FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

CERES in Proof Schemata

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der technischen Wissenschaften

eingereicht von

Mikheil Rukhaia
Matrikelnummer 0827684

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ.Prof. Dr.phil. Alexander Leitsch

Diese Dissertation haben begutachtet:

(Univ.Prof. Dr.phil. Alexander Leitsch) (Dr. Nicolas Peltier)

Wien, 15.11.2012

(Mikheil Rukhaia)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

CERES in Proof Schemata

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der technischen Wissenschaften

by

Mikheil Rukhaia
Registration Number 0827684

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ.Prof. Dr.phil. Alexander Leitsch

The dissertation has been reviewed by:

(Univ.Prof. Dr.phil. Alexander Leitsch) (Dr. Nicolas Peltier)

Wien, 15.11.2012

(Mikheil Rukhaia)

To my love(ly) Kate.

Erklärung zur Verfassung der Arbeit

Mikheil Rukhaia
Favoritenstrasse 9-11, 1040 Wien

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.

(Ort, Datum) (Unterschrift Verfasser)

Contents

Kurzfassung

Die Methode der Schnittelimination nimmt als Kern von Gerhard Gentzens Hauptsatz eine wichtige Rolle im Bereich der Logik ein. Verwendet ein Beweis jedoch das Induktionsprinzip, ist Schnittelimination im Allgemeinen nicht mehr möglich. Ein Ansatz ist, Schnittelimination für eine abzählbar unendliche Folge von Beweisen, ein sogenanntes Beweisschema, zu definieren. Da das Resultat nun ebenfalls ein Beweisschema ist, lassen sich die auf reduktiven Ansätzen basierenden Methoden nicht mehr anwenden. Das dabei auftretende Problem ist, dass eine Schnittregel nicht immer schrittweise in Richtung der Axiome verschoben werden kann. Daher muss auf eine andere Schnitteliminationstechnik zurückgegriffen werden.

Die Schnitteliminationsmethode CERES basiert auf dem Konzept der charakteristischen Klauselmenge. Diese wird aus einem Beweis im Sequenzkalkül extrahiert und ist immer unerfüllbar. Eine Resolutionswiderlegung dieser Klauselmenge wird als Vorlage für einen neuen Beweis herangezogen, der nur mehr Schnitte über Atomformeln enthält. Dazu wird jede Klausel durch eine Projektion des ursprünglichen Beweises ersetzt und jeder Resolutionsschritt als Schnitt über einer Atomformel nachvollzogen. Damit analysiert CERES im Gegensatz zu den reduktiven Methoden alle Schnittformeln auf einmal.

Diese Dissertation verallgemeinert CERES sowohl für Aussagenlogik als auch für Prädikatenlogik erster Stufe auf Beweisschemata. Dazu wird der Sequenzkalkül $\mathbf{LK}_s$ eingeführt, der die Darstellung von Beweisschemata mittels primitiver Rekursion zulässt. Es wird gezeigt, wie sich Schemata für die charakteristische Klauselmenge und die Beweisprojektionen erstellen lassen. Analog wird auch ein schematischer Resolutionskalkül definiert, der die Widerlegung der charakteristischen Klauselmenge erlaubt. Diese Widerlegung wird nun zusammen mit den Projektionen als Vorlage zur Konstruktion eines Beweisschemas herangezogen, das ausschließlich atomare Schnitte verwendet. Daraus ergibt sich eine Methode zur Schnittelimination für Beweisschemata.

Abstract

Gentzen's cut-elimination theorem is one of the most important theorems of logic. But it is proved that, in the presence of induction, cut-elimination is not possible in general. One way to overcome this problem is to define an infinite sequence of proofs in a uniform way (i.e. a proof schema), and do cut-elimination for the proof schema. This makes sense if the cut-free proofs also can be described uniformly. But the reductive methods fail to do so. The reason is that shifting cuts is not always possible in such proof schemata. Therefore another cut-elimination procedure is needed.

The cut-elimination method CERES (for first- and higher-order classical logic) is based on the notion of a characteristic clause set, which is extracted from an **LK**-proof and is always unsatisfiable. A resolution refutation of this clause set can be used as a skeleton for a proof with atomic cuts only (atomic cut normal form). This is achieved by replacing the input clauses from the resolution refutation by corresponding projections of the original proof. So, in contrast to reductive methods, CERES analyzes the proof as a whole.

We present a generalization of the CERES method to propositional and first-order proof schemata. We define a schematic version of the sequent calculus called $\mathbf{LK}_s$, and a notion of proof schema based on primitive recursive definitions. A method is developed to extract schematic characteristic clause sets and schematic projections from these proof schemata. We also define a schematic resolution calculus for refutation of schemata of clause sets, which can be applied to refute the schematic characteristic clause sets. Finally the projection schemata and resolution schemata are plugged together and a schematic representation of the atomic cut normal forms is obtained. Hence, we obtain a cut-elimination procedure for proof schemata.

Acknowledgements

I would like to thank my supervisor, Prof. Alexander Leitsch. His expansive knowledge and thoughtful advice though the many revisions of my thesis helped me complete this body of work.

I would also like to thank Nicolas Peltier, who accepted the role of reviewer for my thesis. His comments and suggestions brought the clarity needed for the greater scientific community to benefit from my work.

I would like to give thanks to all professors and teachers in Vienna University of Technology, especially Prof. Matthias Baaz, who's lectures gave the knowledge of Proof Theory required to make a contribution to the subject.

I would like to thank all members of Theory and Logic group, especially Cvetan Dunchev and Daniel Weller, for new ideas and our many discussions. This thesis could not have been finished without them. Special thanks goes to David Cerna for correcting some of my English mistakes and Martin Riener for writing Kurzfassung for me.

I would also like to thank all the people I have met during my study in Vienna. Without them my life would have been a lot less interesting.

Last, but not least, I would like to thank my family and friends, for their patience and encouragement.

Chapter 1

Introduction

Gentzen's cut-elimination theorem is one of the most important theorems of logic. Removing cuts from formal proofs corresponds to the elimination of intermediary lemmas from mathematical proofs. In cut-free proofs all statements are subformulas of the end-sequent. Cut-free proofs are analytic in sense that they allow extraction of the "hidden" knowledge, such as Herbrand disjunctions and interpolants. Systems, that have a cut-elimination theorem, are easily proved to be consistent.

Gentzen's original proof of his cut-elimination theorem is constructive. He gave a reductive cut-elimination method, which can be interpreted as a proof rewriting system, which is terminating, and normal forms of this system are cut-free proofs. Such kind of systems usually are nondeterministic, i.e. no explicit algorithm is specified which selects the reduction rule to be applied next. Another reductive method was given also by Tait and Schütte. These two methods differ in the selection of the cut that has to be eliminated first. These methods analyze only small parts of proofs (the derivation of the uppermost logical operator of a cut-formula) and leave other parts of the proof unchanged, which may lead to major redundancy.

A cut-elimination method, called CERES, was developed by Matthias Baaz and Alexander Leitsch in [BL00], which radically differs from reductive cut-elimination methods. CERES is a cut-elimination method by resolution, reducing cut-elimination to a theorem proving problem. In this method not only the derivation of the uppermost logical operator of a cut-formula is analyzed, but all derivations of cut-formulas at once. This analysis leads to the construction of an unsatisfiable set of clauses, called characteristic clause set. Then a resolution refutation of the characteristic clause set is used as a skeleton of a proof with atomic cuts only. The method was originally developed for first-order logic, but later it was extended to second- and higher-order logics (see [HLWP09, HLW11]). The CERES method was imple-

mented and real mathematical proofs were analyzed using the CERES-system (see [BHL$^+$05, BHL$^+$08, DLL$^+$10]).

Another very important tool in computer science and mathematics in general is induction. But it adds complexity to proofs, because the rule corresponds to an infinitary modus ponens rule. Also, on the syntactic side, while logical rules change the complexity of formulas, the induction rule, although it may change formulas, not necessarily changes the complexity of formulas. Therefore, reducing the complexity of formulas cannot be used to show the termination of cut-elimination in the presence of induction.

Gentzen's procedure, and reductive cut-elimination methods in general, shift cuts upwards until they are eliminated. When an induction rule occurs in the proof, such kind of shift over it is not always possible, therefore cut-elimination is not possible in general in the presence of induction.

This issue first was investigated in [Tak87]. From the Gentzen's proof of consistency of Peano arithmetic it is shown that cut-elimination is possible when induction is grounded; in this case the induction term can be evaluated to numeral and can be replaced by a finite number of cuts.

Later, in [MM00] a reductive cut-elimination method was given for intuitionistic proof systems with induction. In [Lib08] a cut-elimination theorem for subclass of inductive proofs of weakly quantified theorems was proved.

In [Bro05] the author presented a so-called cyclic proof system and showed that such kind of systems subsume the use of the induction rule. He also mentioned the problem that there is no proof of the cut-elimination theorem for classical cyclic proof systems in the literature. The point is that reductive cut-elimination methods cannot be extended to proofs with cycles (shifting cuts over cycles is a major problem).

Schemata are widely used in mathematics on a meta-level as an alternative to induction. Since cut-elimination in the presence of induction is so problematic, there was an attempt in [BHL$^+$08] to use schemata in practice, instead of induction, to analyze interesting mathematical proofs using the CERES method. In this paper the CERES-system was applied to (a formalization of) a mathematical proof: Fürstenberg's proof of the infinity of primes [AZ99]. The proof was formalized as a sequence of proofs $(\pi_k)_{k\in\mathbb{N}}$ showing that the assumption that there exist exactly k primes is contradictory. The analysis was performed in a semi-automated way: $\mathrm{CL}(\pi_k)$ (the characteristic clause set of π_k) was computed for some small values of k and from this, a general scheme $\mathrm{CL}(\pi_n)$ was constructed and refuted by hand. Even the analysis of this proof was very interesting: from Fürstenberg's proof, which makes use of topological concepts, Euclid's elementary proof was obtained by cut-elimination. However, it was unsatisfactory that the fact, that $\mathrm{CL}(\pi_n)$ is really the correct schema for all $n \in \mathbb{N}$, could not be verified, and

that the analysis of $\mathrm{CL}(\pi_n)$ could not be performed in a computer-aided way.

The aim of this thesis is to define a language for treating schemata on the object level and to present a cut-elimination method *CERES*$_s$ for it. In such a way we avoid the explicit use of induction and get strong tool to represent and reason on proofs with some kind of cycles. The method *CERES*$_s$ overcomes the shortcoming of reductive cut-elimination methods on such proofs by analyzing the proof as a whole and eliminating all cuts together. Hence we obtain a cut-elimination method for cyclic proof systems. It will be shown that there is a translation from a proof with an ordinary induction rule into our system, and since cut-elimination is possible in our settings, hence showing that our system has an advantage over usual systems with induction rule.

To achieve this aim, we define a schematic version of sequent calculus, called $\mathbf{LK}_s$. A schematic proof is a tuple of pairs of $\mathbf{LK}_s$-proofs (corresponding to the base and step cases of an inductive definition). The *CERES*$_s$ method is based on the notion of a schematic characteristic clause set, which is extracted from a schematic proof and is always unsatisfiable. This will close the gap in the application described above (and in future ones) by automatically computing the correct schema $\mathrm{CL}(\pi_n)$. We use a resolution refutation of this clause set as a skeleton for a proof schema with atomic cuts only. This is achieved by replacing clauses from the resolution refutation by the corresponding projection schemata of the original proof schema. We show that there is a correspondence between the schematic and the standard CERES methods: when given a proof schema, one can instantiate it for some specific number n, get an **LK**-proof and apply the standard CERES method on it, or apply *CERES*$_s$ directly to the given proof schema and then instantiate the ACNF schema for the number n. Of course these two normal forms cannot be the same in general, since cut-elimination is not confluent in classical logics. Finally, we illustrate the method *CERES*$_s$ by applying it to some examples in a semi-automated way.

The rest of the thesis is organized in the following way: In Chapter 2, we introduce some basic concepts of proof theory, such as the sequent calculus **LK** and the cut-elimination theorem. The next chapters benefit a lot from [DLRW12, DLRW].

In Chapter 3 the notions of formula and proof schemata, as well the calculus $\mathbf{LK}_s$ and the resolution calculus for reasoning on schematic clause sets will be presented for propositional logic.

Chapter 4 is devoted to the definition of the *CERES*$_s$ method for propositional $\mathbf{LK}_s$ and the data structures central to it: the characteristic term schema and the projection term schema, from which characteristic clause set schema and projection set can be extracted respectively. Also, the main

theoretical result on the $CERES_s$ method is stated.

Then, in Chapter 5, the language, as well the sequent and resolution calculus is extended to first-order logic. The $CERES_s$ method is adjusted to first-order proof schemata and its complexity is discussed.

Moreover, in Chapter 6, we apply the method semi-automatically to some (formalizations of) mathematical proofs. Finally, in Chapter 7 we summarize our work and give some hints about the future work.

Chapter 2

Preliminaries

In this chapter we define basic notions of proof theory, on which this thesis is based. First, the sequent calculus **LK** is defined. Then the cut-elimination theorem for the calculus **LK** will be given and Gentzen's reductive method of cut-elimination be described. In the third section, **LK** is extended with the induction rule and cut-elimination in the extended calculus is discussed. Finally, the CERES method is described informally.

2.1 Sequent Calculus LK

The first-order language, defined in [Tak87], consists of countable sets of variables, n-ary function and predicate symbols. *Terms* are built in the usual inductive fashion from variables and function symbols; and *formulas* are built inductively from atoms using the logical connectives $\neg, \wedge, \vee, \supset, \forall$ and $\exists$ as usual. A variable occurrence in a formula is called *bound* if it is in the scope of $\forall$ or $\exists$ connectives, otherwise it is called *free*. The notions of interpretation, satisfiability and validity of formulas are defined in the usual classical sense.

Atoms are *subformulas* of itself and the subformulas of formulas $\neg A$, $A \bullet B$ and $(Qx)A(x)$, for $\bullet \in \{\wedge, \vee, \supset\}$ and $Q \in \{\forall, \exists\}$, are the subformulas of A, A and B, and $A(t)$ respectively, for an arbitrary term t, and the formulas itself.

Definition 2.1.1 (Sequent)**.** An expression of the form $\Gamma \vdash \Delta$, where Γ and Δ are finite (maybe empty) multisets of formulas, is called a *sequent*. Γ is called an *antecedent* and Δ is called a *succedent* of the sequent. Two sequents are *equal* iff multisets represented by their antecedents and respectively succedents are equal.

Subformulas of a sequent are subformulas of those formulas, which occur in the sequent. We will use Greek letters $\Gamma, \Delta, \Pi, \Lambda$ (possibly with subscripts) to denote finite (maybe empty) multisets of formulas.

Semantically a sequent $S : A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ is equivalent to the formula $F : (A_1 \wedge \cdots \wedge A_n) \supset (B_1 \vee \cdots \vee B_m)$. We say that I is an interpretation of S iff it is an interpretation of F. S is valid iff F is valid. An empty sequent $\vdash$ corresponds to $\top \supset \bot$, which is $\bot$ (falsum).

Definition 2.1.2 (Inference)**.** An inference is an expression of the form

$$\frac{S_1}{S} \quad \text{or} \quad \frac{S_1 \qquad S_2}{S}$$

where S_1 and S_2 are called upper sequents and S is called lower sequent of the inference. The formulas, the inference operates on, are called *auxiliary formulas* and the formula, it derives, is called *principal formula*. The auxiliary formulas are called *ancestors* of the principal formula. The ancestor relationship is transitive.

Definition 2.1.3 (Initial sequent)**.** A sequent of the form $A \vdash A$, where A is an arbitrary atomic formula, is called an initial sequent.

Definition 2.1.4 (Calculus **LK**)**.** The sequent calculus **LK** contains initial sequents as axioms and consists of the following inference rules:

1. Propositional rules:
   - $\neg$ introduction
     $$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta} \neg : l \quad \text{and} \quad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A} \neg : r$$
   - $\wedge$ introduction
     $$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \wedge : l \quad \text{and} \quad \frac{\Gamma \vdash \Delta, A \qquad \Pi \vdash \Lambda, B}{\Gamma, \Pi \vdash \Delta, \Lambda, A \wedge B} \wedge : r$$
   - $\vee$ introduction
     $$\frac{A, \Gamma \vdash \Delta \qquad B, \Pi \vdash \Lambda}{A \vee B, \Gamma, \Pi \vdash \Delta, \Lambda} \vee : l \quad \text{and} \quad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B} \vee : r$$
   - $\supset$ introduction
     $$\frac{\Gamma \vdash \Delta, A \qquad B, \Pi \vdash \Lambda}{A \supset B, \Gamma, \Pi \vdash \Delta, \Lambda} \supset : l \quad \text{and} \quad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \supset B} \supset : r$$
2. Quantifier rules:
   - $\forall$ introduction

$$\frac{A(t), \Gamma \vdash \Delta}{(\forall x)A(x), \Gamma \vdash \Delta} \forall\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A(u)}{\Gamma \vdash \Delta, (\forall x)A(x)} \forall\colon r$$

where t is an arbitrary term and u is a free variable not occurring in the lower sequent; u is called an *eigenvariable* and the condition, that it should not occur in the lower sequent is called the *eigenvariable condition*. The $\forall\colon l$ rule is called a weak quantifier rule and the $\forall\colon r$ rule is called a strong quantifier rule.

- $\exists$ introduction

$$\frac{A(u), \Gamma \vdash \Delta}{(\exists x)A(x), \Gamma \vdash \Delta} \exists\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, (\exists x)A(x)} \exists\colon r$$

where t is an arbitrary term and u is an eigenvariable satisfying the eigenvariable condition. The $\exists\colon l$ rule is called a strong quantifier rule and the $\exists\colon r$ rule is called a weak quantifier rule.

3. Structural rules:

- Weakening rules:

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} w\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A} w\colon r$$

- Contraction rules:

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} c\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A} c\colon r$$

- Cut rule:

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Pi \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda}\, cut$$

Remark 2.1.5. Since we have defined sequents by means of multisets, we do not need exchange or permutation rules.

Definition 2.1.6 (**LK**-derivation)**.** An **LK**-derivation ϕ is a rooted tree, where nodes are sequents and edges are inference rules. The root sequent S is called an end-sequent and it is written at the bottom. Let $\mathcal{A}$ be the set of sequents at the leaf nodes of ϕ, then we say that ϕ is a derivation of S from $\mathcal{A}$ (notation $\mathcal{A} \vdash_{\mathbf{LK}_s} S$).

Definition 2.1.7 (**LK**-proof and **LK**-subproof)**.** An **LK**-proof is an **LK**-derivation with initial sequents at the leaf nodes. If ϕ is an **LK**-proof of the end-sequent $\Gamma \vdash \Delta$, then we write:

$$\begin{array}{c} \phi \\ \Gamma \vdash \Delta \end{array}$$

and say that $\Gamma \vdash \Delta$ is provable in **LK**. An **LK**-proof without the *cut* rule is called *cut-free*. An **LK**-subproof of ϕ is any sub-tree of ϕ which is also an **LK**-proof.

Proposition 2.1.1. *The calculus* **LK** *is sound and complete, i.e. a sequent S is derivable in* **LK** *iff S is valid.*

Proof. In [Tak87]. □

2.2 The Resolution Calculus

Resolution is a very important tool of automated deduction. It was first introduced by Robinson in [Rob65]. Here we briefly introduce the resolution calculus as a specific sequent calculus with atomic sequents, atomic cuts and unification. For a detailed textbook about the resolution calculus we refer the interested reader to [Lei97].

Definition 2.2.1 (Clause)**.** A *clause* is a sequent $\Gamma \vdash \Delta$ where Γ and Δ are multisets of atoms. An empty clause is denoted by $\vdash$.

Clauses are denoted by $C, D, \ldots$ and sets of clauses, shortly *clause sets* by $\mathcal{C}, \mathcal{D}, \ldots$.

Definition 2.2.2 (Substitution)**.** A mapping σ from the set of variables to the set of terms is called a *substitution* iff $\sigma(x) \neq x$ only for finitely many variables.

Definition 2.2.3 (Unifier)**.** A substitution σ is called a *unifier* of a nonempty set of expressions Σ, if the set $\Sigma\sigma$ contains exactly one element.

Definition 2.2.4 (Most general unifier)**.** A substitution σ is called a *most general unifier* (shortly m.g.u) of a nonempty set of expressions Σ, if for every unifier ϑ of Σ there exists a substitution θ, such that $\sigma\theta = \vartheta$.

Now we define the resolution rule, that is a generalization of the *mix* rule by unification, but operates on atomic formulas only.

Definition 2.2.5 (Resolution)**.** The following rule is called a resolution rule:

$$\frac{\Gamma \vdash \Delta, L, \ldots, L \qquad M, \ldots, M, \Pi \vdash \Lambda}{\Gamma\sigma, \Pi\sigma \vdash \Delta\sigma, \Lambda\sigma}$$

Where Δ, Π does not contain L, M; the clauses $\Gamma \vdash \Delta, L, \ldots, L$ and $M, \ldots, M, \Pi \vdash \Lambda$ are variable disjoint and σ is an m.g.u of the atoms L, M. A clause $\Gamma\sigma, \Pi\sigma \vdash \Delta\sigma, \Lambda\sigma$ is called a resolvent of the clauses $\Gamma \vdash \Delta, L, \ldots, L$ and $M, \ldots, M, \Pi \vdash \Lambda$.

Definition 2.2.6 (Resolution deduction)**.** A deduction tree having clauses as nodes and resolution rules as edges is called a *resolution deduction*. If ϕ is a resolution deduction, C is a root node of ϕ and $\mathcal{C}$ is a set of leaf nodes of ϕ, then we say that ϕ is a resolution deduction of C from the clause set $\mathcal{C}$, or simply C is *derivable* from $\mathcal{C}$. A resolution deduction of the $\vdash$ from a set of clauses $\mathcal{C}$, is called a *resolution refutation* of $\mathcal{C}$.

Proposition 2.2.1. *The resolution calculus is sound and refutationally complete, i.e.*

- *if a clause is derivable from a set of clauses, then it is logical consequence of the clause set,*
- *if a set of clauses is unsatisfiable, then there exists a refutation of it.*

Proof. In [Lei97]. □

2.3 The Cut-Elimination Theorem

In this section we recall Gentzen's cut-elimination theorem and some of its consequences, which was given for first-order sequent calculus **LK** in [Tak87]. We proceed with some basic definitions.

Definition 2.3.1 (Regularity)**.** An **LK**-proof is called regular, if all eigenvariables are distinct and if a free variable u occurs as an eigenvariable in a sequent S of the **LK**-proof, then u occurs only in sequents above S.

It is necessary for reductive cut-elimination methods proof to be regular. Clearly, every **LK**-proof can be regularized simply by renaming the eigenvariables. The other important notions for reductive methods are *grade* and *rank*.

Definition 2.3.2 (Grade)**.** The number of logical symbols occurring in a formula A is called a grade of A (or complexity of A) and is denoted with $comp(A)$. The grade of the cut rule is the grade of the cut-formula.

The notion of *formula occurrence* in a proof ϕ is defined as usual and if Ω is a set of formula occurrences and o is a formula occurrence, then we say that o is an Ω-ancestor, if o is an ancestor of some $o' \in \Omega$. If Ω is a set of cut-formula occurrences, then o is called a cut-ancestor.

Definition 2.3.3 (Rank)**.** Let ϕ be a proof with the cut rule as the last inference. Let ϕ_l, ϕ_r be two subproofs of ϕ with cut rule left and right premises as end-sequents. The left (right) rank of ϕ is a number of sequents in the ϕ_l (ϕ_r), in which the cut-formula occurs in its succedent (antecedent). The rank of ϕ is a sum of the left and right ranks and denoted with $rank(\phi)$.

Note that, according to the definition above, a proof ϕ is either cut-free or $rank(\phi) \geq 2$.

Theorem 2.3.1 (Cut-elimination)**.** *If a sequent S is provable in* **LK***, then it is provable without a cut rule.*

Proof (Sketch). The proof of theorem is by double induction over grade and rank of the cut rule, reducing them with the transformation rules given in Appendix A. The full detailed proof can be found in [Tak87]. □

Gentzen's original proof is using the *mix* rule, but *mix* can be simulated using the contraction and *cut* rules. Therefore the transformation rules in Appendix A differs from the original transformation rules defined in [Tak87].

Gentzen gave a constructive proof of his cut-elimination theorem from which a nondeterministic algorithm can be extracted. This algorithm is the following:

- Select an uppermost cut rule ρ.
- If the rank of ρ, $rank(\rho) = 2$, then use the grade reduction rules defined in Appendix A to reduce complexity of the cut-formula.
- If the rank of ρ, $rank(\rho) > 2$, then use the rank reduction rules defined in Appendix A to reduce the rank of ρ.
- Repeat this procedure until all cuts are eliminated.

The most famous consequences of the cut-elimination theorem are Gentzen's midsequent theorem and the subformula property.

Theorem 2.3.2 (Mid-sequent)**.** *If a sequent S, which contains only prenex formulas, is provable in* **LK***, then there is a cut-free proof of S, which contains a sequent S_1, called mid-sequent, which is quantifier free and*

- *Every inference above S_1 is either structural or propositional,*
- *Every inference after S_1 is either structural or a quantifier inference.*

Theorem 2.3.3 (Subformula property)**.** *Every sequent, which is* **LK***-provable, is provable with its subformulas only.*

2.4 The CERES method

CERES is a cut-elimination method by resolution first introduced in [BL00]. It was originally developed for first-order logic, but later, the CERES method was extended to second-order [HLWP09] and higher-order logics [HLW11]. Unlike reductive cut-elimination methods, which analyze only the derivation of the uppermost logical operator of a cut-formula, the CERES method is based on a *global* analysis of the proof. The method is very general and can be used in different first-order theories like *finitely valued* and *Gödel logics* (see [BL11]). Recently it was extended to *intuitionistic logic* as well [LRP12].

