sunburst view of the EventuallyConstantSchemaRefutation instantiated with 1.

It is interesting to note that while the proof schema grows linearly, the refutation schema grows exponentially. Below, the refutation schemata is instantiated with 1.

```
scala> import examples.EventuallyConstantSchemaRefutation
scala> import examples.EventuallyConstantSchemaRefutation.ctx
scala> val proof1 = instantiateProof.Instantiate( le"Top (s 0) " )
scala> prooftool( proof1 )
```

The output is illustrated in Figure 6.10.

In Figure 6.11 we illustrate the EventuallyConstantRefutation instantiated with 3, in Figure 6.12 instantiated with 5 and in Figure 6.13 instantiated with 7. The instantiated schema in Figure 6.13 has 128 branches.

As for the FunctionIterationRefutation, we can construct an expansion proof for the EventuallyConstantSchemaRefutation. Note that it provides similar information as the obtained substitutions discussed in [CL16]. Figure 6.14 illustrates a partial representation of the expansion proof of EventuallyConstantSchemaRefutation instantiated with 7 (as the expansion proof is too large for being displayed, we present only the most significant part).

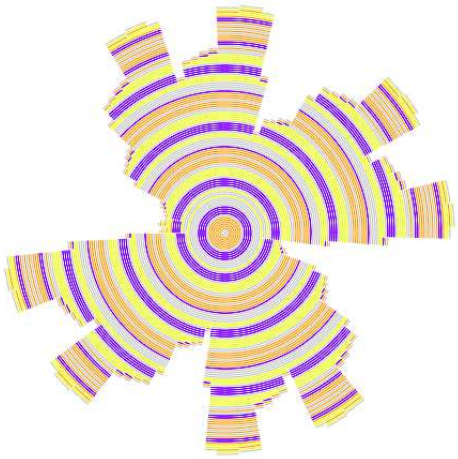

Figure 6.11: Sunburst view of the EventuallyConstantSchemaRefutation instantiated with 4.

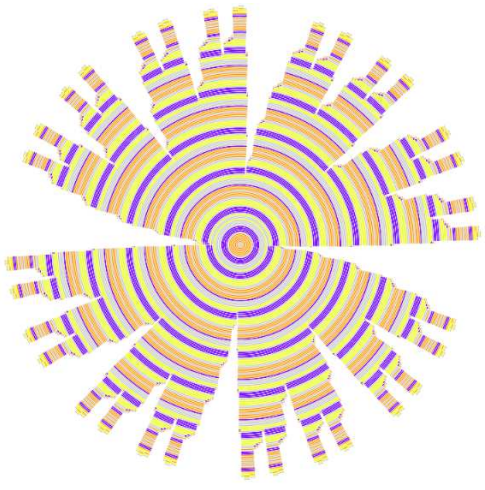

Figure 6.12: Sunburst view of the EventuallyConstantSchemaRefutation instantiated with 4.

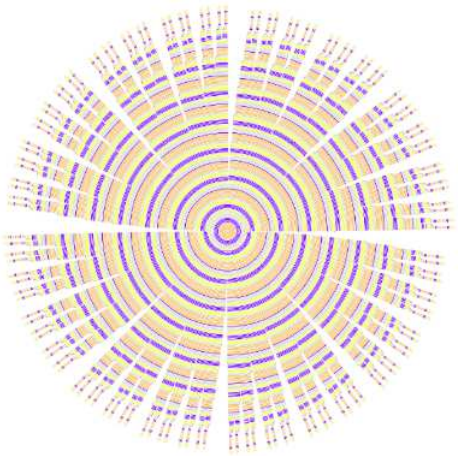

Figure 6.13: Sunburst view of the EventuallyConstantSchemaRefutation instantiated with 7.