Here we describe the method informally. The CERES method consists of the following steps:

Skolemization of proof. A proof is skolemized if the end-sequent of the proof does not contain strong quantifier occurrences. Therefore skolemization of a proof means removal of strong quantifiers from its end-sequent. Skolemization is necessary because otherwise eigenvariable conditions may be violated in proof projections (see below). Note that even proofs with cuts can be skolemized, the cut-formulas itself cannot. For a formal definition of skolemization see [BL99].

Computation of characteristic clause set. Originally, in [BL00], a notion of characteristic clause set was introduced. Later, in [BL06], a notion of characteristic clause term was defined, which can be evaluated in a different ways to sets of clauses such as characteristic clause set, proof profile [Het08] and the like (see [Pal09]). Anyway, all these structures are based on analysis of cut-formula derivations in a proof. The cut-formula ancestors at the leaf nodes of the proof is selected, which correspond to singleton clauses[1], and according to cut-formula derivations the characteristic clause set (or similar structure mentioned above) is produced.

Refutation of characteristic clause set. The resolution refutation of characteristic clause set (proof profile or the like) serves as a skeleton of (essential) cut-free proof. The term *essential* means that there is at most atomic cuts in the proof. The elimination of atomic cuts (if it is possible at all) is mathematically inessential.

Note that if the atomic axiom sequents are not closed under cuts (like the equality axioms), then the atomic cuts cannot be eliminated. We

[1] If there are arbitrary atomic axioms admitted in the proof, then the selected clauses at the leafs are not singleton, but arbitrary.

refer the interested reader to [Tak87] for more detailed discussion of this issue.

Computation of proof projections. The proof projections are cut-free parts of the original proof, obtained by omitting inference rules producing a cut-formula. Hence if there is a strong quantifier rule application, producing a formula occurring in the end-sequent, and the corresponding weak quantifier rule application is omitted since it was producing a cut-formula, the proof projection will not be sound any more. Therefore skolemization of the original proof is necessary. There is a correspondence between sets of projections and characteristic clause sets: for every clause in the characteristic clause set, there exists a corresponding proof projection in the projection set. This correspondence is important for atomic cut normal form, since it ensures that every clause in the leaves of resolution refutation can be replaced by a proof projection.

Production of ACNF. The Atomic Cut Normal Form (ACNF) is a proof obtained from resolution refutation by plugging at the leaves the corresponding proof projections. Note that in general, the clauses in resolution refutation are variants of the clauses in characteristic clause set, therefore the substitutions used in the refutation should be applied to proof projections as well.

The formal definitions, extended to proof schemata, will be given in the next chapters, but still we refer the interested reader to the most recent book about the CERES method [BL11].

Chapter 3

Propositional Schematic Proof Systems

In this chapter we will introduce basic terms and notations that will be used throughout the whole thesis. First we define a notion of propositional formula schemata, syntax and semantics for it. Then a calculus, called $\mathbf{LK}_s$, will be given for reasoning on formula schemata. The last section is about propositional clause schemata, and a resolution calculus, called $\mathcal{R}_s$, will be defined. All these notions will be extended to first-order logic in Chapter 5.

3.1 Schematic Language

Propositional formula schemata were first introduced in [ACP09] and a subclass, called *regular schemata* was investigated and proved to be decidable. Later, in [ACP11] a new subclass, called *bound-linear schemata* was defined and a reduction algorithm to regular schemata was given, hence decidability of bound-linear class was proven. Here a slightly modified version of bound-linear schemata will be defined, but the expressive power (i.e. decidability) will be preserved.

3.1.1 Syntax

We assume a countably infinite set of *index variables* (intended to be interpreted over $\mathbb{N}$), and a countably infinite set of *propositional symbols*. As we will see later in this section, index variables can be free or bound. Free index variables are called *parameters*. The set of *linear arithmetic expressions* or simply *arithmetic expressions* is defined as usual from the constant 0 and the index variables over the signature $s, +$ (the usual properties of $+$ are

assumed, e.g. $s(0) + s(0)$ is equivalent to $s(s(0))$). We frequently identify $\alpha \in \mathbb{N}$ with $s^\alpha(0)$. If n is an index variable then $n + \cdots + n$ (α times) is denoted by $\alpha.n$. Also we denote:

- elements of $\mathbb{N}$ by $\alpha, \beta, \ldots$,
- bound index variables by $i, j, l, \ldots$,
- parameters by $k, m, n, \ldots$,
- linear arithmetic expressions by $a, b, \ldots$,
- propositional symbols by $p, q, \ldots$.

We say an arithmetic expression a is *ground* if a does not contain index variables. Obviously, ground arithmetic expressions can be rewritten to numerals and we identify them with their normal forms (and therefore with the natural numbers).

A *substitution* is a function mapping every index variable to an arithmetic expression. The substitution of an index variable k by an arithmetic expression a is denoted with $[k/a]$.

Definition 3.1.1 (Indexed proposition)**.** An expression of the form p_a, where a is an arithmetic expression and p a propositional symbol, is called an *indexed proposition*. If a is ground, then we speak about ground indexed propositions, which are called *propositional variables*.

Definition 3.1.2 (Formula schemata)**.** We define formula schemata inductively in the following way:

- An indexed proposition is an (atom) formula schema.
- If A and B are formula schemata, then so are $\neg A$, $A \vee B$, $A \wedge B$ and $A \supset B$.
- If A is a formula schema, a, b are arithmetic expressions and i is an index variable not bound in A, then $\bigwedge_{i=a}^{b} A$ and $\bigvee_{i=a}^{b} A$ are formula schemata such that i is bound in both formula schemata.

We denote formula schemata by $A, B, \ldots$. The notion of application of substitution σ to formula schema A is defined as usual and is denoted by $A\sigma$. The notation $A(k)$ is used to indicate a parameter k in A, and $A(a)$ then denotes $A[k/a]$.

Remark 3.1.3. We introduced $\supset$ for convenience and assume $\supset$ is right associative, then the chain of implications like $A(0) \supset A(1) \supset \cdots \supset A(n+1)$ can be represented as $(\bigvee_{i=0}^{n} \neg A(i)) \vee A(n+1)$ or $(\neg \bigwedge_{i=0}^{n} A(i)) \vee A(n+1)$. Note that $(((A(0) \supset A(1)) \supset A(2)) \cdots) \supset A(n+1)$ cannot be directly expressed in our formalism, but a sat-equivalent formula can be encoded.

Definition 3.1.4 (Subformulas)**.** Subformulas are defined as follows:

- An atomic formula schema is a subformula of itself.
- The subformulas of $\neg A$ are subformulas of A and $\neg A$ itself.
- The subformulas of $A \vee B$, $A \wedge B$ and $A \supset B$ are subformulas of A and of B and the formula itself.
- The subformulas of $\bigwedge_{i=a}^{b+1} A(i)$ are subformulas of $A(b+1)$ and of $\bigwedge_{i=a}^{b} A(i)$; and the formula itself.
- The subformulas of $\bigvee_{i=a}^{b+1} A(i)$ are subformulas of $A(b+1)$ and of $\bigvee_{i=a}^{b} A(i)$; and the formula itself.

Now, a formula schema A is *bound-linear* [ACP11] iff the following conditions are fulfilled:

1. A contains at most one parameter n.
2. Every indexed proposition in A is of the form $p_{\alpha.n+\beta.i+\gamma}$, where n is a parameter, i a bound variable, α, γ are arbitrary natural numbers and $\beta \in \{0, 1\}$.
3. If A contains an iteration $\bigwedge_{i=a}^{b} B$ (or $\bigvee_{i=a}^{b} B$) then a, b are of the form $\alpha.n + \beta$ and $\gamma.n + \kappa.j + \iota$ respectively, where $\alpha, \beta, \gamma, \iota$ are arbitrary natural numbers, $\kappa \in \{0, 1\}$ and j is a bound variable.

Informally, the indices and iteration bounds should contain at most one parameter and at most one bound variable with coefficient 1.

3.1.2 Semantics

An interpretation is a pair of functions, $I = (\mathcal{I}, \mathcal{I}_p)$, s.t. $\mathcal{I}$ maps index variables to the natural numbers and $\mathcal{I}_p$ maps propositional variables to the truth values. Let $\mathcal{I}(a)$ be a homomorphic extension of $\mathcal{I}$ for an arithmetic expression a. Let σ be a substitution, then by $I\sigma$ we denote the interpretation defined as follows: $\mathcal{I}_p\sigma = \mathcal{I}_p$ and $\mathcal{I}\sigma(n) = \mathcal{I}(n\sigma)$ for every index variable n.

A truth value $[\![A]\!]_I$ of a formula schema A in an interpretation I is defined inductively:

- $[\![p_a]\!]_I = \mathcal{I}_p(p_{\mathcal{I}(a)})$.
- $[\![\neg A]\!]_I = \mathbf{T}$ iff $[\![A]\!]_I = \mathbf{F}$.
- $[\![A \vee B]\!]_I = \mathbf{T}$ iff $[\![A]\!]_I = \mathbf{T}$ or $[\![B]\!]_I = \mathbf{T}$.
- $[\![A \wedge B]\!]_I = \mathbf{T}$ iff $[\![A]\!]_I = \mathbf{T}$ and $[\![B]\!]_I = \mathbf{T}$.
- $[\![A \supset B]\!]_I = \mathbf{T}$ iff $[\![A]\!]_I = \mathbf{F}$ or $[\![B]\!]_I = \mathbf{T}$.
- $\left[\!\!\left[\bigwedge_{i=a}^{b} A\right]\!\!\right]_I = \mathbf{T}$ iff for every natural number α s.t. $\mathcal{I}(a) \leq \alpha \leq \mathcal{I}(b)$, $[\![A]\!]_{I[i/\bar{\alpha}]} = \mathbf{T}$, where $\bar{\alpha}$ is a numeral corresponding to α.
- $\left[\!\!\left[\bigvee_{i=a}^{b} A\right]\!\!\right]_I = \mathbf{T}$ iff there is an natural number α s.t. $\mathcal{I}(a) \leq \alpha \leq \mathcal{I}(b)$, $[\![A]\!]_{I[i/\bar{\alpha}]} = \mathbf{T}$, where $\bar{\alpha}$ is a numeral corresponding to α.

Remark 3.1.5. If $\mathcal{I}(a) > \mathcal{I}(b)$, then $\bigwedge_{i=a}^{b} A$ is always true and $\bigvee_{i=a}^{b} A$ is always false for any formula schema A.

A formula schema A is satisfiable iff there is an interpretation I, s.t. $[\![A]\!]_I = \mathbf{T}$, otherwise A is unsatisfiable. If an interpretation I satisfies A, we write $I \models A$.

We say that a formula schema is ground, if it does not contain free index variables, i.e. parameters. Then obviously, a ground formula schema is a usual propositional formula (ground iterations are equivalent to a finite conjunction/disjunction of propositional variables).

Proposition 3.1.1. *The satisfiability problem is semi-decidable for formula schemata.*

Proof. Let A be a formula schema. Then for every interpretation I, $I \models A$ iff there exists a ground substitution σ, s.t. $I \models A\sigma$. Since $A\sigma$ is a ground formula schema, i.e. just a propositional formula, satisfiability of $A\sigma$ is decidable. Also the set of ground substitutions is recursively enumerable. Then checking satisfiability of A becomes semi-decidable. □

Now, we define a subclass of bound-linear schemata, called *regular* [ACP11]. A formula schema A is regular iff the following conditions are fulfilled:

1. A contains at most one parameter n.
2. If A contains an iteration $\bigwedge_{i=a}^{b} B$ (or $\bigvee_{i=a}^{b} B$) then a, b are of the form α and $n + \beta$ respectively, where $\alpha, \beta \in \mathbb{N}$, B does not contain any iteration and every indexed proposition in B is of the form $p_{i+\gamma}$, for $\gamma \in \mathbb{N}$.

3. All iterations in A have the same bound.

Proposition 3.1.2. *The satisfiability problem is decidable for bound-linear schemata.*

Proof (Sketch). The proof is by reduction to regular schemata. To do so, the following steps are needed:

1. Eliminate nested iterations.

2. Transform every iteration into an iteration over an interval of the form $[\alpha, n + \beta],\ \alpha, \beta \in \mathbb{N}$.

3. Remove the parameter from indices.

4. Align iterations in such a way that all iterations have the same bound.

Then, the obtained regular schema can be decided by a tableaux calculus, called *STAB*, with the following strategy: if applicable, first apply decomposition rules for propositional connectives, then apply either a so called looping rule or a closure rule. If none of these rules is applicable, decompose the iteration over a maximal interval with the iteration rules.

For a full description of transformation and the decision procedure for regular schemata we refer the interested reader to [ACP11]. □

The key point here is that the indices contain bound variables with coefficients ≤ 1. If we allow coefficients to be grater than one, then the class becomes undecidable. For more discussion about decidability and undecidability issues please consult [ACP11].

3.2 Sequent Calculus $\mathbf{LK}_s$

In this section we define a version of the classical propositional sequent calculus **LK**. It will differ from **LK** in two ways: The formulas it will operate on will be regular formula schemata, and special initial sequents called *proof links* will be allowed. Note that already in [ACP09], a tableaux calculus *STAB* for formula schemata is introduced. Our calculus differs from it in some aspects: most importantly, we include the cut rule, which allows the formalization of proofs that use lemmas. Furthermore, instead of a looping rule (which is geared towards automated theorem proving), we use a different approach, based on recursive specifications of proofs which is more suited to the formalization of proofs found by humans.

Definition 3.2.1 (Sequent schemata)**.** An expression of the form $\Gamma \vdash \Delta$, where Γ and Δ are finite (maybe empty) multisets of formula schemata, is called a sequent schema. If S is a sequent schema and a an arithmetic expression, the notation $S(a)$ is defined as for formula schemata.

Semantics of sequent schemata is defined in the similar way as for sequents, but with respect to the interpretation defined in Section 3.1.2. Now, initial sequent schemata are expressions of the form $A \vdash A$, where A is an arithmetic expression. Usually we will refer to sequent schemata as sequents.

Definition 3.2.2 (Proof link)**.** If φ is a proof symbol, a is an arithmetic expression, and S a sequent schema, then the expression $\frac{(\varphi(a))}{S}$ is called a *proof link*.

We have seen that the expressive power of bound-linear schemata is the same as the one of regular schemata. Since regular schemata are simpler and easier to handle, from now on we consider only regular schemata.

The schematic **LK** is built according to the usual rules of classical propositional sequent calculus **LK** with the proviso that schematic formulas are considered modulo the equalities $A(0) = \bigwedge_{i=0}^{0} A(i)$ and $(\bigwedge_{i=0}^{n} A(i)) \wedge A(n+1) = \bigwedge_{i=0}^{n+1} A(i)$ (and analogously for $\bigvee$)[1].

Definition 3.2.3 (Calculus $\mathbf{LK}_s$)**.** The sequent calculus $\mathbf{LK}_s$ contains initial sequent schemata or proof links as axioms and consists of the propositional and structural **LK** inference rules.

Definition 3.2.4 ($\mathbf{LK}_s$-proof and $\mathbf{LK}_s$-subproof)**.** An $\mathbf{LK}_s$-proof is a rooted tree, where the leaf nodes are initial sequents or proof links, other nodes are arbitrary sequents and edges are inference rules. If ϕ is an $\mathbf{LK}_s$-proof, then the root sequent $\Gamma \vdash \Delta$ of ϕ is called the end-sequent of ϕ; then we write:

$$\begin{array}{c} \phi \\ \Gamma \vdash \Delta \end{array}$$

and say that $\Gamma \vdash \Delta$ is provable in $\mathbf{LK}_s$. An $\mathbf{LK}_s$-proof without the *cut* rule is called *cut-free*. An $\mathbf{LK}_s$-subproof of ϕ is any sub-tree of ϕ which is also an $\mathbf{LK}_s$-proof.

We use the notation $\pi(k)$ and $\pi(a)$ for $\mathbf{LK}_s$-proofs analogously to formula schemata.

[1] These equalities are used so that the rules for $\wedge$ and $\vee$ can be applied to $\bigwedge$ and $\bigvee$.

An $\mathbf{LK}_s$-proof is called *ground* if it does not contain index variables and proof links. Note that a ground $\mathbf{LK}_s$-proof is essentially an $\mathbf{LK}$-proof (obtained by replacing ground $\bigwedge, \bigvee$ by a finite number of $\wedge, \vee$).

For practical use, we need to add a notion of recursion to the $\mathbf{LK}_s$-proofs defined above. This will yield our notion of *proof schemata*:

Definition 3.2.5 (Proof schemata)**.** Let $\psi_1, \ldots, \psi_\alpha$ be proof symbols and $S_1(n), \ldots, S_\alpha(n)$ be sequents. Then, a *proof schema* Ψ is a tuple of pairs[2]

$$\langle (\pi_1, \nu_1(k)), \ldots, (\pi_\alpha, \nu_\alpha(k)) \rangle$$

such that:

1. Each pair $(\pi_\beta, \nu_\beta(k))$ is associated with ψ_β for all $\beta = 1, \ldots, \alpha$,
2. π_β is a ground $\mathbf{LK}_s$-proof of $S_\beta(0)$, for all $\beta = 1, \ldots, \alpha$,
3. $\nu_\beta(k)$ is an $\mathbf{LK}_s$-proof of $S_\beta(k+1)$ such that $\nu_\beta(k)$ contains only one parameter k and proof links of the form:

$$\dfrac{(\psi_\beta(k))}{S_\beta(k)} \qquad \text{and/or} \qquad \dfrac{(\psi_\gamma(a))}{S_\gamma(a)}$$

   for $\beta < \gamma$ and a arbitrary.

We assume an identification between formula occurrences in the end-sequents of π_β and $\nu_\beta(k)$ (so that we can speak of occurrences in the end-sequents of ψ_β). We also say that $S_1(n)$ is the end-sequent of Ψ.

We now give a syntactic meaning to proof schemata: A proof schema Ψ with end-sequent $S(n)$ can be considered as a representation of a sequence $(\pi_0, \pi_1, \ldots)$ of ground $\mathbf{LK}_s$-proofs that prove the sequents $S(0), S(1), \ldots$. For this definition, we consider $\mathbf{LK}_s$-proofs as terms and define a rewrite system for them. The result of this system for $\alpha \in \mathbb{N}$ will be exactly the proof π_α from the sequence of proofs just mentioned.

Definition 3.2.6 (Evaluation of proof schemata)**.** Let Ψ be a proof schema given in Definition 3.2.5. We define the rewrite rules for proof links

$$\dfrac{(\psi_\beta(0))}{S} \to \pi_\beta, \qquad \text{and} \qquad \dfrac{(\psi_\beta(k+1))}{S} \to \nu_\beta(k)$$

[2]Sometimes we may also write $\langle \psi_1, \ldots, \psi_\alpha \rangle$

for all $1 \leq \beta \leq \alpha$.

Now, for $\gamma \in \mathbb{N}$ we define $\psi_\beta{\downarrow_\gamma}$ as a normal form of $\dfrac{(\psi_\beta(\gamma))}{S(\gamma)}$ under the rewrite system just given. Further, we define $\Psi{\downarrow_\gamma} = \psi_1{\downarrow_\gamma}$.

Remark 3.2.7. Strictly speaking, $\psi_\beta \downarrow_\gamma$ is the normal form of $\dfrac{(\psi_\beta(\bar{\gamma}))}{S(\bar{\gamma})}$ where $\bar{\gamma}$ is the numeral for γ.

Example 3.2.8. Let us consider the following proof schema Ψ:

$$\langle(\pi, \nu(k))\rangle,$$

where π is:

$$\dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_0 \supset p_1 \vdash p_1} \supset : l$$

and $\nu(k)$:

$$\dfrac{\dfrac{\dfrac{(\psi(k))}{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}) \vdash p_{k+1}} \qquad p_{k+2} \vdash p_{k+2}}{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}), p_{k+1} \supset p_{k+2} \vdash p_{k+2}} \supset : l}{p_0, \bigwedge_{i=0}^{k+1}(p_i \supset p_{i+1}) \vdash p_{k+2}} \wedge : l$$

Then, according to Definition 3.2.6, $\Psi{\downarrow_0}$ is just π; $\Psi{\downarrow_1}$ is:

$$\dfrac{\dfrac{\dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_0 \supset p_1 \vdash p_1} \supset : l \qquad p_2 \vdash p_2}{p_0, p_0 \supset p_1, p_1 \supset p_2 \vdash p_2} \supset : l}{p_0, (p_0 \supset p_1) \wedge (p_1 \supset p_2) \vdash p_2} \wedge : l$$

$\Psi{\downarrow_2}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_0 \supset p_1 \vdash p_1} \supset : l \qquad p_2 \vdash p_2}{p_0, p_0 \supset p_1, p_1 \supset p_2 \vdash p_2} \supset : l}{p_0, (p_0 \supset p_1) \wedge (p_1 \supset p_2) \vdash p_2} \wedge : l \qquad p_3 \vdash p_3}{p_0, (p_0 \supset p_1) \wedge (p_1 \supset p_2), p_2 \supset p_3 \vdash p_3} \supset : l}{p_0, (p_0 \supset p_1) \wedge (p_1 \supset p_2) \wedge (p_2 \supset p_3) \vdash p_3} \wedge : l$$

and so on.

Now we prove that the definition of evaluation of proof schemata is indeed correct.

Proposition 3.2.1 (Soundness)**.** *Let* $\Psi\colon \langle(\pi_1, \nu_1(k)), \ldots, (\pi_\alpha, \nu_\alpha(k))\rangle$ *be a proof schema with end-sequent* $S(n)$*. Then, for every* $\gamma \in \mathbb{N}$ *and* $1 \leq \beta \leq \alpha$*,* $\psi_\beta{\downarrow_\gamma}$ *is a ground* $\mathbf{LK}_s$*-proof with end-sequent* $S_\beta(\gamma)$*. Hence* $\Psi{\downarrow_\gamma}$ *is a ground* $\mathbf{LK}_s$*-proof with end-sequent* $S(\gamma)$*.*

Proof. We proceed by double induction over α and γ. Assume $\alpha = 1$. If $\gamma = 0$, then the proof link rewrites to π_1, which is as desired by definition. Let assume $\gamma > 0$ and the proposition holds for all natural numbers less than γ. By definition, $\nu_1(k)$ may only contain proof links to $\psi_1(k)$ and by induction hypothesis $\dfrac{(\psi_1(\gamma-1))}{S_1(\gamma-1)}$ rewrites to an $\mathbf{LK}_s$-proof of $S_1(\gamma-1)$. Hence the proof link to $\psi_1(\gamma)$ rewrites to an $\mathbf{LK}_s$-proof of $S_1(\gamma)$. All these $\mathbf{LK}_s$-proofs are ground since π_1 is ground and k is the only parameter of $\nu_1(k)$.

Now, assume $\alpha > 1$ and the proposition holds for all proof schemata with proof symbols less than α. Again, if $\gamma = 0$, the proof link rewrites to π_β, for all $1 \leq \beta \leq \alpha$. If $\gamma > 0$, then for proof links to $\psi_\beta(\gamma-1)$ the above argument applies. So, consider proof links to $\psi_{\beta'}(\gamma')$, for $\beta' > \beta$. Then obviously, the proof schema $\Psi'\colon \langle(\pi_{\beta'}, \nu_{\beta'}(k)), \ldots, (\pi_\alpha, \nu_\alpha(k))\rangle$ has less proof symbols than Ψ, therefore by (outer) induction hypothesis the proposition holds for Ψ'. Then clearly, $\psi_\beta{\downarrow_\gamma}$ is a ground $\mathbf{LK}_s$-proof of $S_\beta(\gamma)$ for all $1 \leq \beta \leq \alpha$. □

Note that the calculus $\mathbf{LK}_s$ is not complete in general, but it is complete for a subset of regular schemata, where all iterations are aligned on the interval $[0, n+\alpha]$ for some $\alpha \in \mathbb{N}$. For such a valid regular schema, a proof schema can be obtained in a similar way as was discussed in the proof of Proposition 3.1.2. The difference is that instead of a looping rule we put a proof link in the proof.

3.3 Resolution Calculus $\mathcal{R}_s$

In this section we formalize the resolution calculus as a specific sequent calculus with atomic sequent schemata and atomic cuts. Later in Chapter 5 we extend the calculus to first-order schemata with unification. The resolution calculus for formula schemata was investigated in [AP11, AEP]. In [AP11] a translation from tableaux to resolution calculus was defined, but the format of specification is quite complex, hard to read and not intuitive to be interpreted by humans (even for trivial examples refutations are very long and counterintuitive). Later, in [AEP] a version of propositional resolution calculus schemata was defined and its properties were investigated. Except usual resolution and factorization rules, the so called *pruning rule* was introduced, which is used to detect cycles in the refutation.

Here we define the calculus $\mathcal{R}_s$, which is more general than the one defined in [AEP]. Our notion of resolution proof will be based on recursion, therefore we do not need the pruning rule. We define the calculus as a term

algebra. Such kind of definition is not new and it was investigated for example in [CL08, FMWP10]. Our notation differs from the usual ones in the following way: while usually the resolution term is defined as a binary term, written in infix form, we define the resolution term as a ternary term and write it in suffix form. The third place in a resolution term is needed to indicate the resolved atom explicitly.

Definition 3.3.1 (Clause)**.** A clause is a sequent $\Gamma \vdash \Delta$ where Γ and Δ are multisets of indexed propositions (atom formula schemata). An empty clause is denoted with $\vdash$.

We denote clauses with C, D and sets of clauses with $\mathcal{C}, \mathcal{D}$. As for proofs, we require that a clause contains at most one parameter and it is called ground, if it is parameter-free.

Definition 3.3.2 (Literal)**.** An indexed proposition, or a negated indexed proposition is called a literal. Indexed propositions are called positive literals and negated indexed propositions are called negative literals.

So far all these definitions were somewhat standard. Now, we introduce a type of variables, called *clause variables*, which are variables over clauses and are denoted with $X, Y, \ldots$. These variables correspond to some kind of "placeholders" for clauses. Below we define clause schemata and will see that the use of clause variables is needed to schematize clauses, which differ from each other by a fixed number of atoms.

Definition 3.3.3 (Clause schema)**.** A clause schema is defined recursively as follows:

- Clauses and clause variables are clause schemata w.r.t. $\mathcal{R} = \emptyset$.
- $c(n, X_1, \ldots, X_\alpha)$ is a clause schema w.r.t. $\mathcal{R}$:

$$\begin{array}{rcl} c(0, X_1, \ldots, X_\alpha) & \to & C \circ X_1 \circ \cdots \circ X_\alpha; \\ c(k+1, X_1, \ldots, X_\alpha) & \to & c(k, X_1, \ldots, X_\alpha) \circ D \end{array}$$

  where C is a ground clause and D an arbitrary clause not containing parameter different from k.

- finally, if c_1 and c_2 are clause schemata w.r.t. $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively, then $c_1 \circ c_2$ is a clause schema w.r.t $\mathcal{R}_1 \cup \mathcal{R}_2$.

The intended meaning of the $\circ$ operator is that if given two clauses $\Gamma \vdash \Delta$ and $\Pi \vdash \Lambda$, then $(\Gamma \vdash \Delta) \circ (\Pi \vdash \Lambda)$ is the clause $\Gamma, \Pi \vdash \Delta, \Lambda$. Note that the $\circ$ operator semantically corresponds to a disjunction.

Remark 3.3.4. Definition 3.3.3 admits the representation of clauses of variable length, in contrast to the sequents of $\mathbf{LK}_s$ calculus.

The notion of application of substitution is extended to clause schemata in a straightforward way. Additionally we introduce a *clause substitution* to be a mapping from clause variables to clauses. Let $C_1, \ldots, C_\alpha$ be clauses not containing parameters different from n and $\gamma \in \mathbb{N}$, then $\theta = [X_1/C_1, \ldots, X_\alpha/C_\alpha]$ is a clause substitution and $c(n, C_1, \ldots, C_\alpha)\downarrow_\gamma$ denotes a clause which is normal form of $c(n, X_1, \ldots, X_\alpha)\theta[n/\gamma]$ w.r.t. $\mathcal{R}$. We denote clause schemata with $c, d, \ldots$.

Example 3.3.5. Let X be a clause variable. Consider a clause schema $c(n, X)$ w.r.t $\mathcal{R}$:

$$\begin{array}{rcl} c(0, X) & \to & (\vdash p_0) \circ X, \\ c(k+1, X) & \to & c(k, X) \circ (\vdash p_{k+1}). \end{array}$$

Then $c(n, \vdash)\downarrow_\alpha$ and $c(n, \vdash q_0)\downarrow_\alpha$ are clauses $\vdash p_0, \ldots, p_\alpha$ and $\vdash q_0, p_0, \ldots, p_\alpha$ respectively, for all $\alpha \geq 0$.

Below we define so-called resolution terms, which corresponds to some kind of skeleton for resolution deductions. But the definition of resolution terms is much more general and not every resolution term is a resolution deduction in the usual sense.

Definition 3.3.6 (Resolution term)**.** We define resolution terms inductively:

- A clause schema c w.r.t. a rewrite system $\mathcal{R}$ is a resolution term.
- Let t_1 and t_2 be resolution terms w.r.t. $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively and A be an indexed proposition. Then $r(t_1; t_2; A)$ is a resolution term w.r.t. $\mathcal{R}_1 \cup \mathcal{R}_2$.

Let $\mathcal{C}$ be a set of clause schemata. A resolution term t is *based on* $\mathcal{C}$ if all (instances of) clause schemata in t are also in $\mathcal{C}$.

A normal form of a resolution term is computed by normalizing all clause schemata occurring in the term.

Example 3.3.7. Let $c(n, X)$ be the clause schema w.r.t. rewrite system $\mathcal{R}$ defined in Example 3.3.5. Then

$$r(r(c(n, X);\ p_n \vdash;\ p_n);\ q_0, q_1 \vdash;\ q_0)$$

is a resolution term. Informally, the normal form of this term for $\alpha \in \mathbb{N}$ and $X = \vdash q_0$ is:

$$r(r(\vdash q_0, p_0, \ldots, p_\alpha;\ p_\alpha \vdash;\ p_\alpha);\ q_0, q_1 \vdash;\ q_0).$$

Note that the inner resolution term always corresponds to resolution deduction in usual sense, but the outer term is not, until X is instantiated to $\vdash q_0$. Anyway, the inference is correct and the argument is the following: let $X = \Gamma \vdash \Delta,\ q_0 \notin \Delta$, then the term for $\alpha \in \mathbb{N}$ is:

$$r(r(\Gamma \vdash \Delta, p_0, \ldots, p_\alpha;\ p_\alpha \vdash;\ p_\alpha);\ q_0, q_1 \vdash;\ q_0)$$

and the end-sequent is $q_1, \Gamma \vdash \Delta, p_0, \ldots, p_{\alpha-1}$, but it is a weakening of $\Gamma \vdash \Delta, p_0, \ldots, p_{\alpha-1}$ (the end-sequent of inner resolution step). Therefore the inference is still correct, but redundant.

The above example motivates the definition of notion of resolution deduction, which ensures that r-expressions without clause variables are evaluated to resolution deductions in our sense.

Definition 3.3.8 (Resolvent)**.** Let $C\colon \Gamma \vdash \Delta$ and $D\colon \Pi \vdash \Lambda$ be clauses and A an indexed proposition. Then $|r(C, D, A)|$ is a clause defined as $\Gamma, \Pi \setminus A \vdash \Delta \setminus A, D_2$, where $\Pi \setminus A$ and $\Delta \setminus A$ denotes the multi-sets of atoms in Π and Δ respectively, after removal of all occurrences of A. The clause $|r(C, D, A)|$ is called a *resolvent* of C and D on A and A is called a *resolved atom*.

Remark 3.3.9. In case A does not occur in Δ and/or Π, the clause $|r(C, D, A)|$ is not a resolvent in the usual sense, but a clause which is subsumed by C or D; thus, also in this case, $|r(C, D, A)|$ is a logical consequence of C and D.

Definition 3.3.10 (Resolution deduction)**.** If C is a clause, then C is a resolution deduction with end-sequent C. If ϱ_1 and ϱ_2 are resolution deductions with end-sequents D_1 and D_2 respectively, then the resolution term $r(\varrho_1, \varrho_2, A)$ is a resolution deduction with end-sequent $D = |r(D_1, D_2, A)|$.

Let t be a resolution term based on $\mathcal{C}$; if t is a resolution deduction, then we say t *is resolution deduction from* $\mathcal{C}$. Additionally, if the end-sequent of t is $\vdash$, then t is called a *resolution refutation* of $\mathcal{C}$.

Remark 3.3.11. According to this definition, every resolution deduction is a resolution term, but not vice versa. For example, a clause schema is resolution term, but not resolution deduction.

It is clear, that any resolution deduction can easily be transformed into a resolution tree in an obvious way.

Example 3.3.12. Let us consider the resolution term from Example 3.3.7 and assume $n = 1$, $X = \vdash q_0$. Then $r(r(\vdash q_0, p_0, p_1;\ p_1 \vdash;\ p_1);\ q_0, q_1 \vdash;\ q_0)$ corresponds to the following resolution tree:

$$\dfrac{\dfrac{\vdash q_0, p_0, p_1 \qquad p_1 \vdash}{\vdash q_0, p_0} \qquad q_0, q_1 \vdash}{q_1 \vdash p_0}$$

Now, we define a notion of resolution proof schema in the similar way as we did for proof schemata. We will add a notion of recursion to the resolution terms and associate these terms with rewriting system in such a way, that when a natural number is given, we evaluate the resolution proof schema to the resolution deduction.

Definition 3.3.13 (Resolution proof schema)**.** A resolution proof schema is a structure $R = ((\varrho_1, \ldots, \varrho_\alpha), \mathcal{R})$ where the ϱ_i denote resolution terms and $\mathcal{R} = \mathcal{R}_1 \cup \ldots \cup \mathcal{R}_\alpha$, where the $\mathcal{R}_i$ (for $0 \leq i \leq \alpha$) are defined as follows:

$$\begin{aligned} \varrho_i(0, \bar{X}_i) &\to s_i, \\ \varrho_i(k+1, \bar{X}_i) &\to t_i[\varrho_i(k, \bar{s}_0^i), \varrho_{l_1}(a_1^i, \bar{s}_1^i), \ldots, \varrho_{l_{j(i)}}(a_{j(i)}^i, \bar{s}_{j(i)}^i)] \end{aligned}$$

where

- $\bar{X}_i$ are vectors of clause variables occurring in s_i and t_i,
- s_i are parameter-free resolution terms,
- $a_1^i, \ldots, a_{j(i)}^i$ are arithmetic terms,
- $\bar{s}_0^i, \ldots, \bar{s}_{j(i)}^i$ are vectors of clause schemata,
- $t_i[\varrho_i(k, \bar{s}_0^i), \varrho_{l_1}(a_1^i, \bar{s}_1^i), \ldots, \varrho_{l_{j(i)}}(a_{j(i)}^i, \bar{s}_{j(i)}^i)]$ are resolution terms after replacement of some clause schemata by the terms $\varrho_i(k, \bar{s}_0^i), \varrho_{l_1}(a_1^i, \bar{s}_1^i), \ldots, \varrho_{l_{j(i)}}(a_{j(i)}^i, \bar{s}_{j(i)}^i)$ where $i < l_1 < \cdots < l_{j(i)} \leq \alpha$.

Let θ be a clause substitution and $\gamma \in \mathbb{N}$, then $R{\downarrow_\gamma}$ denotes a resolution term which is the normal form of $\varrho_1(n, \bar{X}_1)\theta[n/\gamma]$ w.r.t. $\mathcal{R}$.

For a complete picture, we need to introduce notions of clause set schema and of resolution refutation schema, which will be a specific instance of a resolution proof schema.

Before we define a notion of clause set schema, we extend the composition operator to sets of clauses in the following way: let $\mathcal{C}, \mathcal{D}$ be two sets of clauses, then $\mathcal{C} \circ \mathcal{D} = \{C \circ D \mid C \in \mathcal{C}, D \in \mathcal{D}\}$. Note that the $\circ$ operator corresponds to a semantic disjunction of clause sets.

For example, assume $\mathcal{C} = \{\vdash p_0;\ \vdash q_0\}$ and $\mathcal{D} = \{p_0 \vdash;\ q_0 \vdash\}$, then $\mathcal{C} \circ \mathcal{D} = \{p_0 \vdash p_0;\ q_0 \vdash p_0;\ p_0 \vdash q_0;\ q_0 \vdash q_0\}$.

Now, clause set schemata will be defined in a similar way as resolution proof schemata, based on clause-set terms. Again we introduce a new type of variables, called *clause-set variables*, which are variables over clause-sets and are denoted with $\mathcal{X}, \mathcal{Y}, \ldots$.

Definition 3.3.14 (Clause-set term)**.** Clause-set terms are defined inductively using binary symbols $\oplus$ and $\otimes$ (which semantically correspond to conjunctions and disjunctions respectively) in the following way:

- Clause sets and clause-set variables are clause-set terms.
- If t_1 and t_2 are clause-set terms, then $t_1 \oplus t_2$ and $t_1 \otimes t_2$ are clause-set terms.

We say that clause-set term is *ground* if it does not contain clause-set variables and parameters.

Definition 3.3.15. Let t be a ground clause-set term, then the transformation $|t|$ is defined recursively:

- $|\mathcal{C}| = \mathcal{C}$, for $\mathcal{C}$ being a clause-set.
- $|t_1 \otimes t_2| = |t_1| \circ |t_2|$,
- $|t_1 \oplus t_2| = |t_1| \cup |t_2|$.

Definition 3.3.16 (Clause set schema)**.** A clause set schema is a structure $\mathcal{C}(n) = ((\mathcal{C}_1, \ldots, \mathcal{C}_\alpha), \mathcal{R})$ where the $\mathcal{C}_i$ are clause-set symbols denoting clause-set terms and $\mathcal{R} = \mathcal{R}_1 \cup \ldots \cup \mathcal{R}_\alpha$, where the $\mathcal{R}_i$ (for $0 \leq i \leq \alpha$) are defined as follows:

$$\begin{aligned} \mathcal{C}_i(0, \bar{X}_i, \bar{\mathcal{X}}_i) &\rightarrow s_i, \\ \mathcal{C}_i(k+1, \bar{X}_i, \bar{\mathcal{X}}_i) &\rightarrow t_i \end{aligned}$$

where s_i are ground clause-set terms and t_i are clause-set terms not containing parameters different from k and are obtained after replacement of some clause-set variables by $\mathcal{C}_i(k, \bar{X}_i, \bar{\mathcal{X}}_i), \mathcal{C}_{l_1}(a^i_1, \bar{X}_{l_1}, \bar{\mathcal{X}}_{l_1}), \ldots, \mathcal{C}_{l_{j(i)}}(a^i_{j(i)}, \bar{X}_{l_{j(i)}}, \bar{\mathcal{X}}_{l_{j(i)}})$ where $l_1, \ldots, l_{j(i)}$ are indices different from i and $a^i_1, \ldots, a^i_{j(i)}$ are arithmetic expressions such that $\mathcal{R}$ is always terminating.

Let *clause-set substitution* be a mapping from clause-set variables to clause-set terms. Let ϑ be a clause-set substitution, θ be a clause substitution and $\gamma \in \mathbb{N}$, then $\mathcal{C}(n)\downarrow_\gamma$ denotes a clause set $|\mathcal{C}|$ where $\mathcal{C}$ is a normal form of $\mathcal{C}_1(n, \bar{X}_1, \bar{\mathcal{X}}_1)\vartheta\theta[n/\gamma]$ w.r.t. $\mathcal{R}$. By abuse of notation we denote clause set schemata with $\mathcal{C}(n), \mathcal{D}(n), \ldots$.

Definition 3.3.17 (Resolution refutation schema)**.** A resolution proof schema is called a *resolution refutation schema of a clause set schema* $\mathcal{C}(n)$ if for every assignment β for n, $\varrho_1(n, \vdash, \ldots, \vdash)\downarrow_\beta$ is a resolution refutation of $\mathcal{C}(n)\downarrow_\beta$.

Example 3.3.18. Let consider the clause set schema $\mathcal{C}(n) = ((\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3), \mathcal{R})$ where $\mathcal{R}$ is:

$$\begin{array}{rcl}
\mathcal{C}_1(0) & \to & \{\vdash p_0;\ p_0 \vdash\} \\
\mathcal{C}_1(k+1) & \to & \mathcal{C}_2(k+1) \oplus \mathcal{C}_3(k+1) \\
\mathcal{C}_2(0) & \to & \{\vdash p_0\} \\
\mathcal{C}_2(k+1) & \to & \mathcal{C}_2(k) \otimes \{\vdash p_{k+1}\} \\
\mathcal{C}_3(0) & \to & \{p_0 \vdash\} \\
\mathcal{C}_3(k+1) & \to & \mathcal{C}_3(k) \oplus \{p_{k+1} \vdash\}
\end{array}$$

A resolution refutation schema of $\mathcal{C}(n)$ can be defined as the resolution proof schema $R = ((\varrho), \mathcal{R})$, where $\mathcal{R}$ is the following rewrite system:

$$\begin{array}{rcl}
\varrho(0, X) & \to & r((\vdash p_0) \circ X;\ p_0 \vdash;\ p_0), \\
\varrho(k+1, X) & \to & r(\varrho(k, (\vdash p_{k+1}) \circ X);\ p_{k+1} \vdash;\ p_{k+1})
\end{array}$$

Then a refutation of the clause set $\mathcal{C}(n)\downarrow_\alpha$ is defined by the term $\varrho(n, \vdash)\downarrow_\alpha$ for all $\alpha \in \mathbb{N}$.

Let compute some instances. For $\alpha = 0$, $\mathcal{C}(n)\downarrow_0 = \{\vdash p_0;\ p_0 \vdash\}$ and $\varrho(n, \vdash)\downarrow_0$ is

$$\dfrac{\vdash p_0 \qquad p_0 \vdash}{\vdash}$$

for $\alpha = 1$, $\mathcal{C}(n)\downarrow_1 = \{\vdash p_0, p_1;\ p_0 \vdash;\ p_1 \vdash\}$ and $\varrho(n, \vdash)\downarrow_1$ is

$$\dfrac{\dfrac{\vdash p_0, p_1 \qquad p_0 \vdash}{\vdash p_1} \qquad p_1 \vdash}{\vdash}$$

Informally, for all $\alpha \in \mathbb{N}$, $\mathcal{C}(n)\downarrow_\alpha = \{\vdash p_0, \ldots, p_\alpha;\ p_0 \vdash;\ p_1 \vdash; \ldots;\ p_\alpha \vdash\}$ and $\varrho(n, \vdash)\downarrow_\alpha$ is

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\vdash p_{\alpha-1}, p_\alpha} \qquad p_{\alpha-1} \vdash}{\vdash p_\alpha} \qquad p_\alpha \vdash}{\vdash}$$

Finally, we give soundness and completeness results for $\mathcal{R}_s$. Soundness is trivial, but the calculus is not complete in general. But it is refutationally complete for bound-linear schemata.

Proposition 3.3.1 (Soundness)**.** *The calculus $\mathcal{R}_s$ is sound: if an empty clause is derivable from a clause set schema $\mathcal{C}$, then $\mathcal{C}$ is unsatisfiable.*

Proof. Trivial by soundness of resolution rule. □

Now we prove a weak form of completeness. Namely, we prove that for every unsatisfiable clause set $\mathcal{C}$, there exists a refutation by means of [AEP]. But note that it is not possible to transform such kind of refutations into our formalism in general, because of two reasons: the first is the lack of mutual recursion and the second is that the refutations in [AEP] are built by starting to resolve on the literals with "maximal" indices, whereas we have the opposite for the resolution proof schemata described above.

Proposition 3.3.2 (Weak completeness)**.** *If a clause set schema $\mathcal{C}$ of bound-linear schemata is unsatisfiable, then there exists a refutation of $\mathcal{C}$.*

Proof (Sketch). The proof is by transforming every bound-linear clause set schema into a clause set by means of [AEP] and applying the corresponding completeness result (Theorem 7.1 in [AEP]).

There are two main steps in this transformation. The first step is to encode our inductive definitions in the clause form by means of [AEP]. Note that a clause set schema can be easily transformed into a tuple of definitions of the form

$$C(0) \to B \qquad \text{and} \qquad C(k+1) \to I \tag{3.1}$$

for B, I being some formula schemata, where I can contain other inductively defined formula schemata. Now, (3.1) can be encoded into the expression, for a new predicate symbol ϕ denoting C:

$$\{\phi_0 \Leftrightarrow B,\ \phi_{k+1} \Leftrightarrow I\} \tag{3.2}$$

where $A_1 \Leftrightarrow A_2$ is an abbreviation for $\neg A_1 \vee A_2, A_1 \vee \neg A_2$. Then, (3.2) can be directly translated into clausal form $\mathcal{C}$ by means of [AEP].

The second step is to normalize atoms, occurring in $\mathcal{C}$, in the sense of [AEP]. It is enough to encode every atom of the form $p_{k+\alpha}$, for some $\alpha > 1$, occurring in $\mathcal{C}$, in the following way: introduce new predicate symbol p^α, with the intended meaning $p^\alpha_k \Leftrightarrow p_{k+\alpha}$ and define

$$\left\{p^\beta_k \Leftrightarrow p^{\beta-1}_{k+1},\ p^0_k \Leftrightarrow p_k \mid 0 < \beta \leq \alpha\right\}$$

For more details about the transformation please consult [AEP]. □

3.3.1 Discussion

There are two main problems which arose from the calculus $\mathcal{R}_s$. The first is how to find a resolution refutation schema for the given clause set schema. The second is how to verify that the given resolution proof schema is indeed a resolution refutation schema of the given clause set schema. Although these questions are very natural, possible solutions are difficult and thus the subject of future research. Here we discuss some intuitive ways these problems can possibly be solved.

Intuitively, one strategy to find a resolution refutation schema is to "compute" recursions in clause set schema and then derive the empty clause from this "real" clause set. If the clauses in the resolution term are such that the number of literals depends on a parameter, then there is a need for clause variables to be introduced in this term. For example, consider the clause set schema from Example 3.3.18. It contains one clause with parameter-dependent literals. Being that the clause is used for the refutation, it is necessary there to be one clause variable in the resolution term.

A solution to the second problem can be to compute an instance of clause set and resolution proof schemata and check that every clause at the leaf nodes in the resolution refutation occur in the clause set. But of course this can not be done for all instances, therefore some kind of inductive checking is needed. Below we describe one such algorithm informally.

Let us given a clause set schema $\mathcal{C}(n)$ and a resolution proof schema $R = ((\varrho_1, \ldots, \varrho_\alpha), \mathcal{R})$. To check that R is a refutation schema of $\mathcal{C}(n)$ we need to check that for all $\beta \in \mathbb{N}$, $\varrho_1(n, \vdash, \ldots, \vdash)\downarrow_\beta$ is a refutation of $\mathcal{C}(n)\downarrow_\beta$, i.e. $\varrho_1(n, \vdash, \ldots, \vdash)\downarrow_\beta$ derives $\vdash$ and every clause at leaf nodes is in $\mathcal{C}(n)\downarrow_\beta$. The algorithm is the following:

1. Ensure that ϱ_1 is a refutation: check inductively the more general statement, that $\varrho_1(n, \bar{X}_1)$ derives $X_1 \circ \cdots \circ X_{\alpha_1}$ and for each $1 < \beta \leq \alpha$, $\varrho_\beta(n, \bar{X}_\beta)$ derives $C \circ X_{l_1} \circ \cdots \circ X_{l_\beta}$, for a clause C. On the base case of ϱ_β, for all $1 \leq \beta \leq \alpha$, simply check the resolution steps. In the inductive case, for $\varrho_\beta(k+1, \bar{X}_\beta)$, assume induction hypothesis that $\varrho_\gamma(a, \overline{C \circ X}_\gamma)$ proves $C_{l_1} \circ X_{l_1} \circ \cdots \circ C_{l_\gamma} \circ X_{l_\gamma}$ for all $\beta < \gamma \leq \alpha$ and check the resolution steps in $\varrho_\beta(k+1, \bar{X}_\beta)$. Then it is clear that $\varrho_1(n, \vdash, \ldots, \vdash)$ derives the empty clause.

2. Ensure that every clause schema occurring at leaf nodes in R is in $\mathcal{C}(n)$ modulo factoring.

It is clear that the step (1) can always be performed (even automatically), but to find a general algorithm, how one ought to construct the clause schemata from the clause set schema for the step (2), is an open problem.

Example 3.3.19. Let $\mathcal{C}(n) = ((\mathcal{C}_1, \mathcal{C}_2), \mathcal{R})$ be a clause set schema with $\mathcal{R}$:

$$\begin{array}{rcl} \mathcal{C}_1(0) & \to & \{\vdash p_0;\ p_0 \vdash\} \\ \mathcal{C}_1(k+1) & \to & \mathcal{C}_2(k+1) \oplus \{p_{k+1} \vdash\} \\ \mathcal{C}_2(0) & \to & \{\vdash p_0\} \\ \mathcal{C}_2(k+1) & \to & \mathcal{C}_2(k) \oplus \{p_k \vdash p_{k+1}\} \end{array}$$

The resolution term for $\mathcal{C}(0)$ is trivial. For $n > 0$, by "computing" the recursion of $\mathcal{C}_2$, we can see that $\vdash p_{k+1}$ is easily derivable. If this derivation is $\delta(k+1)$, then the resolution term for the step case is $r(\delta(k+1);\ p_{k+1} \vdash;\ p_{k+1})$. Therefore a resolution refutation schema of $\mathcal{C}(n)$ is $R = ((\varrho, \delta), \mathcal{R})$ with $\mathcal{R}$:

$$\begin{array}{rcl} \varrho(0) & \to & r(\vdash p_0;\ p_0 \vdash;\ p_0), \\ \varrho(k+1) & \to & r(\delta(k+1);\ p_{k+1} \vdash;\ p_{k+1}), \\ \delta(0) & \to & \vdash p_0, \\ \delta(k+1) & \to & r(\delta(k);\ p_k \vdash p_{k+1};\ p_k). \end{array}$$

Then a refutation of the clause set $\mathcal{C}(n)\downarrow_\alpha$ is defined by the term $\varrho(n)\downarrow_\alpha$ for all $\alpha \in \mathbb{N}$. Informally, for all $\alpha \in \mathbb{N}$, $\mathcal{C}(n)\downarrow_\alpha = \{\vdash p_0;\ p_0 \vdash p_1; \ldots;\ p_{\alpha-1} \vdash p_\alpha;\ p_\alpha \vdash\}$ and $\varrho(n)\downarrow_\alpha$ is

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash p_0 \qquad p_0 \vdash p_1}{\vdash p_1} \qquad p_1 \vdash p_2}{\vdots}}{\vdash p_{\alpha-1}} \qquad p_{\alpha-1} \vdash p_\alpha}{\vdash p_\alpha} \qquad p_\alpha \vdash}{\vdash}$$

and we can check that all clauses at the leaf nodes are in the clause set. Moreover, for this simple example we can apply an inductive argument and check steps (1) and (2) of the algorithm. For (1), it is easy to see that $\varrho(0)$ derives $\vdash$ and $\delta(n)$ derives $\vdash p_n$, then clearly, $\varrho(n)$ derives $\vdash$. For (2), obviously all of the clauses $\vdash p_0$, $p_0 \vdash$, $p_k \vdash p_{k+1}$ and $p_{k+1} \vdash$ are in $\mathcal{C}(n)$.