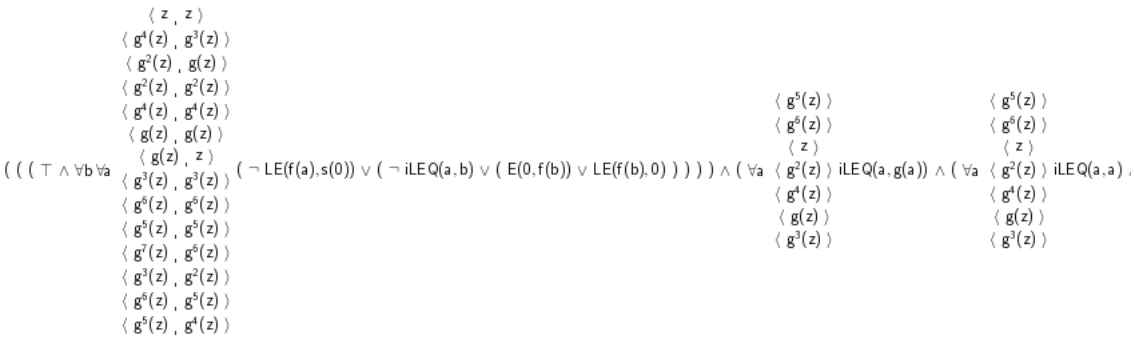

Figure 6.14: Part of the expansion proof of the EventuallyConstantSchemaRefutation instance 7.

CHAPTER 7

Future Work

In this chapter we will give an outlook on what will be investigated in future years.

The main goal of this investigation is to develop a fully automated analysis of mathematical proofs, as for instance Fürstenberg's proof of the infinitude of primes. As already described, a semi-automated proof analysis using schemata was performed on Fürstenberg's proof and major parts of the analysis had to be performed by hand [F55, BHL+08]. The analysis showed that from Fürstenberg's proof Euclid's elementary proof could be obtained by applying a formal cut-elimination procedure. Though a fully automated analysis of this proof is not yet within reach, this work constitutes a major step in this direction. Some missing artefacts will be investigated in near future, as for instance the full implementation of the developed proof analysis methods. The automatic construction of a projection schema from an input proof schema is currently under development in Gapt. This implementation will then be used to automatically extract schematic Herbrand sequents and schematic expansion proofs from projection schemata in Gapt. Moreover, the construction of a simple refutation schema (see Section 4.4) in Gapt will be further investigated. In particular, the current representation of a simple refutation schema in Gapt is as a proof schema, following the rules of **LK**. Given the structure of and the possibilities in Gapt, it is not yet possible to define a calculus with introduction and elimination rules as needed for the RPL_0^{Ψ} calculus. Having all these algorithms developed, we plan to implement the proof analysis method as has been already done for the first-order case (as described in Section 6.1).

As already noted in Section 5.6, in a schematic setting we have not yet included equality as paramodulation rules that operate on any specific occurrence in an expression. Handling this kind of paramodulation is a non-trivial task and requires further investigation. Of course, having a proof analysis method that covers inductive proofs that contain equality rules is needed to perform a fully automated analysis of mathematical proofs and therefore we plan to investigate this topic in future work.

It is also possible to develop a more general definition of proof and refutation schemata, following the call graph formalism introduced in [CLL19]. There, call graphs are used to define the semantics of schematic derivations that allow mutual recursion.

It should also be noted that the methods for proof analysis as developed in this work can be easily extended to extract information different to Herbrand sequents from formalized proofs. For instance, it is possible to extract interpolants from CERES normal forms. In the same way as the Herbrand sequent of a CERES normal form can already be extracted from the proof projection and the refutation of the characteristic formula (see Section 3.3), it is also possible to extract the interpolant of the CERES normal form already from the proof projection and the refutation of the characteristic formula. This investigation would result in a fast computation of interpolants based on CERES but without actually constructing a proof in normal form.

Moreover, the methodology developed in this work can also be used for other CERES methods. Indeed, in the first-order case we can exchange the calculus **LK** with an arbitrary calculus that is conservative over the extraction of Herbrand sequents. For instance, we could consider the globally sound but possibly locally unsound calculi $\mathbf{LK}^{+}$ and $\mathbf{LK}^{++}$ introduced in [AB19] and investigate a CERES(-like) method that constructs Herbrand sequents from the proof projections and the resolution refutation of the characteristic formula. Then, we can investigate a similar method for the calculus $\mathbf{LQ}^{++}$ (a calculus for quantifier macros) introduced in [BL19]. Note that $\mathbf{LQ}^{++}$ can be seen as a first step in defining a calculus for Henkin quantifiers, hence it paves the way for an efficient method extracting Herbrand sequents and expansion proofs for a logic with Henkin quantifiers.