Chapter 4

The Method *CERES*$_s$

As was discussed in the introduction, reductive cut-elimination methods fail on cyclic proofs, because cut cannot be shifted over cycles. It is the case in our calculus also. Reductive cut-elimination methods work for proof schemata only if cuts can be eliminated inside the $\mathbf{LK}_s$-proof. Otherwise such methods does not work, because shifting cuts over proof links fail. Since CERES is based on the global analysis of the proof, we choose CERES to be extended to proof schemata. Hence we define the method *CERES*$_s$ for sequent calculus $\mathbf{LK}_s$.

To keep things simple, first *CERES*$_s$ for propositional $\mathbf{LK}_s$ is given, which will be extended to first-order $\mathbf{LK}_s$ in the next chapter. The point is that there is no need of skolemization and regularization in the propositional calculus. Therefore the problems related to skolemization and regularization, as well possible solutions will be discussed in Chapter 5.

4.1 Characteristic Terms

At the heart of the CERES method lies the *characteristic clause set*, which describes the cuts in a proof. The connection between cut-elimination and the characteristic clause set is that any resolution refutation of the characteristic clause set can be used as a skeleton of a proof containing only atomic cuts.

Recall the discussion from Section 2.4, that the characteristic clause set can either be defined directly, or it can be obtained via a transformation from a *characteristic term*. Here we use the later approach and define a *schematic characteristic term* and an evaluation of it to *characteristic clause set schema*.

Our main aim is to extend the usual inductive definition of the characteristic term to the case of proof links. This will give rise to a notion of

schematic characteristic term. As usual, a *clause-set term* is a term built inductively from clauses and the binary symbols $\otimes, \oplus$ (we assume associativity of these symbols and sometimes omit the unnecessary parenthesis in terms). The usual definition of the characteristic term depends upon the cut-status of the formula occurrences in a proof (i.e. whether a given formula occurrence is a cut-ancestor, or not). But assume that a formula occurrence in a proof link in a proof schema is a cut-ancestor. Then this formula gives rise to many formula occurrences in its evaluation, which all will be cut-ancestors. Therefore we need some machinery to track the cut-status of formula occurrences through proof links. Hence we introduce the notion of *cut-configuration*.

Definition 4.1.1 (Cut-configuration)**.** Let Ω be a set of formula occurrences from the end-sequent of a proof schema pair ψ, then Ω is called a *configuration for* ψ and if additionally, every formula occurrence in Ω is a cut-ancestor, then Ω is called a *cut-configuration for* ψ. If Ω is $\emptyset$, then Ω is called an *empty cut-configuration for* ψ.

Remark 4.1.2. For simplicity and readability, in practice we will write formulas in a set instead of occurrences.

Example 4.1.3. Let us consider the proof schema $\Psi = \langle \psi, \varphi \rangle$ of the sequent $p_0, \bigwedge_{i=0}^{n}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{n+1} p_i$, defined in the following way: ψ is associated with the pair: π_1:

$$\dfrac{\dfrac{\dfrac{}{p_0 \vdash p_0} \quad \dfrac{}{p_1 \vdash p_1}}{p_0, p_0 \supset p_1 \vdash p_1} \supset : l}{p_0, p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1} \wedge : r}{p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1} c : l$$

and $\nu_1(k)$:

$$\dfrac{\dfrac{(\psi(k))}{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{k+1} p_i} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\varphi(k+1))}{\bigwedge_{i=0}^{k+1} p_i \vdash \bigwedge_{i=0}^{k+1} p_i} \quad \dfrac{}{p_{k+1} \vdash p_{k+1}} \quad \dfrac{}{p_{k+2} \vdash p_{k+2}}}{p_{k+1}, p_{k+1} \supset p_{k+2} \vdash p_{k+2}} \supset : l}{\bigwedge_{i=0}^{k+1} p_i, p_{k+1}, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} \wedge : r}{\bigwedge_{i=0}^{k+1} p_i, \bigwedge_{i=0}^{k} p_i, p_{k+1}, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} w : l}{\bigwedge_{i=0}^{k+1} p_i, \bigwedge_{i=0}^{k+1} p_i, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} \wedge : l}{\bigwedge_{i=0}^{k+1} p_i, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} c : l}{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}), p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} cut}{p_0, \bigwedge_{i=0}^{k+1}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{k+2} p_i} \wedge : l$$

and φ is associated with the pair: $\pi_2 \colon p_0 \vdash p_0$ and $\nu_2(k)$:

$$\dfrac{\dfrac{\dfrac{(\varphi(k))}{\bigwedge_{i=0}^{k} p_i \vdash \bigwedge_{i=0}^{k} p_i} \quad \dfrac{}{p_{k+1} \vdash p_{k+1}}}{\bigwedge_{i=0}^{k} p_i, p_{k+1} \vdash \bigwedge_{i=0}^{k+1} p_i} \wedge : r}{\bigwedge_{i=0}^{k+1} p_i \vdash \bigwedge_{i=0}^{k+1} p_i} \wedge : l$$

The end-sequent of ψ is $p_0, \bigwedge_{i=0}^{n}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{n+1} p_i$ and the end-sequent of φ is $\bigwedge_{i=0}^{n} p_i \vdash \bigwedge_{i=0}^{n} p_i$. Then there are four configurations for φ: $\emptyset$, $\{\bigwedge_{i=0}^{n} p_i \vdash\}$, $\{\vdash \bigwedge_{i=0}^{n} p_i\}$ and $\{\bigwedge_{i=0}^{n} p_i \vdash\ ;\ \vdash \bigwedge_{i=0}^{n} p_i\}$. Clearly, for ψ there are eight configurations and in general, if there are m formula occurrences in the end-sequent of a proof ϕ, then there will be 2^m configurations for ϕ. But not all these configurations are cut-configurations. For example, for ψ the cut-configurations are $\emptyset$ and $\{\vdash \bigwedge_{i=0}^{n+1} p_i\}$ and for φ: $\{\bigwedge_{i=0}^{n} p_i \vdash\}$ and $\{\bigwedge_{i=0}^{n} p_i \vdash\ ;\ \vdash \bigwedge_{i=0}^{n} p_i\}$. Note that, according to the definition, $\emptyset$ is also a cut-configuration for φ, but this cut-configuration is not relevant for us (since it is not "reachable").

We will represent the characteristic term of a proof link in our object language: For all proof symbols ψ and its (cut-)configurations Ω we assume a unique indexed proposition symbol $\mathrm{cl}^{\psi,\Omega}$ called *clause-set symbol*. The intended semantics of $\mathrm{cl}_a^{\psi,\Omega}$ is "the characteristic clause set of $\psi(a)$, with the (cut-)configuration Ω".

In fact, a characteristic term can be computed w.r.t any configuration Ω, but not all these terms are interesting for us. We only need characteristic terms w.r.t relevant cut-configurations.

Definition 4.1.4 (Characteristic term)**.** Let π be an $\mathbf{LK}_s$-proof and Ω a (cut-)configuration. In the following, by $\Gamma_\Omega, \Delta_\Omega$ and Γ_C, Δ_C we will denote multisets of formulas of Ω- and cut-ancestors respectively and Γ, Δ denote multisets of formulas not being Ω- or cut-ancestor. Let ρ be an inference in π. We define a clause-set term $\Theta_\rho(\pi, \Omega)$ inductively:

- if ρ is an axiom of the form $\Gamma_\Omega, \Gamma_C, \Gamma \vdash \Delta_\Omega, \Delta_C, \Delta$, then $\Theta_\rho(\pi, \Omega) = \Gamma_\Omega, \Gamma_C \vdash \Delta_\Omega, \Delta_C$
- if ρ is a proof link of the form $\dfrac{(\psi(a))}{\Gamma_\Omega, \Gamma_C, \Gamma \vdash \Delta_\Omega, \Delta_C, \Delta}$ then define Ω' as the cut-configuration corresponding to formula occurrences from $\Gamma_\Omega, \Gamma_C \vdash \Delta_\Omega, \Delta_C$ and $\Theta_\rho(\pi, \Omega) = \mathrm{cl}_a^{\psi,\Omega'}$
- if ρ is a unary rule with immediate predecessor ρ', then $\Theta_\rho(\pi, \Omega) = \Theta_{\rho'}(\pi, \Omega)$.
- if ρ is a binary rule with immediate predecessors ρ_1, ρ_2, then
  - if the auxiliary formulas of ρ are Ω- or cut-ancestors, then $\Theta_\rho(\pi, \Omega) = \Theta_{\rho_1}(\pi, \Omega) \oplus \Theta_{\rho_2}(\pi, \Omega)$,
  - otherwise $\Theta_\rho(\pi, \Omega) = \Theta_{\rho_1}(\pi, \Omega) \otimes \Theta_{\rho_2}(\pi, \Omega)$.

Finally, define $\Theta(\pi, \Omega) = \Theta_{\rho_0}(\pi, \Omega)$, where ρ_0 is the last inference of π, and $\Theta(\pi) = \Theta(\pi, \emptyset)$.

Example 4.1.5. Let us consider the proof schema Ψ defined in Example 4.1.3 and compute the characteristic terms for the relevant cut-configurations. We denote nonempty cut-configurations in the following way:

$$\begin{array}{lcl} \Omega_\psi & = & \{\vdash \bigwedge_{i=0}^{n+1} p_i\} \\ \Omega_\varphi & = & \{\bigwedge_{i=0}^{n} p_i \vdash\} \\ \Omega'_\varphi & = & \{\bigwedge_{i=0}^{n} p_i \vdash \, ; \vdash \bigwedge_{i=0}^{n} p_i\} \end{array}$$

Then the characteristic terms of Ψ for these cut-configurations are:

$$\begin{array}{lcl} \Theta(\pi_1, \emptyset) & = & \vdash \otimes(\vdash \otimes \vdash) \\ \Theta(\pi_1, \Omega_\psi) & = & \vdash p_0 \oplus (\vdash \otimes \vdash p_1) \\ \Theta(\nu_1(k), \emptyset) & = & \mathrm{cl}_k^{\psi,\Omega_\psi} \oplus (\mathrm{cl}_{k+1}^{\varphi,\Omega_\varphi} \otimes (p_{k+1} \vdash \otimes \vdash)) \\ \Theta(\nu_1(k), \Omega_\psi) & = & \mathrm{cl}_k^{\psi,\Omega_\psi} \oplus (\mathrm{cl}_{k+1}^{\varphi,\Omega'_\varphi} \oplus (p_{k+1} \vdash \otimes \vdash p_{k+2})) \\ \\ \Theta(\pi_2, \Omega_\varphi) & = & p_0 \vdash \\ \Theta(\pi_2, \Omega'_\varphi) & = & p_0 \vdash p_0 \\ \Theta(\nu_2(k), \Omega_\varphi) & = & \mathrm{cl}_k^{\psi,\Omega_\varphi} \otimes p_{k+1} \vdash \\ \Theta(\nu_2(k), \Omega'_\varphi) & = & \mathrm{cl}_k^{\psi,\Omega'_\varphi} \oplus p_{k+1} \vdash p_{k+1} \end{array}$$

We say that a characteristic term is *ground* if it does not contain index variables and clause-set symbols. Analogously to proof schemata, we define a notion of evaluation of characteristic terms:

Definition 4.1.6 (Evaluation)**.** We define the rewrite rules for clause-set symbols for all proof symbols ψ_β and cut-configurations Ω:

$$\mathrm{cl}_0^{\psi_\beta,\Omega} \to \Theta(\pi_\beta, \Omega), \qquad \mathrm{cl}_{k+1}^{\psi_\beta,\Omega} \to \Theta(\nu_\beta(k), \Omega),$$

for all $1 \leq \beta \leq \alpha$. Next, let $\gamma \in \mathbb{N}$ and let $\mathrm{cl}^{\psi_\beta,\Omega}\downarrow_\gamma$ be a normal form of $\mathrm{cl}_\gamma^{\psi_\beta,\Omega}$ under the rewrite system just given. Then define $\Theta(\psi_\beta, \Omega) = \mathrm{cl}_n^{\psi_\beta,\Omega}$ and $\Theta(\Psi, \Omega) = \Theta(\psi_1, \Omega)$ and finally the *schematic characteristic term* $\Theta(\Psi) = \Theta(\Psi, \emptyset)$.

Here comes the second reason why we chose to define the characteristic clause set via the characteristic term: The clause-set term is closed under the rewrite rules we have given for the clause-set symbols, while the notion of clause set is not (a clause will in general become a formula when subjected to the rewrite rules). Also, the representation of the characteristic clause set as a term is much more concise than an explicit representation (since the distributivity rule increases the size of the formula exponentially).

Example 4.1.7. Let us consider the proof schema and the characteristic terms from Example 4.1.3 and Example 4.1.5 respectively. Then the schematic characteristic term for Ψ, $\Theta(\Psi) = \mathrm{cl}^{\psi,\emptyset}$ and the normal forms of it for $0, 1, 2$ are the following terms:

$$\begin{aligned}
\Theta(\Psi)\downarrow_0 &= \vdash \otimes \vdash \otimes \vdash \\
\Theta(\Psi)\downarrow_1 &= \vdash p_0 \oplus (\vdash \otimes \vdash p_1) \oplus (p_0 \vdash \otimes p_1 \vdash \otimes p_1 \vdash \otimes \vdash) \\
\Theta(\Psi)\downarrow_2 &= \vdash p_0 \oplus (\vdash \otimes \vdash p_1) \oplus p_0 \vdash p_0 \oplus p_1 \vdash p_1 \oplus (p_1 \vdash \otimes \vdash p_2) \oplus \\
&\quad (p_0 \vdash \otimes p_1 \vdash \otimes p_2 \vdash \otimes p_2 \vdash \otimes \vdash)
\end{aligned}$$

and informally, for all $\gamma \in \mathbb{N}$:

$$\begin{aligned}
\Theta(\Psi)\downarrow_\gamma &= \vdash p_0 \oplus (\vdash \otimes \vdash p_1) \oplus p_0 \vdash p_0 \oplus \cdots \oplus p_\gamma \vdash p_\gamma \oplus \\
&\quad (p_1 \vdash \otimes \vdash p_2) \oplus \cdots \oplus (p_{\gamma-1} \vdash \otimes \vdash p_\gamma) \oplus \\
&\quad (p_0 \vdash \otimes \cdots \otimes p_\gamma \vdash \otimes p_\gamma \vdash \otimes \vdash)
\end{aligned}$$

Now, we prove that the notion of characteristic term is well-defined.

Proposition 4.1.1. *Let $\Psi = \langle \psi_1, \ldots, \psi_\alpha \rangle$ be a proof schema. Let $\gamma \in \mathbb{N}$ and Ω be a (cut-)configuration, then $\Theta(\psi_\beta, \Omega)\downarrow_\gamma$ is a ground characteristic term for all $1 \leq \beta \leq \alpha$. Hence $\Theta(\Psi)\downarrow_\gamma$ is a ground characteristic term.*

Proof. By double induction analogously to the proof of Proposition 3.2.1. □

Next, we show that evaluation and extraction of characteristic terms commute, i.e. the characteristic term of an evaluation of a proof schema is the same as an evaluation of the characteristic term of a proof schema. We will later use this property to derive results on schematic characteristic clause sets from standard results on (non-schematic) CERES.

Proposition 4.1.2. *Let Ψ be a proof schema, Ω a cut-configuration and $\gamma \in \mathbb{N}$. Then $\Theta(\Psi\downarrow_\gamma, \Omega) = \Theta(\Psi, \Omega)\downarrow_\gamma$.*

Proof. We proceed by induction on γ. If $\gamma = 0$, then $\Theta(\Psi\downarrow_0, \Omega) = \Theta(\pi_1, \Omega)$ and $\Theta(\Psi, \Omega)\downarrow_0 = \Theta(\pi_1, \Omega)$.

IH1: assume $\gamma > 0$ and for all $\beta < \gamma$, $\Theta(\Psi\downarrow_\beta, \Omega) = \Theta(\Psi, \Omega)\downarrow_\beta$. We proceed by induction on the number α of proof symbols in Ψ.

Let $\alpha = 1$. By the definition of characteristic term, the constructions of $\Theta(\Psi\downarrow_\gamma, \Omega)$ and $\Theta(\Psi, \Omega)\downarrow_\gamma$ differ only on proof links, i.e. if $(\psi_1(k))$ is a proof link in $\nu_1(k)$, then by the definition of evaluation of proof schemata, $\Theta(\psi_1\downarrow_\gamma, \Omega)$ contains the term $\Theta(\psi_1\downarrow_\beta, \Omega')$ and by the definition of evaluation of characteristic term schemata, $\Theta(\Psi, \Omega)\downarrow_\gamma$ contains the term $\Theta(\Psi, \Omega')\downarrow_\beta$,

for some $\beta < \gamma$. Then by the assumption $\Theta(\psi_1\downarrow_\beta, \Omega') = \Theta(\Psi, \Omega')\downarrow_\beta$ and we conclude that $\Theta(\psi_1\downarrow_\gamma, \Omega) = \Theta(\Psi, \Omega)\downarrow_\gamma$.

Now, assume $\alpha > 1$ and the proposition holds for all proof schemata with proof symbols less than α (IH2). Again, for proof links in $\nu_1(k)$ of the form $(\psi_1(k))$ the argument is the same as in the previous case. Let $(\psi_\iota(a))$, $1 < \iota \leq \alpha$, be a proof link in $\nu_1(k)$. Then, again, by the definition of evaluation of proof schemata, $\Theta(\psi_1\downarrow_\gamma, \Omega)$ contains the term $\Theta(\psi_\iota\downarrow_\lambda, \Omega')$ and by the definition of evaluation of characteristic term schemata, $\Theta(\Psi, \Omega)\downarrow_\gamma$ contains the term $\Theta(\Phi, \Omega')\downarrow_\lambda$, where $\Phi = \langle(\pi_\iota, \nu_\iota(k)), \ldots, (\pi_\alpha, \nu_\alpha(k))\rangle$. Clearly, Φ contains less than α proof symbols, then by IH2, $\Theta(\psi_\iota\downarrow_\lambda, \Omega') = \Theta(\Phi, \Omega')\downarrow_\lambda$ and we conclude that $\Theta(\psi_1\downarrow_\gamma, \Omega) = \Theta(\Psi, \Omega)\downarrow_\gamma$. □

From the characteristic term schema we finally define the notion of characteristic clause set schema in the spirit of Definition 3.3.16. We just need to ensure that clause-set symbols do not produce mutual recursion or the recursion is terminating.

We say that a clause-set symbol $\mathrm{cl}^{\psi,\Omega}$ depends on a clause-set symbol $\mathrm{cl}^{\varphi,\Omega'}$, if a term $\Theta(\psi, \Omega)$ contains $\mathrm{cl}^{\varphi,\Omega'}$. We assume that the dependency relation is transitive and reflexive. The following proposition ensures that the dependency relation is not symmetric.

Proposition 4.1.3. *Let $\Psi = \langle\psi_1, \ldots, \psi_\alpha\rangle$ be a proof schema such that for each proof symbol $\psi_\beta \in \Psi$, $\nu_\beta(k)$ contains at most one proof link to ψ_β. Then the dependency relation between clause-set symbols for all proof symbols ψ_β and cut-configurations Ω is asymmetric.*

Proof. We proceed by case distinction. Assume $\mathrm{cl}^{\psi_i,\Omega}$ depends on $\mathrm{cl}^{\psi_j,\Omega'}$ for some $i \neq j$. This means that there is a proof link in $\nu_i(k)$ to ψ_j explicitly or implicitly (i.e in $\nu_i(k)$ there is a proof link to ψ_{i_1}, in $\nu_{i_1}(k)$ there is a proof link to ψ_{i_2} and so on. Finally, in $\nu_{i_l}(k)$ there is a proof link to ψ_j). In both cases, $\mathrm{cl}^{\psi_j,\Omega'}$ cannot depend on $\mathrm{cl}^{\psi_i,\Omega}$ by the definition of proof schemata.

Now, assume there exists a number $1 \leq \beta \leq \alpha$, such that $\mathrm{cl}^{\psi_\beta,\Omega}$ depends on $\mathrm{cl}^{\psi_\beta,\Omega'}$ and vice versa. This means that the cut-configuration Ω gives rise to Ω' and vice versa. Then, since there is only one proof link in $\nu_\beta(k)$, there should exist formula schemata $A(n)$ and $B(n)$ in the end-sequent of ψ_β, such that $A(k)$ is an ancestor of $B(k+1)$ and $B(k)$ is an ancestor of $A(k+1)$. According to our definitions of formula schemata this cannot happen for $A(n) \neq B(n)$, therefore assume $A(n) = B(n)$. We distinguish two cases: if both formulas are in the same side of the sequent, then it is possible to switch the occurrences in such a way that $\Omega = \Omega'$ and $\mathrm{cl}^{\psi_\beta,\Omega} = \mathrm{cl}^{\psi_\beta,\Omega'}$; so assume they are in different sides of the sequent. The only logical connectives that

change sides of the sequent are $\neg$ and $\supset$, but by the definition of formula schemata the only iterated connectives are $\wedge$ and $\vee$. A contradiction. $\square$

If there are two or more proof links to ψ_β in $\nu_\beta(k)$, then the dependency relation is symmetric only if a cut-configuration Ω gives rise to both cut-configurations Ω and Ω' and so does Ω'. But since the structure of $\nu_\beta(k)$ is fixed, the terms $\Theta(\nu_\beta(k), \Omega)$ and $\Theta(\nu_\beta(k), \Omega')$ have the same structure as well. Therefore with the help of additional clause-set symbols, mutual recursion can be eliminated. Hence now on we assume that the dependency relation is asymmetric for an arbitrary proof schema.

Note that the mutual recursion anyway does not harm the characteristic clause set schema in this particular case, because the parameter is strictly decreasing and the rewrite system is still terminating.

Now, we perform some notational changes on characteristic terms, to get a better readable clause-set schema.

Definition 4.1.8 (Characteristic clause set schema)**.** Let $\Psi = \langle \psi_1, \ldots, \psi_\alpha \rangle$ be a proof schema and $\Theta(\Psi)$ a characteristic term schema of it. We assume an ordered list of clause-set symbols from $\Theta(\Psi)$ according to their dependency and replace each member of this list with a new clause-set symbol $\mathcal{C}_\gamma$. For each $\mathcal{C}_\gamma$ assigned to $\mathrm{cl}^{\psi_\beta,\Omega}$, for all $1 \leq \beta \leq \alpha$ and cut-configurations Ω, we define a rewrite system

$$\mathcal{R}_\gamma = \{\mathcal{C}_\gamma(0) \to \|\Theta(\pi_\beta, \Omega)\|; \quad \mathcal{C}_\gamma(k+1) \to \|\Theta(\nu_\beta(k), \Omega)\|\},$$

where $\|\cdot\|$ is defined in the following way:

- $\|\mathrm{cl}_a^{\psi,\Omega}\| = \mathcal{C}_j(a)$, where $\mathcal{C}_j$ is a clause-set symbol assigned to $\mathrm{cl}^{\psi,\Omega}$,
- $\|\Theta\| = |\Theta|$ for Θ being a characteristic term not containing clause-set symbols,
- $\|\Theta_1 \otimes \Theta_2\| = \|\Theta_1\| \otimes \|\Theta_2\|$,
- $\|\Theta_1 \oplus \Theta_2\| = \|\Theta_1\| \oplus \|\Theta_2\|$.

Clearly $\mathrm{cl}^{\psi_1,\emptyset}$ is the first element in the list and $\mathcal{C}_1$ is assigned to it. Then the *schematic characteristic clause set* $\mathrm{CL}(\Psi) = ((\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_l), \mathcal{R})$, where $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_l$. Proposition 4.1.3 and the discussion above ensures that the requirements of Definition 3.3.16 are fulfilled.

For a ground $\mathbf{LK}_s$-proof π and cut-configuration Ω, $\mathrm{CL}(\pi, \Omega) = |\Theta(\pi, \Omega)|$. We define the *standard characteristic clause set* $\mathrm{CL}(\pi) = \mathrm{CL}(\pi, \emptyset)$.

Remark 4.1.9. From the definition above and from Proposition 4.1.2 it is clear that $\mathrm{CL}(\Psi)\downarrow_\gamma = \mathrm{CL}(\Psi\downarrow_\gamma)$ for all $\gamma \in \mathbb{N}$.

Example 4.1.10. Let us consider the characteristic terms defined in Example 4.1.5. It is clear that the clause-set symbols have the following order: $\mathrm{cl}^{\psi,\emptyset}, \mathrm{cl}^{\psi,\Omega_\psi}, \mathrm{cl}^{\varphi,\Omega_\varphi}, \mathrm{cl}^{\varphi,\Omega'_\varphi}$; therefore the schematic characteristic clause set $\mathrm{CL}(\Psi) = ((\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4), \mathcal{R})$ where $\mathcal{R}$ is:

$$\begin{array}{rcl}
\mathcal{C}_1(0) & \to & \{\vdash\} \\
\mathcal{C}_1(k+1) & \to & \mathcal{C}_2(k) \oplus (\mathcal{C}_3(k+1) \otimes \{p_{k+1} \vdash\}) \\
\mathcal{C}_2(0) & \to & \{\vdash p_0 \ ; \vdash p_1\} \\
\mathcal{C}_2(k+1) & \to & \mathcal{C}_2(k) \oplus \mathcal{C}_4(k+1) \oplus \{p_{k+1} \vdash p_{k+2}\} \\
\mathcal{C}_3(0) & \to & \{p_0 \vdash\} \\
\mathcal{C}_3(k+1) & \to & \mathcal{C}_3(k) \otimes \{p_{k+1} \vdash\} \\
\mathcal{C}_4(0) & \to & \{p_0 \vdash p_0\} \\
\mathcal{C}_4(k+1) & \to & \mathcal{C}_4(k) \oplus \{p_{k+1} \vdash p_{k+1}\}
\end{array}$$

It is obvious to see that $\mathcal{C}_4$ contains only tautologies and thus can be deleted. Therefore the simplified version of $\mathrm{CL}(\Psi)$ is $((\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3), \mathcal{R})$, where $\mathcal{R}$ is:

$$\begin{array}{rcl}
\mathcal{C}_1(0) & \to & \{\vdash\} \\
\mathcal{C}_1(k+1) & \to & \mathcal{C}_2(k) \oplus (\mathcal{C}_3(k+1) \otimes \{p_{k+1} \vdash\}) \\
\mathcal{C}_2(0) & \to & \{\vdash p_0 \ ; \vdash p_1\} \\
\mathcal{C}_2(k+1) & \to & \mathcal{C}_2(k) \oplus \{p_{k+1} \vdash p_{k+2}\} \\
\mathcal{C}_3(0) & \to & \{p_0 \vdash\} \\
\mathcal{C}_3(k+1) & \to & \mathcal{C}_3(k) \otimes \{p_{k+1} \vdash\}
\end{array}$$

Let us now consider the characteristic terms defined in Example 4.1.7. The corresponding clause sets are:

$$\begin{array}{rcl}
\mathrm{CL}(\Psi{\downarrow_0}) & = & \{\vdash\} \\
\mathrm{CL}(\Psi{\downarrow_1}) & = & \{\vdash p_0 \ ; \vdash p_1 \ ; \ p_0, p_1, p_1 \vdash\} \\
\mathrm{CL}(\Psi{\downarrow_2}) & = & \{\vdash p_0 \ ; \vdash p_1 \ ; \ p_0 \vdash p_0 \ ; \ p_1 \vdash p_1 \ ; \ p_1 \vdash p_2 \ ; \ p_0, p_1, p_2, p_2 \vdash\}
\end{array}$$

Informally, for all $\gamma \in \mathbb{N}$, the simplified clause set, $\mathrm{CL}(\Psi){\downarrow_\gamma}$ is:

$$\{\vdash p_0 \ ; \vdash p_1 \ ; \ p_1 \vdash p_2 \ ; \ldots ; \ p_{\gamma-1} \vdash p_\gamma \ ; \ p_0, \ldots, p_\gamma, p_\gamma \vdash\}.$$

Now we prove the main result about the characteristic clause set and lift it to the schematic case.

Proposition 4.1.4. *Let π be a ground* $\mathbf{LK}_s$*-proof. Then* $\mathrm{CL}(\pi)$ *is unsatisfiable.*

Proof. By the identification of ground $\mathbf{LK}_s$-proofs with propositional **LK**-proofs, the result follows from the analogous proposition (Proposition 3.2) in [BL00]. □

Proposition 4.1.5. $\mathrm{CL}(\Psi)\downarrow_\gamma$ *is unsatisfiable for all* $\gamma \in \mathbb{N}$ *(i.e.* $\mathrm{CL}(\Psi)$ *is unsatisfiable).*

Proof. By the definition of characteristic clause set schema and Proposition 4.1.2, $\mathrm{CL}(\Psi)\downarrow_\gamma = \mathrm{CL}(\Psi\downarrow_\gamma)$, but the later one is unsatisfiable by Proposition 4.1.4. □

The rewrite rules from Definition 4.1.6 can be used as logical definitions. Hence any theorem prover for propositional schemata can be used to refute $\mathrm{CL}(\Psi)$. One such example is RegSTAB [ACP10], a tableaux solver for propositional schemata, which can translate tableaux proofs into resolution refutations [AP11].

Example 4.1.11. Just to complete the example, we give a resolution refutation schema of the characteristic clause set schema obtained in Example 4.1.10, hence show its unsatisfiability. The resolution proof schema can be defined as $R = ((\varrho_1, \varrho_2, \varrho_3), \mathcal{R})$, where $\mathcal{R}$ is the following rewrite system:

$$\begin{aligned}
\varrho_1(0) &\to \vdash \\
\varrho_1(k+1) &\to \varrho_3(k, p_{k+1} \vdash) \\
\varrho_2(0) &\to \vdash p_1 \\
\varrho_2(k+1) &\to r(\varrho_2(k);\ p_{k+1} \vdash p_{k+2};\ p_{k+1}) \\
\varrho_3(0, X) &\to r(\vdash p_1;\ r(\vdash p_0;\ (p_0, p_1 \vdash) \circ X;\ p_0);\ p_1), \\
\varrho_3(k+1, X) &\to r(\varrho_2(k+1);\ \varrho_3(k, (p_{k+2} \vdash) \circ X);\ p_{k+2})
\end{aligned}$$

Then a refutation of the characteristic clause set $\mathrm{CL}(\Psi)\downarrow_\alpha$ is defined by the term $\varrho_1(n)\downarrow_\alpha$ for all $\alpha \in \mathbb{N}$.

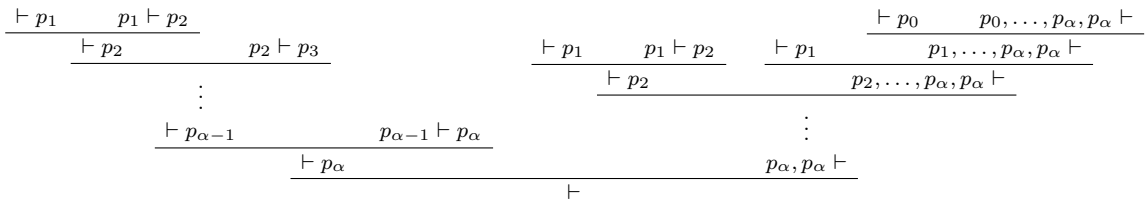
Let compute instances of the refutation schema for $\alpha = 0, 1, 2$ to illustrate that it is a refutation. For $\alpha = 0$, clearly $\varrho_1(n)\downarrow_0 = \vdash$. For $\alpha = 1$, $\varrho_1(n)\downarrow_1$ is

$$\dfrac{\vdash p_1 \qquad \dfrac{\vdash p_0 \qquad p_0, p_1, p_1 \vdash}{p_1, p_1 \vdash}}{\vdash}$$

and for $\alpha = 2$, $\varrho_1(n)\downarrow_2$ is

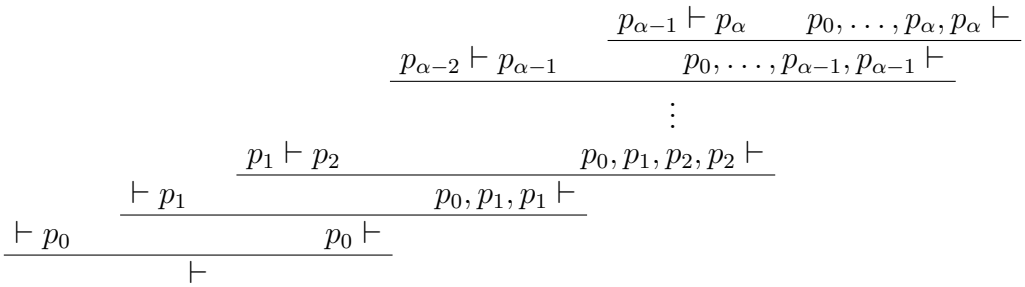
$$\dfrac{\dfrac{\vdash p_1 \qquad p_1 \vdash p_2}{\vdash p_2} \qquad \dfrac{\vdash p_1 \qquad \dfrac{\vdash p_0 \qquad p_0, p_1, p_2, p_2 \vdash}{p_1, p_2, p_2 \vdash}}{p_2, p_2 \vdash}}{\vdash}$$

Informally, for all $\alpha \in \mathbb{N}$, $\varrho_1(n)\downarrow_\alpha$ is



Clearly, all clauses at the leaf nodes are in $\mathrm{CL}(\Psi)\downarrow_\alpha$ (see the simplified clause set obtained in Example 4.1.10).

Finally, we would like to mention that there exist an exponentially smaller refutation of $\mathrm{CL}(\Psi)\downarrow_\alpha$, which is:



The refutation is not expressible in our formalism and the reason is that the recursion goes backwards from α to 0, i.e. on the base step we cannot have a resolution term on the clauses $p_{\alpha-1} \vdash p_\alpha$ and $p_0, \ldots, p_\alpha, p_\alpha \vdash$, resolving p_α.

4.2 Projection Set

The next step in the schematization of the CERES method consists in the definition of schematic proof projections. The aim is, in analogy with the preceding section, to construct a *schematic projection term* that can be evaluated to a set of ground $\mathbf{LK}_s$-proofs. As before, we introduce formal symbols representing sets of proofs; again the notion of $\mathbf{LK}_s$-proof is not closed under the rewrite rules for these symbols, which is the reason for introducing the notion of projection term.

For our term notation we assume for every rule ρ of $\mathbf{LK}_s$ a corresponding *rule symbol* that, by abuse of notation, we also denote by ρ. Given a unary rule ρ and an $\mathbf{LK}_s$-proof π, there are different ways to apply ρ to the end-sequent of π: namely, the choice of auxiliary formulas is free. Formally, the projection terms we construct will include this information so that evaluation is always well-defined, but we will surpress it in the notation since the choice of auxiliary formulas will always be clear from the context.

For every proof symbol ψ and (cut-)configuration Ω, we assume a unique proof symbol $\mathrm{pr}^{\psi,\Omega}$, called *projection term symbol*. The intended semantics of $\mathrm{pr}^{\psi,\Omega}(a)$ is "the set of projections of $\psi(a)$, with the (cut-)configuration Ω". Now, a *projection term* is a term built inductively from sequents and terms $\mathrm{pr}^{\psi,\Omega}(a)$, for some arithmetic expression a, using unary rule symbols, unary symbols $w^{\Gamma\vdash\Delta}$ for all sequents $\Gamma \vdash \Delta$ and binary symbols $\oplus, \otimes_\sigma$ for all binary rules σ.

Definition 4.2.1 (Projection term)**.** Let π be an $\mathbf{LK}_s$-proof and Ω an arbitrary (cut-)configuration for π. Let $\Gamma_\Omega, \Delta_\Omega$ and Γ_C, Δ_C be multisets of formulas corresponding to Ω- and cut-ancestors respectively and Γ, Δ be multisets of formulas not being Ω- or cut-ancestor. We define a projection term $\Xi_\rho(\pi, \Omega)$ inductively:

- If ρ corresponds to an initial sequent S, then we define $\Xi_\rho(\pi, \Omega) = S$.
- If ρ is a proof link in π of the form: $\dfrac{(\psi(a))}{\Gamma_\Omega, \Gamma_C, \Gamma \vdash \Delta_\Omega, \Delta_C, \Delta}$ then define Ω' as the cut-configuration corresponding to formula occurrences from $\Gamma_\Omega, \Gamma_C \vdash \Delta_\Omega, \Delta_C$ and $\Xi_\rho(\pi, \Omega) = \mathrm{pr}^{\psi,\Omega'}(a)$.
- If ρ is a unary inference with immediate predecessor ρ', then:
  - if the auxiliary formula(s) of ρ are Ω- or cut-ancestors, then $\Xi_\rho(\pi, \Omega) = \Xi_{\rho'}(\pi, \Omega)$,
  - otherwise $\Xi_\rho(\pi, \Omega) = \rho(\Xi_{\rho'}(\pi, \Omega))$.
- If σ is a binary inference with immediate predecessors ρ_1 and ρ_2, then:
  - if the auxiliary formulas of σ are Ω- or cut-ancestors, let $\Gamma_i \vdash \Delta_i$ be the ancestors of the end-sequent in the conclusion of ρ_i, for $i = 1, 2$, and define: $\Xi_\sigma(\pi, \Omega) = w^{\Gamma_2\vdash\Delta_2}(\Xi_{\rho_1}(\pi, \Omega)) \oplus w^{\Gamma_1\vdash\Delta_1}(\Xi_{\rho_2}(\pi, \Omega))$,
  - otherwise $\Xi_\sigma(\pi, \Omega) = \Xi_{\rho_1}(\pi, \Omega) \otimes_\sigma \Xi_{\rho_2}(\pi, \Omega)$.

Finally, define $\Xi(\pi, \Omega) = \Xi_{\rho_0}(\pi, \Omega)$, where ρ_0 is the last inference of π.

We say that a projection term is *ground* if it does not contain index variables and projection term symbols.

Example 4.2.2. Let us consider the proof schema $\Psi = \langle \psi, \varphi \rangle$ defined in Example 4.1.3: ψ is associated with the pair: π_1:

$$\dfrac{\dfrac{p_0 \vdash p_0 \qquad \dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_0 \supset p_1 \vdash p_1}\supset\colon l}{p_0, p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1}\wedge\colon r}{p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1}\, c\colon l$$

and $\nu_1(k)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{k+1} p_i}(\psi(k)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\bigwedge_{i=0}^{k+1} p_i \vdash \bigwedge_{i=0}^{k+1} p_i}(\varphi(k+1)) \qquad \dfrac{p_{k+1} \vdash p_{k+1} \qquad p_{k+2} \vdash p_{k+2}}{p_{k+1}, p_{k+1} \supset p_{k+2} \vdash p_{k+2}} \supset : l}{\bigwedge_{i=0}^{k+1} p_i, p_{k+1}, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} \wedge : r}{\bigwedge_{i=0}^{k+1} p_i, \bigwedge_{i=0}^{k} p_i, p_{k+1}, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} w : l}{\bigwedge_{i=0}^{k+1} p_i, \bigwedge_{i=0}^{k+1} p_i, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} \wedge : l}{\bigwedge_{i=0}^{k+1} p_i, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} c : l}{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}), p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i} cut}{p_0, \bigwedge_{i=0}^{k+1}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{k+2} p_i} \wedge : l$$

and φ is associated with the pair: $\pi_2 \colon p_0 \vdash p_0$ and $\nu_2(k)$:

$$\dfrac{\dfrac{\dfrac{}{\bigwedge_{i=0}^{k} p_i \vdash \bigwedge_{i=0}^{k} p_i}(\varphi(k)) \qquad p_{k+1} \vdash p_{k+1}}{\bigwedge_{i=0}^{k} p_i, p_{k+1} \vdash \bigwedge_{i=0}^{k+1} p_i} \wedge : r}{\bigwedge_{i=0}^{k+1} p_i \vdash \bigwedge_{i=0}^{k+1} p_i} \wedge : l$$

Recall the cut-configurations defined in Example 4.1.5:

$$\begin{array}{lcl} \Omega_\psi & = & \{\vdash \bigwedge_{i=0}^{n+1} p_i\} \\ \Omega_\varphi & = & \{\bigwedge_{i=0}^{n} p_i \vdash\} \\ \Omega'_\varphi & = & \{\bigwedge_{i=0}^{n} p_i \vdash \ ; \vdash \bigwedge_{i=0}^{n} p_i\} \end{array}$$

Then the projection terms of Ψ for those cut-configurations are:

$$\begin{array}{lcl} \Xi(\pi_1, \emptyset) & = & c_l(p_0 \vdash p_0 \otimes_{\wedge_r} (p_0 \vdash p_0 \otimes_{\supset_l} p_1 \vdash p_1)) \\ \Xi(\pi_1, \Omega_\psi) & = & c_l(w^{p_0, p_0 \supset p_1 \vdash}(p_0 \vdash p_0) \oplus w^{p_0 \vdash}(p_0 \vdash p_0 \otimes_{\supset_l} p_1 \vdash p_1)) \\ \Xi(\nu_1(k), \emptyset) & = & \wedge_l(w^{p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i}(\mathrm{pr}^{\psi, \Omega_\psi}(k)) \oplus \\ & & \quad w^{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}) \vdash}(\mathrm{pr}^{\varphi, \Omega_\varphi}(k+1) \otimes_{\wedge_r} \\ & & \qquad (p_{k+1} \vdash p_{k+1} \otimes_{\supset_l} p_{k+2} \vdash p_{k+2}))) \\ \Xi(\nu_1(k), \Omega_\psi) & = & \wedge_l(w^{p_{k+1} \supset p_{k+2} \vdash}(\mathrm{pr}^{\psi, \Omega_\psi}(k)) \oplus \\ & & \quad w^{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}) \vdash}(w^{p_{k+1} \supset p_{k+2} \vdash}(\mathrm{pr}^{\varphi, \Omega'_\varphi}(k+1)) \oplus \\ & & \qquad w^{\vdash}(p_{k+1} \vdash p_{k+1} \otimes_{\supset_l} p_{k+2} \vdash p_{k+2}))) \\ \Xi(\pi_2, \Omega_\varphi) & = & p_0 \vdash p_0 \\ \Xi(\pi_2, \Omega'_\varphi) & = & p_0 \vdash p_0 \\ \Xi(\nu_2(k), \Omega_\varphi) & = & \mathrm{pr}^{\psi, \Omega_\varphi}(k) \otimes_{\wedge_r} p_{k+1} \vdash p_{k+1} \\ \Xi(\nu_2(k), \Omega'_\varphi) & = & w^{\vdash}(\mathrm{pr}^{\psi, \Omega'_\varphi}(k)) \oplus w^{\vdash}(p_{k+1} \vdash p_{k+1}) \end{array}$$

We now define the evaluation of projection terms, which is compatible with the respective definition for characteristic terms.

Definition 4.2.3 (Evaluation)**.** We define the rewrite rules for projection term symbols for all proof symbols ψ_β and cut-configurations Ω:

$$\mathrm{pr}^{\psi_\beta, \Omega}(0) \to \Xi(\pi_\beta, \Omega), \qquad \mathrm{pr}^{\psi_\beta, \Omega}(k+1) \to \Xi(\nu_\beta(k), \Omega),$$

for all $1 \leq \beta \leq \alpha$. Next, let $\gamma \in \mathbb{N}$ and let $\mathrm{pr}^{\psi_\beta,\Omega}\downarrow_\gamma$ be a normal form of $\mathrm{pr}^{\psi_\beta,\Omega}(\gamma)$ under the rewrite system just given. Then define $\Xi(\psi_\beta,\Omega) = \mathrm{pr}_n^{\psi_\beta,\Omega}$ and $\Xi(\Psi,\Omega) = \Xi(\psi_1,\Omega)$ and finally the *schematic projection term* $\Xi(\Psi) = \Xi(\Psi,\emptyset)$.

Next, we prove analogous results for projection terms as we proved for characteristic terms.

Proposition 4.2.1. *Let Ψ be a proof schema, Ω a cut-configuration and $\gamma \in \mathbb{N}$. Then $\Xi(\Psi\downarrow_\gamma,\Omega) = \Xi(\Psi,\Omega)\downarrow_\gamma$.*

Proof. We proceed as in the proof of Proposition 4.1.2. □

We will define a map from ground projection terms to sets of ground $\mathbf{LK}_s$-proofs. For this, we need some auxiliary notation. The discussion regarding the notation for the application of rules from the beginning of this section applies here.

Definition 4.2.4. Let ρ be a unary and σ a binary rule. Let φ,π be $\mathbf{LK}_s$-proofs, then $\rho(\varphi)$ is the $\mathbf{LK}_s$-proof obtained from φ by applying ρ, and $\sigma(\varphi,\pi)$ is the proof obtained from the proofs φ and π by applying σ.

We extend these notions to sets of $\mathbf{LK}_s$-proofs. Let P,Q be such sets. Then $\rho(P) = \{\rho(\pi) \mid \pi \in P\}$, $P^{\Gamma\vdash\Delta} = \{\pi^{\Gamma\vdash\Delta} \mid \pi \in P\}$, where $\pi^{\Gamma\vdash\Delta}$ is π followed by weakenings adding $\Gamma \vdash \Delta$, and $P \times_\sigma Q = \{\sigma(\varphi,\pi) \mid \varphi \in P, \pi \in Q\}$.

Definition 4.2.5. Let Ξ be a ground projection term. Then we define a set of ground $\mathbf{LK}_s$-proofs $|\Xi|$ in the following way:

- $|\Gamma \vdash \Delta| = \{\Gamma \vdash \Delta\}$,
- $|\rho(\Xi)| = \rho(|\Xi|)$ for unary rule symbols ρ,
- $|w^{\Gamma\vdash\Delta}(\Xi)| = |\Xi|^{\Gamma\vdash\Delta}$,
- $|\Xi_1 \oplus \Xi_2| = |\Xi_1| \cup |\Xi_2|$,
- $|\Xi_1 \otimes_\sigma \Xi_2| = |\Xi_1| \times_\sigma |\Xi_2|$ for binary rule symbols σ.

For ground $\mathbf{LK}_s$-proofs π and cut-configurations Ω we define $\mathrm{PR}(\pi,\Omega) = |\Xi(\pi,\Omega)|$ and the *standard projection set* $\mathrm{PR}(\pi) = \mathrm{PR}(\pi,\emptyset)$. For a proof schema Ψ and $\gamma \in \mathbb{N}$ we define $\mathrm{PR}(\Psi)\downarrow_\gamma = |\Xi(\Psi)\downarrow_\gamma|$.

Example 4.2.6. Let us consider the projection terms defined in Example 4.2.2. Then the schematic projection term for Ψ, $\Xi(\Psi) = \mathrm{pr}^{\psi,\emptyset}$ and the normal forms of it for $0,1,2$ are the following terms:

$$
\begin{array}{rcl}
\Xi(\Psi)\downarrow_0 & = & c_l(p_0 \vdash p_0 \otimes_{\wedge_r} (p_0 \vdash p_0 \otimes_{\supset_l} p_1 \vdash p_1)) \\
\Xi(\Psi)\downarrow_1 & = & \wedge_l(w^{p_1 \supset p_2 \vdash} \textstyle\bigwedge_{i=0}^{2} p_i (c_l(w^{p_0, p_0 \supset p_1 \vdash}(p_0 \vdash p_0) \oplus \\
 & & \qquad w^{p_0 \vdash}(p_0 \vdash p_0 \otimes_{\supset_l} p_1 \vdash p_1))) \oplus \\
 & & \quad w^{p_0, p_0 \supset p_1 \vdash}((p_0 \vdash p_0 \otimes_{\wedge_r} p_1 \vdash p_1) \otimes_{\wedge_r} (p_1 \vdash p_1 \otimes_{\supset_l} p_2 \vdash p_2))) \\
\Xi(\Psi)\downarrow_2 & = & \wedge_l(w^{p_2 \supset p_3 \vdash} \textstyle\bigwedge_{i=0}^{3} p_i (\wedge_l(w^{p_1 \supset p_2 \vdash}(c_l(w^{p_0, p_0 \supset p_1 \vdash}(p_0 \vdash p_0) \oplus \\
 & & \qquad w^{p_0 \vdash}(p_0 \vdash p_0 \otimes_{\supset_l} p_1 \vdash p_1))) \oplus \\
 & & \quad w^{p_0, p_0 \supset p_1 \vdash}(w^{p_1 \supset p_2 \vdash}(w^{\vdash}(p_0 \vdash p_0) \oplus \\
 & & \qquad w^{\vdash}(p_1 \vdash p_1)) \oplus \\
 & & \qquad w^{\vdash}(p_1 \vdash p_1 \otimes_{\supset_l} p_2 \vdash p_2)))) \oplus \\
 & & w^{p_0, \bigwedge_{i=0}^{1}(p_i \supset p_{i+1}) \vdash}(((p_0 \vdash p_0 \otimes_{\wedge_r} p_1 \vdash p_1) \otimes_{\wedge_r} p_2 \vdash p_2) \otimes_{\wedge_r} \\
 & & \qquad (p_2 \vdash p_2 \otimes_{\supset_l} p_3 \vdash p_3)))
\end{array}
$$

Now, we compute the first two projection sets ($\mathrm{PR}(\Psi)\downarrow_2$ is already too large and therefore it is left out):

$$
\mathrm{PR}(\Psi)\downarrow_0 \;=\; \left\{ \dfrac{\dfrac{p_0 \vdash p_0 \qquad \dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_0 \supset p_1 \vdash p_1}\supset\!: l}{p_0, p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1}\wedge\!: r}{p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1}\, c\!: l \right\}
$$

$$
\mathrm{PR}(\Psi)\downarrow_1 \;=\; \left\{ \dfrac{\dfrac{\dfrac{\dfrac{p_0 \vdash p_0}{p_0, p_0, p_0 \supset p_1 \vdash p_0}\, w\!: l*}{p_0, p_0 \supset p_1 \vdash p_0}\, c\!: l}{p_0, p_0 \supset p_1, p_1 \supset p_2 \vdash p_0, \bigwedge_{i=0}^{2} p_i}\, w\!: l, r}{p_0, \bigwedge_{i=0}^{1}(p_i \supset p_{i+1}) \vdash p_0, \bigwedge_{i=0}^{2} p_i}\wedge\!: l \right\} ;
$$

$$
\left\{ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_0 \supset p_1 \vdash p_1}\supset\!: l}{p_0, p_0 \supset p_1 \vdash p_1}\, w\!: l}{p_0, p_0, p_0 \supset p_1 \vdash p_1}\, c\!: l}{p_0, p_0 \supset p_1 \vdash p_1}\, w\!: l, r}{p_0, p_0 \supset p_1, p_1 \supset p_2 \vdash p_1, \bigwedge_{i=0}^{2} p_i}\wedge\!: l}{p_0, \bigwedge_{i=0}^{1}(p_i \supset p_{i+1}) \vdash p_1, \bigwedge_{i=0}^{2} p_i} \right\} ;
$$

$$
\left\{ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_1 \vdash p_0 \wedge p_1}\wedge\!: r \qquad \dfrac{p_1 \vdash p_2 \qquad p_1 \vdash p_2}{p_1, p_1 \supset p_2 \vdash p_2}\supset\!: l}{p_0, p_1, p_1, p_1 \supset p_2 \vdash \bigwedge_{i=0}^{2} p_i}\wedge\!: r}{p_0, p_1, p_1, p_1 \supset p_2 \vdash \bigwedge_{i=0}^{2} p_i}\, w\!: l*}{p_0, p_1, p_1, p_0, p_0 \supset p_1, p_1 \supset p_2 \vdash \bigwedge_{i=0}^{2} p_i}\wedge\!: l}{p_0, p_1, p_1, p_0, \bigwedge_{i=0}^{1}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{2} p_i} \right\}
$$

Recall the clause sets:

$$
\mathrm{CL}(\Psi)\downarrow_0 = \{\vdash\} \quad \text{and} \quad \mathrm{CL}(\Psi)\downarrow_1 = \{\vdash p_0 \,;\, \vdash p_1 \,;\, p_0, p_1, p_1 \vdash\}.
$$

It is easy to see that the projection from $\mathrm{PR}(\Psi)\downarrow_0$ corresponds to the empty clause in $\mathrm{CL}(\Psi)\downarrow_0$ and the projections from $\mathrm{PR}(\Psi)\downarrow_1$ correspond to the clauses in $\mathrm{CL}(\Psi)\downarrow_1$ respectively.