List of Figures

List of Tables

Index

Bibliography

[AB19] Juan P. Aguilera and Matthias Baaz. Unsound inferences make proofs shorter. *J. Symb. Log.*, 84(1):102–122, 2019.

[And71] Peter B. Andrews. Resolution in type theory. *J. Symb. Log.*, 36(3):414–432, 1971.

[BBLS06] Ulrich Berger, Stefan Berghofer, Pierre Letouzey, and Helmut Schwichtenberg. Program extraction from normalization proofs. *Studia Logica*, 82(1):25–49, 2006.

[BBS02] Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114(1):3–25, 2002.

[BCF08] Matthias Baaz, Agata Ciabattoni, and Christian G Fermüller. Cut elimination for first order Gödel logic by hyperclause resolution. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 451–466. Springer, 2008.

[BH11] Matthias Baaz and Stefan Hetzl. On the non-confluence of cut-elimination. *J. Symb. Log.*, 76(1):313–340, 2011.

[BHL+05] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Cut-elimination: experiments with ceres. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 481–495. Springer, 2005.

[BHL+06] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Proof transformation by CERES. In *Mathematical Knowledge Management, 5th International Conference, MKM 2006, Wokingham, UK, August 11-12, 2006, Proceedings*, pages 82–93, 2006.

[BHL+08] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Ceres: An analysis of Fürstenberg's proof of the infinity of primes. *Theoretical Computer Science*, 403(2-3):160–175, 2008.

[BL94] Matthias Baaz and Alexander Leitsch. On skolemization and proof complexity. *Fundamenta Informaticae*, 20(4):353–379, 1994.

[BL00] Matthias Baaz and Alexander Leitsch. Cut-elimination and redundancy-elimination by resolution. *Journal of Symbolic Computation*, 29(2):149–177, 2000.

[BL06] Matthias Baaz and Alexander Leitsch. Towards a clausal analysis of cut-elimination. *Journal of Symbolic Computation*, 41(3):381–410, 2006.

[BL11] Matthias Baaz and Alexander Leitsch. *Methods of Cut-elimination*, volume 34. Springer Science & Business Media, 2011.

[BL19] Matthias Baaz and Anela Lolic. Note on globally sound analytic calculi for quantifier macros. In *WoLLIC*, volume 11541 of *Lecture Notes in Computer Science*, pages 486–497. Springer, 2019.

[BLL18] Matthias Baaz, Alexander Leitsch, and Anela Lolic. A sequent-calculus based formulation of the extended first epsilon theorem. In *LFCS*, volume 10703 of *Lecture Notes in Computer Science*, pages 55–71. Springer, 2018.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[BS11] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

[Bus94] Samuel R. Buss. On herbrand's theorem. In Daniel Leivant, editor, *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 1994.

[CL16] David M. Cerna and Alexander Leitsch. Schematic cut elimination and the ordered pigeonhole principle. In *IJCAR*, volume 9706 of *Lecture Notes in Computer Science*, pages 241–256. Springer, 2016.

[CL17] David M. Cerna and Michael Peter Lettmann. Integrating a global induction mechanism into a sequent calculus. In *TABLEAUX*, volume 10501 of *Lecture Notes in Computer Science*, pages 278–294. Springer, 2017.

[CLL19] David Cerna, Alexander Leitsch, and Anela Lolic. Schematic refutations of formula schemata. *CoRR*, abs/1902.08055, 2019.

[CLRW17] David Cerna, Alexander Leitsch, Giselle Reis, and Simon Wolfsteiner. Ceres in intuitionistic logic. *Annals of Pure and Applied Logic*, 168:1783–1836, 2017.

[DLL+13] Cvetan Dunchev, Alexander Leitsch, Tomer Libal, Martin Riener, Mikheil Rukhaia, Daniel Weller, and Bruno Woltzenlogel-Paleo. Prooftool: a gui for the gapt framework. *arXiv preprint arXiv:1307.1942*, 2013.