From the example above, we can see that for each clause from the characteristic clause set, there is a corresponding proof in the projection set. This is a general property and not only specific to this example. Hence, the following result describes this relation between the standard projection set and characteristic clause set in the ground case. It will allow us to construct, together with a resolution refutation of $\mathrm{CL}(\Psi)$, essentially cut-free proofs of $S(\gamma)$ for all $\gamma \in \mathbb{N}$. Finally, the result is lifted to the schematic case.

Proposition 4.2.2. *Let π be a ground $\mathbf{LK}_s$-proof with end-sequent S, then for all clauses $C \in \mathrm{CL}(\pi)$, there exists a ground $\mathbf{LK}_s$-proof $\pi \in \mathrm{PR}(\pi)$ with end-sequent $S \circ C$.*

Proof. By the identification of ground $\mathbf{LK}_s$-proofs with propositional **LK**-proofs, the result follows from the Definition 4.2.5 and from the analogous result (Lemma 3.1) in [BL00]. □

Proposition 4.2.3. *Let Ψ be a proof schema and $\gamma \in \mathbb{N}$, then $\mathrm{PR}(\Psi\downarrow_\gamma) = \mathrm{PR}(\Psi)\downarrow_\gamma$.*

Proof. The result follows directly from Proposition 4.2.1. □

Proposition 4.2.4. *Let Ψ be a proof schema with end-sequent $S(n)$ and $\gamma \in \mathbb{N}$. Then for every clause $C \in \mathrm{CL}(\Psi)\downarrow_\gamma$ there exists a ground $\mathbf{LK}_s$-proof $\pi \in \mathrm{PR}(\Psi)\downarrow_\gamma$ with end-sequent $C \circ S(\gamma)$.*

Proof. By Proposition 4.1.2, $\mathrm{CL}(\Psi)\downarrow_\gamma = \mathrm{CL}(\Psi\downarrow_\gamma)$, and by Proposition 4.2.3, $\mathrm{PR}(\Psi)\downarrow_\gamma = \mathrm{PR}(\Psi\downarrow_\gamma)$. Then the result follows from Proposition 4.2.2, since $\Psi\downarrow_\gamma$ has end-sequent $S(\gamma)$ by definition. □

4.3 Atomic Cut Normal Form

While the ACNF in ordinary CERES method was an **LK**-proof, in this case the ACNF schema is a pair of projection term and resolution refutation schema. When a concrete number is given, then the corresponding ground $\mathbf{LK}_s$-proof is computed with atomic cuts only.

Definition 4.3.1 (Transformation)**.** Let ϱ be a ground resolution refutation. Then the transformation $T(\varrho)$ is defined inductively:

- if $\varrho = C$ for a clause C, then $T(\varrho) = C$,
- if $\varrho = r(\varrho_1, \varrho_2, P)$, then $T(\varrho)$ is:

$$\dfrac{\dfrac{T(\varrho_1)}{\dfrac{\Gamma \vdash \Delta, P, \ldots, P}{\Gamma \vdash \Delta, P}\, c\colon r*} \quad \dfrac{T(\varrho_2)}{\dfrac{P, \ldots, P, \Pi \vdash \Lambda}{P, \Pi \vdash \Lambda}\, c\colon l*}}{\Gamma, \Pi \vdash \Delta, \Lambda}\, cut$$

Definition 4.3.2 (ACNF)**.** Let Ψ be a proof schema of the end-sequent $S(n)$, $\mathrm{PR}(\Psi)$ be its projection term schema and R be a resolution refutation schema of the characteristic clause set schema $\mathrm{CL}(\Psi)$. Then a pair $(\mathrm{PR}(\Psi), R)$ is an Atomic Cut Normal Form (shortly ACNF) schema of Ψ.

The ACNF for an arbitrary $\alpha \in \mathbb{N}$ is defined in the following way: for all $C \in \mathrm{CL}(\Psi)\downarrow_\alpha$, C being a leaf node in $T(R\downarrow_\alpha)$, we take a corresponding proof projection $\phi_C \in \mathrm{PR}(\Psi)\downarrow_\alpha$. Then define $\mathbf{LK}_s$-proof π_α of the end-sequent $S(\alpha)$ by replacing all initial sequents C in $T(R\downarrow_\alpha)$ by ϕ_C and adding necessary contractions.

Example 4.3.3. Let consider the proof schema Ψ defined in Example 4.1.3: ψ is associated with the pair: π_1:

$$\dfrac{\dfrac{p_0 \vdash p_0 \qquad \dfrac{p_0 \vdash p_0 \qquad p_1 \vdash p_1}{p_0, p_0 \supset p_1 \vdash p_1}\supset\colon l}{p_0, p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1}\wedge\colon r}{p_0, p_0 \supset p_1 \vdash p_0 \wedge p_1}\, c\colon l$$

and $\nu_1(k)$:

$$\dfrac{\dfrac{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{k+1} p_i \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\varphi(k+1))}{\bigwedge_{i=0}^{k+1} p_i \vdash \bigwedge_{i=0}^{k+1} p_i} \qquad \dfrac{p_{k+1} \vdash p_{k+1} \qquad p_{k+2} \vdash p_{k+2}}{p_{k+1}, p_{k+1} \supset p_{k+2} \vdash p_{k+2}}\supset\colon l}{\bigwedge_{i=0}^{k+1} p_i, p_{k+1}, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i}\wedge\colon r}{\bigwedge_{i=0}^{k+1} p_i, \bigwedge_{i=0}^{k} p_i, p_{k+1}, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i}\, w\colon l}{\bigwedge_{i=0}^{k+1} p_i, \bigwedge_{i=0}^{k+1} p_i, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i}\wedge\colon l}{\bigwedge_{i=0}^{k+1} p_i, p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i}\, c\colon l}{p_0, \bigwedge_{i=0}^{k}(p_i \supset p_{i+1}), p_{k+1} \supset p_{k+2} \vdash \bigwedge_{i=0}^{k+2} p_i}\, cut}{p_0, \bigwedge_{i=0}^{k+1}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{k+2} p_i}\wedge\colon l$$

and φ is associated with the pair: $\pi_2\colon p_0 \vdash p_0$ and $\nu_2(k)$:

$$\dfrac{\dfrac{\dfrac{(\varphi(k))}{\bigwedge_{i=0}^{k} p_i \vdash \bigwedge_{i=0}^{k} p_i} \qquad p_{k+1} \vdash p_{k+1}}{\bigwedge_{i=0}^{k} p_i, p_{k+1} \vdash \bigwedge_{i=0}^{k+1} p_i}\wedge\colon r}{\bigwedge_{i=0}^{k+1} p_i \vdash \bigwedge_{i=0}^{k+1} p_i}\wedge\colon l$$

Recall its resolution refutation schema from Example 4.1.11: $R = ((\varrho_1, \varrho_2, \varrho_3), \mathcal{R})$,

where $\mathcal{R}$ is the following rewrite system:

$$\begin{array}{rcl}
\varrho_1(0) & \to & \vdash \\
\varrho_1(k+1) & \to & \varrho_3(k, p_{k+1} \vdash) \\
\varrho_2(0) & \to & \vdash p_1 \\
\varrho_2(k+1) & \to & r(\varrho_2(k);\ p_{k+1} \vdash p_{k+2};\ p_{k+1}) \\
\varrho_3(0, X) & \to & r(\vdash p_1;\ r(\vdash p_0;\ (p_0, p_1 \vdash) \circ X;\ p_0);\ p_1), \\
\varrho_3(k+1, X) & \to & r(\varrho_2(k+1);\ \varrho_3(k, (p_{k+2} \vdash) \circ X);\ p_{k+2})
\end{array}$$

and the projection sets from Example 4.2.6. Then the atomic cut normal form of Ψ for $\alpha = 0$ is simply π_1, since the resolution refutation is $\vdash$ and the empty clause is replaced by the corresponding projection (which is π_1) from $\mathrm{PR}(\Psi\downarrow_0)$.

Let now compute the ACNF of Ψ for $\alpha = 1$. First we compute $T(\varrho_1(n)\downarrow_1)$, which is:

$$\dfrac{\dfrac{\vdash p_0 \qquad p_0, p_1, p_1 \vdash}{p_1, p_1 \vdash}\, cut \qquad \dfrac{}{p_1 \vdash}\, c\colon l}{\vdash}\, cut$$

Next, we denote projections from $\mathrm{PR}(\Psi)\downarrow_1$ by $\phi_{\vdash p_0}$, $\phi_{\vdash p_1}$ and $\phi_{p_0,p_1,p_1\vdash}$, the formulas $\bigwedge_{i=0}^{1}(p_i \supset p_{i+1})$ and $\bigwedge_{i=0}^{2} p_i$ by A and B respectively. Then append these projections in the proof skeleton above and add some contractions at the end; so we get:

$$\dfrac{\dfrac{\dfrac{(\phi_{\vdash p_1})}{p_0, A \vdash B, p_1} \qquad \dfrac{\dfrac{(\phi_{\vdash p_0})}{p_0, A \vdash B, p_0} \qquad \dfrac{(\phi_{p_0,p_1,p_1\vdash})}{p_0, p_1, p_1, p_0, A \vdash B}\, cut}{p_1, p_1, p_0, A, p_0, A \vdash B, B}\, c\colon l}{p_1, p_0, A, p_0, A \vdash B, B}\, cut}{p_0, A, p_0, A, p_0, A \vdash B, B, B}\, c\colon l, r*$$

$$p_0, \textstyle\bigwedge_{i=0}^{1}(p_i \supset p_{i+1}) \vdash \bigwedge_{i=0}^{2} p_i$$

This completes our example.

Note the difference to the straightforward method: first computing the instance $\Psi\downarrow_\alpha$ and then using ordinary CERES on it. In schematic method, $\Psi\downarrow_\alpha$ is not computed at all, however uniform representation of the sequence of cut-free proofs is obtained. The later is not possible by the straightforward method.

Finally, we can summarize the CERES method of cut-elimination for proof schemata and give the main result. The whole procedure *CERES*$_s$ on schemata is defined in the following way: let Ψ be a proof schema, then we distinguish two phases – schematic construction and evaluation.

Schematic construction. This phase consists of the following steps:

- compute $\mathrm{CL}(\Psi)$;
- compute $\mathrm{PR}(\Psi)$;
- construct a resolution refutation schema R of $\mathrm{CL}(\Psi)$.

Evaluation. Given a natural number α, this phase consists of the following steps:

- compute $\mathrm{PR}(\Psi)\downarrow_\alpha$;
- compute $R\downarrow_\alpha$ and $T(R\downarrow_\alpha)$;
- append the corresponding projections in $\mathrm{PR}(\Psi)\downarrow_\alpha$ to $T(R\downarrow_\alpha)$ and propagate the contexts down in the proof. Finally add contractions if necessary.

Definition 4.3.4 (Proof length)**.** Let π be an $\mathbf{LK}_s$-proof. The length of π, denoted by $l(\pi)$, is the number of sequents occurring in π. For a set of $\mathbf{LK}_s$-proofs P, $l(P) = \max\{l(\pi) \mid \pi \in P\}$.

Clearly, l can be trivially extended to ground resolution terms, therefore below it is also used as a measure for them.

Theorem 4.3.1. *Let Ψ be a proof schema with end-sequent $S(n)$. Then the evaluation phase of $CERES_s$ produces for all $\alpha \in \mathbb{N}$ a ground $\mathbf{LK}_s$-proof π_α of $S(\alpha)$ with at most atomic cuts such that its size $l(\pi_\alpha)$ is polynomial in $l(R\downarrow_\alpha) \cdot l(\mathrm{PR}(\Psi)\downarrow_\alpha)$.*

Proof. Let $\alpha \in \mathbb{N}$. By Proposition 4.2.3 we obtain for any clause in $\mathrm{CL}(\Psi)\downarrow_\alpha$ a corresponding projection in $\mathrm{PR}(\Psi)\downarrow_\alpha$. Let R be a resolution refutation schema for $\mathrm{CL}(\Psi)$ constructed in the schematic construction phase of $CERES_s$ and $T(R\downarrow_\alpha)$ the corresponding tree. Clearly the length of any projection is at most $l(\mathrm{PR}(\Psi)\downarrow_\alpha)$ and $l(T(R\downarrow_\alpha))$ is polynomial in $l(R\downarrow_\alpha)$. Therefore, $l(\pi_\alpha)$ is polynomial in $l(R\downarrow_\alpha) \cdot l(\mathrm{PR}(\Psi)\downarrow_\alpha)$. Moreover, the resulting proof π_α of $S(\alpha)$ obtained in the last step of evaluation phase contains at most atomic cuts. □

Chapter 5

Extensions to First-Order Schemata

Before defining the language and the method extensions formally, we start with a motivation example, where cut-elimination is not possible. First recall the discussion from the introduction. We need to define a proof, where the induction rule will produce a strong quantifier and then this formula will be cut out.

Let us consider Peano arithmetic (PA), defined in [Tak87], with the induction rule:

$$\frac{\Gamma, A(\alpha) \vdash \Delta, A(s(\alpha))}{A(\bar{0}), \Gamma \vdash \Delta, A(t)}$$

where α is an eigenvariable not occurring in $A(\bar{0}), \Gamma, \Delta$; $A(\alpha)$ is called the *induction invariant*. This rule can simulate the binary induction rule

$$\frac{\Gamma \vdash \Delta, A(\bar{0}) \qquad \Pi, A(\alpha) \vdash \Lambda, A(s(\alpha))}{\Gamma, \Pi \vdash \Delta, \Lambda, A(t)}\, ind$$

with additional cut(s), but we want to avoid as many cuts as possible. Therefore we consider the *ind* rule as a part of PA.

To avoid equality reasoning in **LK**-proofs, we admit atomic equality axioms of the form $A(s), t = s \vdash A(t)$ and $\vdash s = s$.

Let us consider the sequent S:

$$Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash$$
$$(\forall n)((P(\hat{f}(n,c)) \supset P(g(n,c))) \supset (P(c) \supset P(g(n,c))))$$

where g is a binary function symbol, f is a unary one and $\hat{f}$ a binary function symbol s.t.

$$Def(\hat{f}):\ (\forall x)\hat{f}(\bar{0}, x) = x,\ (\forall n)(\forall x)\hat{f}(s(n), x) = f(\hat{f}(n, x))$$

Obviously, no Herbrand sequent exists for S, therefore it cannot be proven without induction; some inductive lemma is needed which will prove something like $(\forall n)(\forall x)(P(x) \supset P(\hat{f}(n,x)))$ or a more general statement. We use the following lemma:

$$Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall n)(\forall x)(P(x) \supset P(\hat{f}(n,x)))$$

A proof ψ of this inductive lemma could be:

$$\dfrac{\dfrac{(\psi_1) \qquad\qquad (\psi_2)}{(\forall x)\hat{f}(\bar{0},x) = x \vdash (\forall x)(P(x) \supset P(\hat{f}(\bar{0},x))) \qquad \Gamma, (\forall x)(P(x) \supset P(\hat{f}(\alpha,x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(s(\alpha),x)))}\, ind}{\dfrac{Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(\gamma,x)))}{Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall n)(\forall x)(P(x) \supset P(\hat{f}(n,x)))}\, \forall : r}$$

where Γ is $(\forall n)(\forall x)\hat{f}(s(n),x) = f(\hat{f}(n,x)), (\forall x)(P(x) \supset P(f(x)))$; ψ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(u), \hat{f}(\bar{0},u) = u \vdash P(\hat{f}(\bar{0},u))}{\hat{f}(\bar{0},u) = u \vdash P(u) \supset P(\hat{f}(\bar{0},u))}\, \supset : r}{(\forall x)\hat{f}(\bar{0},x) = x \vdash P(u) \supset P(\hat{f}(\bar{0},u))}\, \forall : l}{(\forall x)\hat{f}(\bar{0},x) = x \vdash (\forall x)(P(x) \supset P(\hat{f}(\bar{0},x)))}\, \forall : r$$

and ψ_2 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(\alpha,u)) \vdash P(\hat{f}(\alpha,u)) \qquad P(f(\hat{f}(\alpha,u))), \hat{f}(s(\alpha),u) = f(\hat{f}(\alpha,u)) \vdash P(\hat{f}(s(\alpha),u))}{P(\hat{f}(\alpha,u)) \supset P(f(\hat{f}(\alpha,u))), \hat{f}(s(\alpha),u) = f(\hat{f}(\alpha,u)), P(\hat{f}(\alpha,u)) \vdash P(\hat{f}(s(\alpha),u))}\, \supset : l}{P(u) \vdash P(u) \qquad \Gamma, P(\hat{f}(\alpha,u)) \vdash P(\hat{f}(s(\alpha),u))}\, \forall : l*}{P(u), \Gamma, P(u) \supset P(\hat{f}(\alpha,u)) \vdash P(\hat{f}(s(\alpha),u))}\, \supset : l}{\Gamma, P(u) \supset P(\hat{f}(\alpha,u)) \vdash P(u) \supset P(\hat{f}(s(\alpha),u))}\, \supset : r}{\Gamma, (\forall x)(P(x) \supset P(\hat{f}(\alpha,x))) \vdash P(u) \supset P(\hat{f}(s(\alpha),u)))}\, \forall : l}{\Gamma, (\forall x)(P(x) \supset P(\hat{f}(\alpha,x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(s(\alpha),x)))}\, \forall : r$$

Finally, we define φ as (the cut-formula $(\forall n)(\forall x)(P(x) \supset P(\hat{f}(n,x)))$ is denoted with C):

$$\dfrac{\dfrac{(\psi) \qquad\qquad (\chi)}{Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash C \qquad C \vdash (\forall n)((P(\hat{f}(n,c)) \supset P(g(n,c))) \supset (P(c) \supset P(g(n,c))))}\, cut}{Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall n)((P(\hat{f}(n,c)) \supset P(g(n,c))) \supset (P(c) \supset P(g(n,c))))}$$

where χ is an induction-free proof of the form:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(\beta,c)) \vdash P(\hat{f}(\beta,c)) \qquad P(g(\beta,c)) \vdash P(g(\beta,c))}{P(\hat{f}(\beta,c)) \supset P(g(\beta,c)), P(\hat{f}(\beta,c)) \vdash P(g(\beta,c))}\, \supset : l}{P(c) \vdash P(c) \qquad P(\hat{f}(\beta,c)) \supset P(g(\beta,c)), P(\hat{f}(\beta,c)) \vdash P(g(\beta,c))}\, \supset : l}{P(c), P(\hat{f}(\beta,c)) \supset P(g(\beta,c)), P(c) \supset P(\hat{f}(\beta,c)) \vdash P(g(\beta,c))}\, \supset : r}{P(\hat{f}(\beta,c)) \supset P(g(\beta,c)), P(c) \supset P(\hat{f}(\beta,c)) \vdash P(c) \supset P(g(\beta,c))}\, \supset : r}{P(c) \supset P(\hat{f}(\beta,c)) \vdash (P(\hat{f}(\beta,c)) \supset P(g(\beta,c))) \supset (P(c) \supset P(g(\beta,c)))}\, \forall : l*}{(\forall n)(\forall x)(P(x) \supset P(\hat{f}(n,x))) \vdash (P(\hat{f}(\beta,c)) \supset P(g(\beta,c))) \supset (P(c) \supset P(g(\beta,c)))}\, \forall : r}{(\forall n)(\forall x)(P(x) \supset P(\hat{f}(n,x))) \vdash (\forall n)((P(\hat{f}(n,c)) \supset P(g(n,c))) \supset (P(c) \supset P(g(n,c))))}$$

In the attempt of performing reductive cut-elimination, we locate the place in the proof φ, where $(\forall n)$ is introduced. We can apply the corresponding transformation rules from Appendix A which will yield

$$\dfrac{\dfrac{\overset{(\psi')}{\Gamma \vdash (\forall x)(P(x) \supset P(\hat{f}(\beta,x)))} \qquad \overset{(\chi')}{(\forall x)(P(x) \supset P(\hat{f}(\beta,x))) \vdash (P(\hat{f}(\beta,c)) \supset P(g(\beta,c))) \supset (P(c) \supset P(g(\beta,c)))}}{Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash (P(\hat{f}(\beta,c)) \supset P(g(\beta,c))) \supset (P(c) \supset P(g(\beta,c))))}\, cut}{Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall n)((P(\hat{f}(n,c)) \supset P(g(n,c))) \supset (P(c) \supset P(g(n,c))))}\, \forall\colon r$$

where χ' is obtained from χ by removing the last two inferences $\forall\colon l$ and $\forall\colon r$; ψ' is obtained from ψ by deleting the last inference $\forall\colon r$ and replacing γ by β. But now we get stuck, as we cannot "cross" the *ind* rule. Neither can the *ind* rule be eliminated as β is variable. In fact, if we had instead $(\forall x)(P(x) \supset P(\hat{f}(t,x)))$ for a closed term t over $\{\bar{0}, s, +, *\}$ we could prove $\mathrm{PA} \vdash t = \bar{n}$ and also

$$Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(\bar{n},x)))$$

without induction (by iterated cuts) and cut-elimination would proceed.

This problem, however, is neither rooted in the specific form of ψ nor the *ind* rule. In fact, there exists no proof of S with only atomic cuts. In particular, induction on the formula $(\forall n)((P(\hat{f}(n,c)) \supset P(g(n,c))) \supset (P(c) \supset P(g(n,c))))$ fails. In order to prove the end-sequent an inductive lemma is needed (something which implies $(\forall n)(\forall x)(P(x) \supset P(\hat{f}(n,x)))$) and cannot be eliminated.

While there is no proof of S in PA with only atomic cuts, the sequents S_n:

$$Def(\hat{f}), (\forall x)(P(x) \supset P(f(x))) \vdash \\ (P(\hat{f}(\bar{n},c)) \supset P(g(\bar{n},c))) \supset (P(c) \supset P(g(\bar{n},c)))$$

do have such proofs for all n; but instead of a unique proof φ of S we get an infinite sequence of proofs φ_n of S_n, which have (essential) cut-free versions φ'_n. This kind of "infinitary" cut-elimination only makes sense if there exists a uniform representation of the sequence of proofs φ'_n. As we have seen in the previous chapter, the *CERES*$_s$ method has the potential of producing such a uniform representation. Below we extend our definitions to handle first-order proofs, thus paving the way for cut-elimination in the presence of induction.

5.1 Schematic First-Order Language

We define a *schematic first-order language*, a formal language that allows the specification of an (infinite) sequence of first-order terms and formulas

by a finite expression. Towards this, we introduce two types ω and ι, such that ω is a type representing natural numbers and ι is a type representing an arbitrary first-order domain. We extend our language with countable sets of *variables* of type ι, n-ary function symbols of type $\tau_1 \times \cdots \times \tau_n \to \tau$ and n-ary predicate symbols. We assume that function symbols are partitioned into *constant function symbols* and *defined function symbols*. The first set will contain the usual uninterpreted function symbols and the second will allow primitive recursively defined functions in the language. Constant function symbols can have arity 0, but for defined function symbols we assume the type $\omega \times \tau_1 \times \cdots \times \tau_n \to \tau$; therefore they do have arity ≥ 1.

Definition 5.1.1 (Terms)**.** Variables are terms and if f is an n-ary constant function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term. By $V(t)$ we denote the set of variables of a term t.

Note that arithmetic expressions, defined in Chapter 3, are of type ω. Henceforth we consider arithmetic expressions as a subset of the set of terms.

Definition 5.1.2 (Term schemata)**.** Term schemata are defined recursively as follows:

- Terms are terms schemata
- For every $n+1$-ary defined function symbol $\hat{f}$, we assume given two rewrite rules

$$\begin{aligned} \hat{f}(0, x_1, \ldots, x_n) &\to s, \\ \hat{f}(k+1, x_1, \ldots, x_n) &\to t[\hat{f}(k, x_1, \ldots, x_n)] \end{aligned}$$

  such that $V(s) \cup V(t) = \{x_1, \ldots, x_n\}$ and s, t are terms. The expression $t[\hat{f}(k, x_1, \ldots, x_n)]$ denotes a term t with some occurrences of $\hat{f}(k, x_1, \ldots, x_n)$. If $t_1, \ldots, t_n$ are term schemata and a an arithmetic expression, then $f(a, t_1, \ldots, t_n)$, is a term schema.

To distinguish between terms and term schemata, we denote constant function symbols with $f, g, \ldots$ and defined function symbols with $\hat{f}, \hat{g}, \ldots$.

Example 5.1.3. Let f be a unary constant function symbol, $\hat{f}$ be a binary defined function symbol and

$$\begin{aligned} \hat{f}(0, x) &\to x, \\ \hat{f}(k+1, x) &\to f(\hat{f}(k, x)). \end{aligned}$$

Then $\hat{f}(n, x)$ is a term schema representing $f^n(x)$. Note that the rewrite rules for $\hat{f}$ correspond to $Def(\hat{f})$ from the example given in the beginning of this chapter.

Remark 5.1.4. It is obvious that the usual primitive recursive definition of $*$, exponentiation and the like can be represented in our system.

Proposition 5.1.1. *Let t be a term schema. Then primitive recursion defined as rewrite rules for t, is strongly normalizing and confluent.*

Proof. Trivial, since all definitions are primitive recursive. □

We now turn to the definition of schematic formulas. There are two possibilities: one is to directly extend our notion of formula schemata with quantifiers and other is to define formula schemata in primitive recursive fashion as we did for term schemata and as it is done in [DLRW]. The difference between these two languages is that the language in [DLRW] allows iterated $\neg$ and $\supset$. But this breaks Proposition 4.1.3. Therefore we choose the first approach here. Note that this does not affect the expressive power: iterated $\neg$ or $\supset$ can be encoded by auxiliary predicate symbols (see [AEP]).

Definition 5.1.5 (Atom formula schemata)**.** Let P be an n-ary predicate symbol and $t_1, \ldots, t_n$ be term schemata, then $P(t_1, \ldots, t_n)$ is an atom formula schema.

Remark 5.1.6. Note that for a unary predicate symbol P and an arithmetic expression a, $P(a)$ corresponds to an indexed predicate p_a.

Definition 5.1.7 (Formula schemata)**.** We redefine formula schemata in the following way:

- An atom formula schema is a formula schema.
- If A and B are formula schemata, then so are $\neg A$, $A \vee B$, $A \wedge B$, $A \supset B$, $(\forall x)A$ and $(\exists x)A$ for x being a variable of type ι.
- If A is a formula schema, $a\colon \omega, b\colon \omega$ are arithmetic expressions and $i\colon \omega$ is an index variable not bound in A, then $\bigwedge_{i=a}^{b} A$ and $\bigvee_{i=a}^{b} A$ are formula schemata such that i is bound in both formula schemata.

Remark 5.1.8. Note that formula schemata are schematic only w.r.t term schemata and the connectives $\bigwedge, \bigvee$.

We call a formula schema *open* if there is a free variable of type ι in it, otherwise it is called *closed*. For an open formula schema A with free variables $x_1, \ldots, x_n$, we write $A(x_1, \ldots, x_n)$.

We define first-order substitution (shortly *fo-substitution*) as usual, mapping variables of type ι to terms. Then for a formula schema $A(x_1, \ldots, x_n)$ and fo-substitution $\sigma = [x_1/t_1, \ldots, x_n/t_n]$, $A(x_1, \ldots, x_n)\sigma = A(t_1, \ldots, t_n)$.

According to the definition above it is clear that we do not allow quantification over variables of type ω. Although it may happen that different quantifiers bind the same variable, it always will be clear from the context which quantifier binds which variable. We illustrate that with the following example.

Example 5.1.9. Let us consider a formula schema $(\exists y)(\bigvee_{i=0}^{n}(\forall x)A(i,x,y))$. Here, $i\colon \omega, x\colon \iota, y\colon \iota$ are bound variables. Clearly, it represents the formula $(\exists y)((\forall x)A(0,x,y)\vee\cdots\vee(\forall x)A(n,x,y))$ and by renaming of bound variables, this formula is equivalent to $(\exists y)((\forall x_0)A(0,x_0,y)\vee\cdots\vee(\forall x_n)A(n,x_n,y))$.

The semantics of formula schemata is extended to first-order in an obvious way. It is clear that the validity problem of first-order formula schemata is undecidable, therefore we do not restrict arithmetic expressions to be linear, but we keep the restriction that at most one parameter is allowed in a formula schema.

5.2 Extension of $\mathbf{LK}_s$

Having extended the notion of formula schemata, we adapt the calculus $\mathbf{LK}_s$ to first-order schemata. The main difference between propositional and first-order proof schemata are free variables, therefore we should indicate them in sequents and proof links. The motivation is the following: imagine there is a proof ψ of an open sequent $S_1(n,\bar{x})$ for a variable vector $\bar{x}$ and a proof φ needs $S_1(n,\bar{t})$ and $S_1(n,\bar{s})$, for some term vectors $\bar{t}$ and $\bar{s}$, as axioms to prove a sequent S. Therefore in φ there should be some proof links to $\psi[\bar{x}/\bar{t}]$ and $\psi[\bar{x}/\bar{s}]$. So we come up with the following definition.

Definition 5.2.1 (Proof link)**.** If φ is a proof symbol, a is an arithmetic expression, and $S(n,x_1,\ldots,x_l)$ an open sequent schema with a parameter n and free variables $x_1,\ldots,x_l$, then the expression $\dfrac{(\varphi(a,t_1,\ldots,t_l))}{S(a,t_1,\ldots,t_l)}$ is called a *proof link*, for $t_1,\ldots,t_l$ being term schemata not containing parameters different from the one in a.

Except quantifier introduction rules, we need so called definition rules, which will operate on formulas allowing us to rewrite term schemata according their rewrite rules.

Definition 5.2.2 (Calculus $\mathbf{LK}_s$)**.** We extend the calculus $\mathbf{LK}_s$ with the following rules:

- $\forall$ introduction

$$\frac{A(t), \Gamma \vdash \Delta}{(\forall x)A(x), \Gamma \vdash \Delta} \forall\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A(u)}{\Gamma \vdash \Delta, (\forall x)A(x)} \forall\colon r$$

where t is an arbitrary term schema of type ι and $u\colon \iota$ is a free variable not occurring in the lower sequent; u is called an *eigenvariable* and the condition, that it should not occur in the lower sequent is called the *eigenvariable condition*. The $\forall\colon l$ rule is called a weak quantifier rule and the $\forall\colon r$ rule is called a strong quantifier rule.

- $\exists$ introduction

$$\frac{A(u), \Gamma \vdash \Delta}{(\exists x)A(x), \Gamma \vdash \Delta} \exists\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, (\exists x)A(x)} \exists\colon r$$

where t is an arbitrary term schema of type ι and $u\colon \iota$ is an eigenvariable satisfying the eigenvariable condition. The $\exists\colon l$ rule is called a strong quantifier rule and the $\exists\colon r$ rule is called a weak quantifier rule.

- definition rules

$$\frac{A(t), \Gamma \vdash \Delta}{A(s), \Gamma \vdash \Delta}\, def\colon l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, A(s)}\, def\colon r$$

where t, s are term schemata and there exists a rewrite rule $t \to s$.

The definitions of $\mathbf{LK}_s$-proof, proof schema and the like stay the same. Since we changed the definition of proof link, we should change the rewrite rules for them accordingly.

Definition 5.2.3 (Evaluation of proof schemata)**.** Let $\Psi = \langle \psi_1, \ldots, \psi_\alpha \rangle$ be a proof schema. We define the rewrite rules for proof links

$$\underset{S}{(\psi_\beta(0, t_1, \ldots, t_l))} \to \pi_\beta\theta, \qquad \text{and} \qquad \underset{S}{(\psi_\beta(k+1, t_1, \ldots, t_l))} \to \nu_\beta(k)\theta$$

for all $1 \leq \beta \leq \alpha$ and $\theta = \{x_1/t_1, \ldots, x_l/t_l\}$.

Now, for $\gamma \in \mathbb{N}$ we define $\psi_\beta{\downarrow_\gamma}$ as a normal form of $\underset{S}{(\psi_\beta(\gamma, t_1, \ldots, t_l))}$ under the rewrite system just given extended with rewrite rules for defined function symbols. Note that after rewriting, the $def\colon l$ and $def\colon r$ rules become trivial rules, i.e. upper and lower sequents become equal. Therefore these rules can be skipped in $\psi_\beta{\downarrow_\gamma}$. Further, we define $\Psi{\downarrow_\gamma} = \psi_1{\downarrow_\gamma}$.

Proposition 5.2.1 (Soundness of proof schemata)**.** *Let Ψ be a proof schema with end-sequent $S(n, x_1, \ldots, x_\gamma)$, and let $\alpha \in \mathbb{N}$. Then $\Psi{\downarrow_\alpha}$ is a first-order proof of $S(\alpha, x_1, \ldots, x_\gamma){\downarrow}$.*

Proof. First we prove the proposition for a proof schema consisting of one pair only and then extend the result to arbitrary proof schemata. Assume $\Psi = \langle(\pi, \nu(k))\rangle$. We proceed by induction on α. If $\alpha = 0$, $\Psi\downarrow_0 = \pi\downarrow_0$. The later one differs from π only in defined function symbols, therefore $\pi\downarrow_0$ is a proof of $S(0, x_1, \ldots, x_\gamma)\downarrow$. Now assume for all $\beta \leq \alpha$, $\Psi\downarrow_\beta$ is a proof of $S(\beta, x_1, \ldots, x_\gamma)\downarrow$ and consider the case for $\alpha + 1$. If $(\psi(k, x_1, \ldots, x_\gamma))$ is a proof link in $\nu(k)$, then by hypothesis it rewrites to $\Psi\downarrow_\alpha$. Then after applying rewrite rules of defined function symbols to $\nu(\alpha)$, we get a proof of $S(\alpha + 1, x_1, \ldots, x_\gamma)\downarrow$.

The result is easily extended to arbitrary proof schema Ψ, considering the fact that for all $\alpha \in \mathbb{N}$ each pair $(\pi_i, \nu_i(k)) \in \Psi$ is evaluated to a proof of the sequent $S_i(\alpha, x_{i_1}, \ldots, x_{i_\gamma})\downarrow$. □

Now we are ready to formalize the example given in the beginning of this chapter into our calculus $\mathbf{LK}_s$.

Example 5.2.4. Let us given a term schema $\hat{f}(n, x)$ defined in Example 5.1.3. $\Psi = \langle\varphi, \psi\rangle$ is a proof schema of the sequent $S(n)$:

$$(\forall x)(P(x) \supset P(f(x))) \vdash (P(\hat{f}(n, c)) \supset P(g(n, c))) \supset (P(c) \supset P(g(n, c)))$$

where φ is associated with the pair $(\pi_1, \nu_1(k))$ defined as:[1]

$$\dfrac{\dfrac{}{(\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(k+1, x)))}\,(\psi(k+1)) \qquad (1)}{(\forall x)(P(x) \supset P(f(x))) \vdash (P(\hat{f}(k+1, c)) \supset P(g(k+1, c))) \supset (P(c) \supset P(g(k+1, c)))}\; cut$$

where (1) is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(c) \vdash P(c)} \qquad \dfrac{}{P(\hat{f}(k+1, c)) \vdash P(\hat{f}(k+1, c))} \qquad \dfrac{}{P(g(k+1, c)) \vdash P(g(k+1, c))}}{P(\hat{f}(k+1, c)) \supset P(g(k+1, c)), P(\hat{f}(k+1, c)) \vdash P(g(k+1, c))}\supset: l}{P(c), P(\hat{f}(k+1, c)) \supset P(g(k+1, c)), P(c) \supset P(\hat{f}(k+1, c)) \vdash P(g(k+1, c))}\supset: l}{P(\hat{f}(k+1, c)) \supset P(g(k+1, c)), P(c) \supset P(\hat{f}(k+1, c)) \vdash P(c) \supset P(g(k+1, c))}\supset: r}{P(c) \supset P(\hat{f}(k+1, c)) \vdash (P(\hat{f}(k+1, c)) \supset P(g(k+1, c))) \supset (P(c) \supset P(g(k+1, c)))}\supset: r}{(\forall x)(P(x) \supset P(\hat{f}(k+1, x))) \vdash (P(\hat{f}(k+1, c)) \supset P(g(k+1, c))) \supset (P(c) \supset P(g(k+1, c)))}\forall: l$$

and ψ is associated with the pair $(\pi_2, \nu_2(k))$, where π_2 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(0, u)) \vdash P(\hat{f}(0, u))}{P(u) \vdash P(\hat{f}(0, u))}\,def: l}{\vdash P(u) \supset P(\hat{f}(0, u))}\supset: r}{\vdash (\forall x)(P(x) \supset P(\hat{f}(0, x)))}\forall: r}{(\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(0, x)))}\,w: l$$

[1] To gain some space we only give $\nu_1(k)$, π_1 is the same proof grounded by replacing $k + 1$ with 0.

and $\nu_2(k)$ is:

$$\dfrac{\dfrac{\dfrac{}{(\psi(k))}}{(\forall x)(P(x)\supset P(f(x)))\vdash(\forall x)(P(x)\supset P(\hat{f}(k,x)))}\quad(2)}{(\forall x)(P(x)\supset P(f(x))),(\forall x)(P(x)\supset P(f(x)))\vdash(\forall x)(P(x)\supset P(\hat{f}(k+1,x)))}\,cut}{(\forall x)(P(x)\supset P(f(x)))\vdash(\forall x)(P(x)\supset P(\hat{f}(k+1,x)))}\,c\colon l$$

where (2) is:

$$\dfrac{\dfrac{P(u)\vdash P(u)\qquad\dfrac{\dfrac{\dfrac{P(\hat{f}(k,u))\vdash P(\hat{f}(k,u))\qquad\dfrac{P(\hat{f}(k+1,u))\vdash P(\hat{f}(k+1,u))}{P(f(\hat{f}(k,u)))\vdash P(\hat{f}(k+1,u))}\,def\colon l}{P(\hat{f}(k,u)),P(\hat{f}(k,u))\supset P(f(\hat{f}(k,u)))\vdash P(\hat{f}(k+1,u))}\supset\colon l}{P(\hat{f}(k,u)),(\forall x)(P(x)\supset P(f(x)))\vdash P(\hat{f}(k+1,u))}\forall\colon l}{P(u),P(u)\supset P(\hat{f}(k,u)),(\forall x)(P(x)\supset P(f(x)))\vdash P(\hat{f}(k+1,u))}\supset\colon l}{P(u)\supset P(\hat{f}(k,u)),(\forall x)(P(x)\supset P(f(x)))\vdash P(u)\supset P(\hat{f}(k+1,u))}\supset\colon r}{(\forall x)(P(x)\supset P(\hat{f}(k,x))),(\forall x)(P(x)\supset P(f(x)))\vdash P(u)\supset P(u))}\forall\colon l}{(\forall x)(P(x)\supset P(\hat{f}(k,x))),(\forall x)(P(x)\supset P(f(x)))\vdash(\forall x)(P(x)\supset P(\hat{f}(k+1,x)))}\forall\colon r$$

Clearly, it is not the case that only this example can be translated into our formalism. In general, the following proposition holds.

Proposition 5.2.2. *Let ϕ be an* **LK***-proof without nested inductions of a sequent S fulfilling the following conditions:*

1) *The inductions occurring in ϕ are standard inductions on the natural numbers.*

2) *The term definitions in S are primitive recursive.*

Then ϕ can be transformed into a proof schema.

Proof. Let us given an **LK**-proof ϕ of a sequent S satisfying the conditions of the proposition. First the term definitions (statements like $Def(\hat{f})$ from above) should be removed from S, getting a sequent S', and corresponding term schemata be constructed. Next, every axiom of the form $P(s), t = s \vdash P(t)$ should be replaced with:

$$\dfrac{P(t)\vdash P(t)}{P(s)\vdash P(t)}\,def\colon l\qquad\text{or}\qquad\dfrac{P(s)\vdash P(s)}{P(s)\vdash P(t)}\,def\colon r$$

depending whether $t \to s$ or $s \to t$ rewrite rule was created. And all inferences deriving the term definitions should be skipped. Then we get a proof ϕ' of S'.

Now, since ϕ' does not contain nested inductions, S' contains at most one variable, let us say n, of type ω; so we skip in ϕ' all quantifier introduction rules on n and get a proof $\phi'(n)$ of $S(n)$.

Next, we assign a proof symbol ψ_1 to $\phi'(n)$ and for each induction rule in $\phi'(n)$ we introduce a new proof symbol ψ_β, $\beta > 1$. Then replace all subproofs

ending with the *ind* rule by corresponding proof links in $\phi'(n)$ and get an $\mathbf{LK}_s$-proof $\phi''(n)$. Hence, we construct a pair $(\pi_1, \nu_1(k))$, where $\pi_1 = \phi''(0)$ and $\nu_1(k) = \phi''(k+1)$.

Finally, we transform each subproof ending with the *ind* rule into an $\mathbf{LK}_s$-proof pair in the following way: from

$$\dfrac{\begin{array}{cc}(\chi_1) & (\chi_2)\\ \Gamma \vdash \Delta, A(\bar{0}) & \Pi, A(k) \vdash \Lambda, A(k+1)\end{array}}{\Gamma, \Pi \vdash \Delta, \Lambda, A(t)}\, ind$$

we construct π_β:

$$\dfrac{\begin{array}{c}(\chi_1)\\ \Gamma \vdash \Delta, A(\bar{0})\end{array}}{\Gamma, \Pi \vdash \Delta, \Lambda, A(\bar{0})}\, w\colon l, r*$$

and $\nu_\beta(k)$

$$\dfrac{\begin{array}{cc}(\psi_\beta(k)) & (\chi_2)\\ \bar{\Gamma} \vdash \bar{\Delta}, \bar{A}(\bar{k}) & \Pi, A(k) \vdash \Lambda, A(k+1)\end{array}}{\Gamma, \Pi \vdash \Delta, \Lambda, A(k+1)}\, cut, c\colon l, r*$$

Hence, $\Psi = \langle \psi_1, \ldots, \psi_\alpha \rangle$ is a proof schema of $S(n)$. □

5.2.1 Regularization and Skolemization

Regularization and skolemization are very important for the CERES method. In general, if an **LK**-proof is not regular, the characteristic clause set can be satisfiable. If the proof is not skolemized, projections may be invalid **LK**-proofs, because the eigenvariable conditions may be violated (recall the discussion from Section 2.4).

An **LK**-proof ϕ is skolemized if the end-sequent S of ϕ is skolemized, i.e. S does not contain strong quantifiers. There exist algorithms for skolemization and deskolemization of **LK**-proofs (see [BL94, BHW12]). The extension of these algorithms to proof schemata is the subject to future research.

Definition 5.2.5 (Skolemized proof schema)**.** A proof schema Ψ is called skolemized if the end-sequent $S(n)$ of Ψ does not contain strong quantifiers.

Note that for a skolemized proof schema Ψ, there can be some $\psi_\beta \in \Psi$, $\beta \neq 1$, such that the end-sequents $S_\beta(n)$ of ψ_β are not skolemized. For example, the proof schema Ψ from Example 5.2.4 is skolemized, but $\psi \in \Psi$ is not.

Regularization is vital for CERES when two different eigenvariables come from different branches of a binary rule, that produces an ancestor of some

formula into the end-sequent. Therefore we need also the notion of regularization in proof schemata.

Definition 5.2.6 (Regularity)**.** An $\mathbf{LK}_s$-proof is called regular, if all eigenvariables are distinct and if a free variable u occurs as an eigenvariable in a sequent S of the $\mathbf{LK}_s$-proof, then u occurs only in sequents above S.

But this notion of regularity is not sufficient for proof schemata; the reason is illustrated by the following example.

Example 5.2.7. Consider the proof schema Ψ defined in Example 5.2.4. Then clearly, u is an eigenvariable, all $\mathbf{LK}_s$-proofs in Ψ are regular, but when an instance of the proof schema Ψ is computed for some $\alpha \neq 0$, the instance is not regular any more.

To avoid such collision of eigenvariables, a stronger notion of variable is needed. So, we come up with the notion of *schematic variable*.

Definition 5.2.8 (Schematic variables)**.** We introduce *variable function symbols* of type $\omega \to \iota$. Then the *schematic variables* are built from variable function symbols and terms of type ω.

The semantics of schematic variables is that if x is a schematic variable, then for all $\alpha \in \mathbb{N}$, $x(0), \ldots, x(\alpha)$ corresponds to the sequence of first-order variables $x_0, \ldots, x_\alpha$.

There is a "simpler" solution of the problem mention above, by having a "naming convention" that local eigenvariables be enumerated according to the number of iterations, but there are several reasons not to choose this approach. The first is that the formal parameter k of an $\mathbf{LK}_s$-proof will became a part of variable names and the second is that later, in resolution calculus we need schematic variables anyway. Therefore it is better to directly consider such variables in proof schemata; then if a clause set schema is extracted from a proof schema, schematic variables will be already present.

We redefine our notions of term, formula and the like, in the usual inductive fashion, taking into account schematic variables. A similar approach was used in [MM00]. Although the language they consider is first-order, they allow higher-order quantification on terms (but not on predicates). We do not allow quantification over schematic variables, but just on instances of them (which are variables of type ι).

Example 5.2.9. Let x be a schematic variable, f a binary constant function symbol and $\hat{f}$ a defined function symbol with the rewrite rules:

$$\begin{array}{rcl} \hat{f}(0,x) & \to & x(0) \\ \hat{f}(k+1,x) & \to & f(\hat{f}(k,x), x(k+1)) \end{array}$$

then $\hat{f}(n, x)$ is a term schema, and the sequence of terms for $\alpha = 0, 1, 2, \ldots$ is $x_0, f(x_0, x_1), f(f(x_0, x_1), x_2), \ldots$.

Another issue is to distinguish between *global* and *local* eigenvariables of proof schemata. An eigenvariable is *global* in a proof schema, if it is propagated through proof links, otherwise it is *local*. This distinction is motivated by the fact that a global eigenvariable must occur in (at least) two different $\mathbf{LK}_s$-proof pairs, where in one it is just a free variable and in other it is an eigenvariable.

Example 5.2.10. Consider the proof schema $\Psi = \langle \psi, \varphi \rangle$, where ψ is associated with the pair $(\pi_1, \nu_1(k))$, for $\nu_1(k)$:[2]

$$\dfrac{\dfrac{\dfrac{\dfrac{(\varphi(k+1,u))}{\bigvee_{i=0}^{k+1} P(i,u) \vdash \bigvee_{i=0}^{k+1} P(i,u)}\,\forall\colon l}{(\forall x)(\bigvee_{i=0}^{k+1} P(i,x)) \vdash \bigvee_{i=0}^{k+1} P(i,u)}\,\forall\colon r}{(\forall x)(\bigvee_{i=0}^{k+1} P(i,x)) \vdash (\forall x)(\bigvee_{i=0}^{k+1} P(i,x))} \qquad \dfrac{\dfrac{Q(v) \vdash Q(v)}{(\forall x)Q(x) \vdash Q(v)}\,\forall\colon l}{(\forall x)Q(x) \vdash (\forall x)Q(x)}\,\forall\colon r}{\dfrac{(\forall x)(\bigvee_{i=0}^{k+1} P(i,x)), (\forall x)Q(x) \vdash (\forall x)(\bigvee_{i=0}^{k+1} P(i,x)) \wedge (\forall x)Q(x)}{(\forall x)(\bigvee_{i=0}^{k+1} P(i,x)) \wedge (\forall x)Q(x) \vdash (\forall x)(\bigvee_{i=0}^{k+1} P(i,x)) \wedge (\forall x)Q(x)}\,\wedge\colon l}\,\wedge\colon r$$

and φ is associated with $(\pi_2, \nu_2(k))$, for $\pi_2\colon P(0,u) \vdash P(0,u)$ and $\nu_2(k)$:

$$\dfrac{\dfrac{(\varphi(k,u))}{\bigvee_{i=0}^{k} P(i,u) \vdash \bigvee_{i=0}^{k} P(i,u)} \qquad P(k+1,u) \vdash P(k+1,u)}{\dfrac{\bigvee_{i=0}^{k+1} P(i,u) \vdash \bigvee_{i=0}^{k} P(i,u), P(k+1,u)}{\bigvee_{i=0}^{k+1} P(i,u) \vdash \bigvee_{i=0}^{k+1} P(i,u)}\,\vee\colon r}\,\vee\colon l$$

Then $u\colon \iota$ is a global eigenvariable in Ψ and $v\colon \iota$ is a local one. Note that the proof schema Ψ is regular (even for all its instances).

Finally, we define the procedure of regularization. The idea is to replace free variables with schematic ones, producing different (unique) names for them.

Definition 5.2.11 (Regularization)**.** Let Ψ be a proof schema. Assume there are $y_1\colon \iota, \ldots, y_\alpha\colon \iota$ global eigenvariables in Ψ. For each y_j, $1 \leq j \leq \alpha$, create a fresh schematic variable $v_j\colon \omega \to \iota$ and replace y_j with $v_j(0)\colon \iota$ in Ψ. Next, for each pair $(\pi_i, \nu_i(k)) \in \Psi$, for every local eigenvariable $x_j\colon \iota$ in π_i, create a fresh schematic variable $u_j\colon \omega \to \iota$ and replace x_j with $u_j(0)\colon \iota$. Analogously, for every local eigenvariable $x_j\colon \iota$ in $\nu_i(k)$, create a fresh schematic variable $u_j\colon \omega \to \iota$ and replace x_j with $u_j(k+1)\colon \iota$.

[2] π_1 is the same proof grounded by replacing $k+1$ with 0.

Remark 5.2.12. We could leave global eigenvariables as they are in a proof schema, but we replace them for uniformity, all free variables in the proof schema to be the schematic ones.