[DLRW13] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Cut-elimination and proof schemata. In *International Tbilisi Symposium on Logic, Language, and Computation*, pages 117–136. Springer, 2013.

[EHR+16] Gabriel Ebner, Stefan Hetzl, Giselle Reis, Martin Riener, Simon Wolfsteiner, and Sebastian Zivota. System description: Gapt 2.0. In *International Joint Conference on Automated Reasoning*, pages 293–301. Springer, 2016.

[F55̈] Hillel Fürstenberg. On the infinitude of the primes. *American Mathematical Monthly*, (62):353, 1955.

[FW78] Hillel Fürstenberg and Benji Weiss. Topological dynamics and combinatorial number theory. *Journal d'Analyse Mathematique*, pages 34(1):61–85, 1978.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1934-35.

[Gir87] Jean Yves Girard. *Proof Theory and Logical Complexity*, volume 1. Biblopolis, Napoli,1987.

[Göd58] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3-4):280–287, 1958.

[Gon08] Georges Gonthier. Formal proof–the four-color theorem. *Notices of the AMS*, 55(11):1382–1393, 2008.

[HB39] David Hilbert and Paul Bernays. Grundlagen der Mathematik. 2, 1939.

[Her30] Jacques Herbrand. *Recherches sur la théorie de la démonstration.* PhD thesis, Université de Paris, 1930.

[HLRR13] Stefan Hetzl, Tomer Libal, Martin Riener, and Mikheil Rukhaia. Understanding resolution proofs through herbrand's theorem. In Didier Galmiche and Dominique Larchey-Wendling, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 22th International Conference, TABLEAUX 2013, Nancy, France, September 16-19, 2013. Proceedings*, volume 8123 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2013.

[HLW11] Stefan Hetzl, Alexander Leitsch, and Daniel Weller. Ceres in higher-order logic. *Annals of Pure and Applied Logic*, 162(12):1001–1034, 2011.

[HLWP08] Stefan Hetzl, Alexander Leitsch, Daniel Weller, and Bruno Woltzenlogel Paleo. Herbrand sequent extraction. In Serge Autexier, John A. Campbell, Julio Rubio, Volker Sorge, Masakazu Suzuki, and Freek Wiedijk, editors,

Intelligent Computer Mathematics, 9th International Conference, AISC 2008, 15th Symposium, Calculemus 2008, 7th International Conference, MKM 2008, Birmingham, UK, July 28 - August 1, 2008. Proceedings, volume 5144 of *Lecture Notes in Computer Science*, pages 462–477. Springer, 2008.

[HW13] Stefan Hetzl and Daniel Weller. Expansion trees with cut. *arXiv preprint arXiv:1308.0428*, 2013.

[LL19] Alexander Leitsch and Anela Lolic. Extraction of expansion trees. *J. Autom. Reasoning*, 62(3):393–430, 2019.

[LPW17] Alexander Leitsch, Nicolas Peltier, and Daniel Weller. CERES for first-order schemata. *J. Log. Comput.*, 27(7):1897–1954, 2017.

[Mil87] Dale A Miller. A compact representation of proofs. *Studia Logica*, 46(4):347–370, 1987.

[MM00] Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232(1-2):91–119, 2000.

[MZ06] Georg Moser and Richard Zach. The epsilon calculus and Herbrand complexity. *Studia Logica*, 82(1):133–155, 2006.

[Pfe84] Frank Pfenning. Analytic and non-analytic proofs. In *7th International Conference on Automated Deduction*, pages 394–413. Springer, 1984.

[Rei14] Giselle Reis. *Cut-elimination by resolution in intuitionistic logic*. PhD thesis, Technische Universität Wien, 2014.

[Rei15] Giselle Reis. Importing smt and connection proofs as expansion trees. *arXiv preprint arXiv:1507.08715*, 2015.

[Tak87] Gaisi Takeuti. *Proof Theory*. North Holland, second edition, 1987.

[Urb00] Christian Urban. *Classical logic and computation*. PhD thesis, University of Cambridge, 2000.

[VdW27] Bartel L. Van der Waerden. Beweis einer Baudetschen Vermutung. *Nieuw Archiv Wiskunde*, pages 15(2):212–216, 1927.