Proposition 5.2.3. *Regularization is sound, i.e. if a proof schema is sound, a regularized version of it is also sound.*

Proof. Trivial by Definition 5.2.11 and the fact that we only replace free variables of type ι with the free variables of corresponding type. □

Example 5.2.13. Consider the proof schema Ψ given in Example 5.2.4. Let $v\colon \omega \to \iota$ be a schematic variable. To regularize Ψ, we redefine $(\pi_2, \nu_2(k))$ in the following way ($(\pi_1, \nu_1(k))$ stays the same): π_2 is

$$\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(0,v(0))) \vdash P(\hat{f}(0,v(0)))}{P(v(0)) \vdash P(\hat{f}(0,v(0)))}\,def\colon l}{\vdash P(v(0)) \supset P(\hat{f}(0,v(0)))}\,\supset\colon r}{\vdash (\forall x)(P(x) \supset P(\hat{f}(0,x)))}\,\forall\colon r}{(\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(0,x)))}\,w\colon l$$

and $\nu_2(k)$ is:

$$\dfrac{\dfrac{\dfrac{}{(\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(k,x)))}\,(\psi(k)) \qquad (2)}{(\forall x)(P(x) \supset P(f(x))), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(k+1,x)))}\,cut}{(\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(k+1,x)))}\,c\colon l$$

where (2) is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(v(k+1)) \vdash P(v(k+1))} \qquad \dfrac{\dfrac{\dfrac{P(\hat{f}(k,v(k+1))) \vdash P(\hat{f}(k,v(k+1)))}{P(f(\hat{f}(k,v(k+1)))) \vdash P(\hat{f}(k+1,v(k+1)))}\,\supset\colon l}{P(\hat{f}(k,v(k+1))), P(\hat{f}(k,v(k+1))) \supset P(f(\hat{f}(k,v(k+1)))) \vdash P(\hat{f}(k+1,v(k+1)))}\,\forall\colon l}{P(\hat{f}(k,v(k+1))), (\forall x)(P(x) \supset P(f(x))) \vdash P(\hat{f}(k+1,v(k+1)))}\,\supset\colon l}{P(v(k+1)), P(v(k+1)) \supset P(\hat{f}(k,v(k+1))), (\forall x)(P(x) \supset P(f(x))) \vdash P(\hat{f}(k+1,v(k+1)))}\,\supset\colon r}{P(v(k+1)) \supset P(\hat{f}(k,v(k+1))), (\forall x)(P(x) \supset P(f(x))) \vdash P(v(k+1)) \supset P(\hat{f}(k+1,v(k+1)))}\,\forall\colon l}{(\forall x)(P(x) \supset P(\hat{f}(k,x))), (\forall x)(P(x) \supset P(f(x))) \vdash P(v(k+1)) \supset P(v(k+1)))}\,\forall\colon r}{(\forall x)(P(x) \supset P(\hat{f}(k,x))), (\forall x)(P(x) \supset P(f(x))) \vdash (\forall x)(P(x) \supset P(\hat{f}(k+1,x)))}$$

From now on we consider only regular and skolemized proof schemata.

5.3 Extension of $\mathcal{R}_s$

In previous section we extended term- and formula schemata with schematic variables. Therefore we extend the definitions of clause schema, clause set schema, resolution proof schema and the like by indicating the schematic

variables, occurring in these objects, on the left-hand side of the corresponding rewrite rules. Normal forms of these objects are computed with respect to rewrite rules for defined function symbols as well. We do not copy the formal definitions here, but give some examples.

Example 5.3.1. Let u, v be schematic variables of type $\omega \to \iota$ and $\hat{f}$ a binary defined function symbol with the rewrite rules:

$$\begin{aligned} \hat{f}(0,u) &\to u(0), \\ \hat{f}(k+1,u) &\to f(\hat{f}(k,u)), \end{aligned}$$

then $\hat{f}(n,u)$ is a term schema representing $f^\alpha(u_0)$ for all $\alpha \in \mathbb{N}$.

Let X be a clause variable and consider a clause schema $c(n,u,X)$ w.r.t $\mathcal{R}$:

$$\begin{aligned} c(0,u,X) &\to (\vdash P(0,\hat{f}(0,u))) \circ X, \\ c(k+1,u,X) &\to c(k,u,X) \circ (\vdash P(k+1,\hat{f}(k+1,u))), \end{aligned}$$

then $c(n,u,\vdash)\downarrow_\alpha$ are the clauses $\vdash P(0,u_0), \ldots, P(\alpha, f^\alpha(u_0))$ for all $\alpha \geq 0$.

Next, consider the clause set schema $\mathcal{C}(n) = ((\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3), \mathcal{R})$ where $\mathcal{R}$ is:

$$\begin{aligned} \mathcal{C}_1(0,u,v) &\to \{\vdash P(0,\hat{f}(0,u));\ P(0,v(0)) \vdash\} \\ \mathcal{C}_1(k+1,u,v) &\to \mathcal{C}_2(k+1,u) \oplus \mathcal{C}_3(k+1,v) \\ \mathcal{C}_2(0,u) &\to \{\vdash P(0,\hat{f}(0,u))\} \\ \mathcal{C}_2(k+1,u) &\to \mathcal{C}_2(k,u) \otimes \{\vdash P(k+1,\hat{f}(k+1,u))\} \\ \mathcal{C}_3(0,v) &\to \{P(0,v(0)) \vdash\} \\ \mathcal{C}_3(k+1,v) &\to \mathcal{C}_3(k,u) \oplus \{P(k+1,v(k+1)) \vdash\} \end{aligned}$$

then $\mathcal{C}(n)\downarrow_\alpha$, for all $\alpha \geq 0$, are clause sets

$$\{\vdash P(0,u_0), \ldots, P(\alpha, f^\alpha(u_0));\ P(0,v_0) \vdash;\ P(1,v_1) \vdash; \ldots;\ P(\alpha,v_\alpha) \vdash\}.$$

A resolution refutation schema of $\mathcal{C}(n)$ can be defined as the resolution proof schema $R = ((\varrho), \mathcal{R})$, where $\mathcal{R}$ is the following rewrite system:

$$\begin{aligned} \varrho(0,u,v,X) &\to r((\vdash P(0,\hat{f}(0,u))) \circ X;\ P(0,v(0)) \vdash;\ P(0,\hat{f}(0,u))), \\ \varrho(k+1,u,v,X) &\to r(\varrho(k,u,v,(\vdash P(k+1,\hat{f}(k+1,u))) \circ X); \\ &\qquad P(k+1,v(k+1)) \vdash;\ P(k+1,\hat{f}(k+1,u))) \end{aligned}$$

Note that in resolution terms above the third argument is an atom to be unified for resolution. Therefore for the evaluation of resolution proof schemata we need a stronger notion of substitution, which will unify schematic variables with term schemata.

Definition 5.3.2 (Substitution schema)**.** Let $x_1, \ldots, x_\alpha$ be schematic variables of type $\omega \to \iota$ and $t_1, \ldots, t_\alpha$ be term schemata. Then a substitution schema is an expression of the form $[x_1/\lambda k.t_1, \ldots, x_\alpha/\lambda k.t_\alpha]$.

The intended semantics of a substitution schema is that for all $\gamma \in \mathbb{N}$ we have a substitution $[x_1(\gamma)/t_1\downarrow_\gamma, \ldots, x_\alpha(\gamma)/t_\alpha\downarrow_\gamma]$.

Definition 5.3.3 (Evaluation of proof schemata)**.** Let $R = ((\varrho_1, \ldots, \varrho_\alpha), \mathcal{R})$ be a resolution proof schema, θ be a clause substitution, ϑ be a substitution schema and $\gamma \in \mathbb{N}$, then $R\downarrow_\gamma$ denotes a resolution term which is normal form of $\varrho_1(n, \bar{x}_1, \bar{X}_1)\theta\vartheta[n/\gamma]$ w.r.t. $\mathcal{R}$ and rewrite rules for defined function symbols.

We say that a resolution derivation $R\downarrow_\alpha$, for R being a resolution proof schema and $\alpha \in \mathbb{N}$, is *closed*, iff $R\downarrow_\alpha$ does not contain any free variables, otherwise it is called *open*.

Example 5.3.4. Consider the resolution proof schema from Example 5.3.1. Let $\vartheta_1 = [v/\lambda k.\hat{f}(k, u)]$ and $\vartheta_2 = [u/\lambda k.c]$, where c is some constant, be the substitution schemata. Then, for all $\alpha \in \mathbb{N}$, an open refutation of the clause set $\mathcal{C}(n)\downarrow_\alpha$ is defined by the term $\varrho(n, u, v, \vdash)\vartheta_1\downarrow_\alpha$ and a closed one by the term $\varrho(n, u, v, \vdash)\vartheta_1\vartheta_2\downarrow_\alpha$.

Let compute some instances for both of them. For $\alpha = 0$, recall $\mathcal{C}(n)\downarrow_0 = \{\vdash P(0, u_0);\ P(0, v_0) \vdash\}$, then the open and closed refutations are respectively

$$\dfrac{\dfrac{\vdash P(0,u_0) \qquad P(0,u_0) \vdash}{\vdash} v_0/u_0}{}$$

and

$$\dfrac{\dfrac{\vdash P(0,c) \qquad P(0,c) \vdash}{\vdash} v_0/u_0,\ u_0/c}{}$$

for $\alpha = 1$, recall $\mathcal{C}(n)\downarrow_1 = \{\vdash P(0, u_0), P(1, f(u_0));\ P(0, v_0) \vdash;\ P(1, v_1) \vdash\}$, then the open refutation is

$$\dfrac{\dfrac{\dfrac{\vdash P(0,u_0), P(1,f(u_0)) \qquad P(0,u_0) \vdash}{\vdash P(1,f(u_0))} v_0/u_0 \qquad P(1,f(u_0)) \vdash}{\vdash} v_1/f(u_0)}{}$$

and the closed one is

$$\dfrac{\dfrac{\dfrac{\vdash P(0,c), P(1,f(c)) \qquad P(0,c) \vdash}{\vdash P(1,f(c))} v_0/u_0,\ u_0/c \qquad P(1,f(c)) \vdash}{\vdash} v_1/f(u_0),\ u_0/c}{}$$

It is obvious that finding a substitution schema for a given resolution proof schema is not an easy task. In fact, this problem is undecidable, since our term schemata can represent the primitive recursive functions.

Definition 5.3.5 (Unification problem)**.** A term schema t_1 is unifiable with a term schema t_2 iff there exists a substitution schema ϑ, such that for all $\alpha \in \mathbb{N}$, $t_1\vartheta\downarrow_\alpha = t_2\vartheta\downarrow_\alpha$.

Proposition 5.3.1. *The unification problem of term schemata is undecidable.*

Proof. Term schemata can represent primitive recursive functions, according to the definition of term schemata. In [MR67] it is shown that the primitive recursive functions correspond to so called *Loop programs*. Therefore our language of term schemata can express Loop programs. Let ϑ be a substitution schema for t_1, t_2 being term schemata. Then, the question whether ϑ is a unifier of t_1, t_2 reduces to the equivalence problem for programs p_1, p_2, where p_1 is a loop program representing $t_1\vartheta$ and p_2 is a loop program representing $t_2\vartheta$. But, according to [Tsi70], the equivalence problem for loop programs is undecidable in general. □

Finally, it is obvious that the soundness result holds again on extension of $\mathcal{R}_s$, but the completeness result does not hold even for the bound-linear version of first-order schemata because of the undecidability of unification problem.

5.4 *CERES*$_s$ for First-Order Schemata

There are several things to be changed for *CERES*$_s$ as well. First of all, we should indicate free variables in clause-set symbols and redefine characteristic term in the following way.

Definition 5.4.1 (Characteristic term)**.** Let π be an $\mathbf{LK}_s$-proof, Ω a (cut-) configuration and ρ an inference in π. We define a clause-set term $\Theta_\rho(\pi, \Omega)$ inductively:

- if ρ is a proof link of the form $\dfrac{(\psi(a, t_1, \ldots, t_l))}{\Gamma_\Omega, \Gamma_C, \Gamma \vdash \Delta_\Omega, \Delta_C, \Delta}$ then define Ω' as the cut-configuration corresponding to formula occurrences from $\Gamma_\Omega, \Gamma_C \vdash \Delta_\Omega, \Delta_C$ and $\Theta_\rho(\pi, \Omega) = \mathrm{cl}^{\psi,\Omega'}(a, t_1, \ldots, t_l)$
- otherwise $\Theta_\rho(\pi, \Omega)$ is defined as in Definition 4.1.4.

Finally, define $\Theta(\pi, \Omega) = \Theta_{\rho_0}(\pi, \Omega)$, where ρ_0 is the last inference of π, and $\Theta(\pi) = \Theta(\pi, \emptyset)$.

Then we redefine notion of evaluation for characteristic terms accordingly.

Definition 5.4.2 (Evaluation)**.** Let $\Psi = \langle \psi_1, \ldots, \psi_\alpha \rangle$ be a proof schema. We define the rewrite rules for clause-set symbols for all proof symbols ψ_β

and cut-configurations Ω:

$$\begin{aligned} \mathrm{cl}^{\psi_\beta,\Omega}(0,t_1,\ldots,t_l) &\rightarrow \Theta(\pi_\beta,\Omega)\theta, \\ \mathrm{cl}^{\psi_\beta,\Omega}(k+1,t_1,\ldots,t_l) &\rightarrow \Theta(\nu_\beta(k),\Omega)\theta, \end{aligned}$$

for all $1 \leq \beta \leq \alpha$ and $\theta = \{x_1/t_1,\ldots,x_l/t_l\}$.

Next, let $\gamma \in \mathbb{N}$ and let $\mathrm{cl}^{\psi_\beta,\Omega}\downarrow_\gamma$ be a normal form of $\mathrm{cl}^{\psi_\beta,\Omega}(\gamma,t_1,\ldots,t_l)$ under the rewrite system just given extended with rewrite rules for defined function symbols. Then define $\Theta(\psi_\beta,\Omega) = \mathrm{cl}^{\psi_\beta,\Omega_n}$ and $\Theta(\Psi,\Omega) = \Theta(\psi_1,\Omega)$ and finally the *schematic characteristic term* $\Theta(\Psi) = \Theta(\Psi,\emptyset)$.

The results about characteristic term schema from Section 4.1 still hold and the definition of characteristic clause set schema $\mathrm{CL}(\Psi)$ stays the same. Therefore we do not copy those definitions here, but give an example.

Example 5.4.3. Consider the proof schema Ψ defined in Example 5.2.13. The relevant cut-configurations for Ψ are: $\emptyset$ for φ and $\Omega = \{\vdash (\forall x)(P(x) \supset P(\hat{f}(n,x)))\}$ for ψ. The characteristic terms of Ψ for these cut-configurations are:[3]

$$\begin{aligned} \Theta(\pi_1,\emptyset) &= P(\hat{f}(0,v(0))) \vdash P(\hat{f}(0,v(0)))\oplus \vdash P(c) \oplus (P(\hat{f}(0,c)\vdash \otimes \vdash) \\ \Theta(\nu_1(k),\emptyset) &= \mathrm{cl}^{\psi,\Omega}(k+1)\oplus \vdash P(c) \oplus (P(\hat{f}(k+1,c)\vdash \otimes \vdash) \\ \Theta(\pi_2,\Omega) &= P(\hat{f}(0,v(0))) \vdash P(\hat{f}(0,v(0))) \\ \Theta(\nu_2(k),\Omega) &= \mathrm{cl}^{\psi,\Omega}(k) \oplus P(v(k+1)) \vdash P(v(k+1))\oplus \\ &\qquad (P(\hat{f}(k,v(k+1))) \vdash \otimes \vdash P(\hat{f}(k+1,v(k+1)))) \end{aligned}$$

It is clear that the clause-set symbols have the following order: $\mathrm{cl}^{\varphi,\emptyset}, \mathrm{cl}^{\psi,\Omega}$; therefore the schematic characteristic clause set $\mathrm{CL}(\Psi) = ((\mathcal{C}_1,\mathcal{C}_2),\mathcal{R})$ where $\mathcal{R}$ is:

$$\begin{aligned} \mathcal{C}_1(0) &\rightarrow \{P(\hat{f}(0,v(0))) \vdash P(\hat{f}(0,v(0)));\ \vdash P(c);\ P(\hat{f}(0,c)\vdash\} \\ \mathcal{C}_1(k+1) &\rightarrow \mathcal{C}(k+1) \oplus \{\vdash P(c);\ P(\hat{f}(k+1,c)\vdash\} \\ \mathcal{C}_2(0) &\rightarrow \{P(\hat{f}(0,v(0))) \vdash P(\hat{f}(0,v(0)))\} \\ \mathcal{C}_2(k+1) &\rightarrow \mathcal{C}(k) \oplus \{P(v(k+1)) \vdash P(v(k+1));\\ &\qquad P(\hat{f}(k,v(k+1))) \vdash P(\hat{f}(k+1,v(k+1)))\} \end{aligned}$$

and after tautology deletion $\mathrm{CL}(\Psi) = ((\mathcal{C}_1,\mathcal{C}_2),\mathcal{R})$ where $\mathcal{R}$ is:

$$\begin{aligned} \mathcal{C}_1(0) &\rightarrow \{\vdash P(c);\ P(\hat{f}(0,c)\vdash\} \\ \mathcal{C}_1(k+1) &\rightarrow \mathcal{C}(k+1) \oplus \{\vdash P(c);\ P(\hat{f}(k+1,c)\vdash\} \\ \mathcal{C}_2(0) &\rightarrow \emptyset \\ \mathcal{C}_2(k+1) &\rightarrow \mathcal{C}(k) \oplus \{P(\hat{f}(k,v(k+1))) \vdash P(\hat{f}(k+1,v(k+1)))\} \end{aligned}$$

[3]Note that the proof links in Ψ does not contain free variables, therefore clause-set symbols also.

A resolution refutation schema for the characteristic clause set schema can be defined as $R = ((\varrho, \delta), \mathcal{R})$, where $\mathcal{R}$ is the following rewriting system:

$$\begin{array}{rcl} \varrho(0,v) & \to & r(\delta(0,v);\ P(\hat{f}(0,c)) \vdash;\ P(\hat{f}(0,c))), \\ \varrho(k+1,v) & \to & r(\delta(k+1,v);\ P(\hat{f}(k+1,c)) \vdash;\ P(\hat{f}(k+1,c))), \\ \delta(0,v) & \to & \vdash P(c), \\ \delta(k+1,v) & \to & r(\delta(k,v);\ P(v(k+1)) \vdash P(f(v(k+1)));\ P(\hat{f}(k,c))) \end{array}$$

Next we define a predecessor function $\hat{pre}(n)$. Let $\hat{pre}\colon \omega \to \omega$ be a defined function symbol with the rewrite rules

$$\hat{pre}(0) \to 0 \qquad \text{and} \qquad \hat{pre}(k+1) \to k.$$

Then the substitution schema $\theta = \{v/\lambda k.\hat{f}(\hat{pre}(k), c)\}$ can be used as a substitution schema for R and $\varrho(n,v)\theta{\downarrow_\gamma}$ is a resolution refutation of $\mathrm{CL}(\Psi){\downarrow_\gamma}$ for all $\gamma \in \mathbb{N}$. Note that it is even closed resolution refutation.

Let compute some instances. For $\gamma = 0$, $\mathrm{CL}(\Psi){\downarrow_0} = \{\vdash P(c);\ P(c) \vdash\}$ and $\varrho(n,v)\theta{\downarrow_0}$ is:

$$\dfrac{\vdash P(c) \qquad P(c) \vdash}{\vdash}$$

For $\gamma = 1$, $\mathrm{CL}(\Psi){\downarrow_1} = \{\vdash P(c);\ P(v_1) \vdash P(f(v_1));\ P(f(c)) \vdash\}$ and $\varrho(n,v)\theta{\downarrow_1}$ is:

$$\dfrac{\dfrac{\vdash P(c) \qquad P(c) \vdash P(f(c))}{\vdash P(f(c))}\, v_1/c \qquad P(f(c)) \vdash}{\vdash}$$

Now it remains to define projection term schemata and ACNF for the first-order case. Analogously to clause-set symbols, we should indicate free variables also in projection term symbols. But this is not enough, as after evaluation the $def : l$ and $def : r$ rules become redundant, we will skip them directly in projection terms.

Definition 5.4.4 (Projection term)**.** Let π be an $\mathbf{LK}_s$-proof, Ω an arbitrary (cut-)configuration for π and ρ an inference in π. We define a projection term $\Xi_\rho(\pi, \Omega)$ inductively:

- If ρ is a proof link in π of the form: $\dfrac{(\psi(a, t_1, \ldots, t_l))}{\Gamma_\Omega, \Gamma_C, \Gamma \vdash \Delta_\Omega, \Delta_C, \Delta}$ then define Ω' as the cut-configuration corresponding to formula occurrences from $\Gamma_\Omega, \Gamma_C \vdash \Delta_\Omega, \Delta_C$ and $\Xi_\rho(\pi, \Omega) = \mathrm{pr}^{\psi,\Omega'}(a, t_1, \ldots, t_l)$.

- If ρ is a unary inference with immediate predecessor ρ', and the auxiliary formula(s) of ρ are Ω- or cut-ancestors, or ρ is either $def : l$ or $def : r$, then $\Xi_\rho(\pi, \Omega) = \Xi_{\rho'}(\pi, \Omega)$,
- otherwise $\Xi_\rho(\pi, \Omega)$ is defined as in Definition 4.2.1.

Finally, define $\Xi(\pi, \Omega) = \Xi_{\rho_0}(\pi, \Omega)$, where ρ_0 is the last inference of π.

Definition 5.4.5 (Evaluation)**.** Let $\Psi = \langle \psi_1, \ldots, \psi_\alpha \rangle$ be a proof schema. We define the rewrite rules for projection term symbols for all proof symbols ψ_β and cut-configurations Ω:

$$\begin{aligned} \mathrm{pr}^{\psi_\beta,\Omega}(0, t_1, \ldots, t_l) &\rightarrow \Xi(\pi_\beta, \Omega)\theta, \\ \mathrm{pr}^{\psi_\beta,\Omega}(k+1, t_1, \ldots, t_l) &\rightarrow \Xi(\nu_\beta(k), \Omega)\theta, \end{aligned}$$

for all $1 \leq \beta \leq \alpha$ and $\theta = \{x_1/t_1, \ldots, x_l/t_l\}$.

Next, let $\gamma \in \mathbb{N}$ and let $\mathrm{pr}^{\psi_\beta,\Omega}\downarrow_\gamma$ be a normal form of $\mathrm{pr}^{\psi_\beta,\Omega}(\gamma, t_1, \ldots, t_l)$ under the rewrite system just given extended with rewrite rules for defined function symbols. Then define $\Xi(\psi_\beta, \Omega) = \mathrm{pr}_n^{\psi_\beta,\Omega}$ and $\Xi(\Psi, \Omega) = \Xi(\psi_1, \Omega)$ and finally the *schematic projection term* $\Xi(\Psi) = \Xi(\Psi, \emptyset)$.

Note that the projection terms are computed from skolemized proof schemata. But recall that for a skolemized proof schema Ψ, there can exist an $\mathbf{LK}_s$-proof $\psi_\beta \in \Psi, \beta > 1$, such that the end-sequent of ψ_β contains strong quantifiers. Therefore, for some configurations the projections may be unsound, but this never happens for relevant cut-configurations, since the relevant cut-configurations contain all the formulas containing strong quantifiers. Hence the results about projection term schema from Section 4.2 still hold. The definition of $\mathrm{PR}(\Psi)$ stays the same.

Example 5.4.6. Consider the proof schema Ψ defined in Example 5.2.13 and the relevant cut-configurations for Ψ: $\emptyset$ for φ and $\Omega = \{\vdash (\forall x)(P(x) \supset P(\hat{f}(n, x)))\}$ for ψ. We introduce the following abbreviations for formulas:

$$\begin{aligned} A &= (\forall x)(P(x) \supset P(f(x))) \\ B(n) &= (P(\hat{f}(n, c)) \supset P(g(n, c))) \supset (P(c) \supset P(g(n, c)) \\ B_1(n) &= P(\hat{f}(n, c)) \supset P(g(n, c)) \\ B_2(n) &= P(g(n, c)) \end{aligned}$$

Then the projection terms of Ψ for the cut-configurations $\emptyset, \Omega$ are:

$$\begin{aligned} \Xi(\pi_1, \emptyset) = {} & w^{\vdash B(0)}(w_l(P(\hat{f}(0, v(0))) \vdash P(\hat{f}(0, v(0)))))\oplus \\ & w^{A\vdash}(\supset_r (\supset_r (w^{B_1(0)\vdash B_2(0)}(P(c) \vdash P(c))\oplus \\ & \qquad w^{P(c)\vdash}(P(\hat{f}(0, c)) \vdash P(\hat{f}(0, c))\otimes_{\supset_l} \\ & \qquad\quad P(g(0, c)) \vdash P(g(0, c))))))) \end{aligned}$$

$$\begin{array}{lcl}
\Xi(\nu_1(k),\emptyset) & = & w^{\vdash B(k+1)}(\mathrm{pr}^{\psi,\Omega}(k+1))\oplus \\
& & w^{A\vdash}(\supset_r\ (\supset_r\ (w^{B_1(k+1)\vdash B_2(k+1)}(P(c)\vdash P(c))\oplus \\
& & \qquad w^{P(c)\vdash}(P(\hat{f}(k+1,c))\vdash P(\hat{f}(k+1,c))\otimes_{\supset_l} \\
& & \qquad\quad P(g(k+1,c))\vdash P(g(k+1,c))))))
\end{array}$$

$$\Xi(\pi_2,\Omega) \quad = \quad w_l(P(\hat{f}(0,v(0)))\vdash P(\hat{f}(0,v(0))))$$

$$\begin{array}{lcl}
\Xi(\nu_2(k),\Omega) & = & c_l(w^{A\vdash}(\mathrm{pr}^{\psi,\Omega}(k))\oplus w^{A\vdash}(\\
& & \quad w^{A\vdash}(P(v(k+1))\vdash P(v(k+1)))\oplus \\
& & \quad w^{\vdash}(\forall_l(P(\hat{f}(k,v(k+1)))\vdash P(\hat{f}(k,v(k+1)))\otimes_{\supset_l} \\
& & \qquad P(\hat{f}(k+1,v(k+1)))\vdash P(\hat{f}(k+1,v(k+1))))))))
\end{array}$$

It is easy to see that these projection terms does not contain application of $\forall\colon r$ rule and hence $\mathrm{PR}(\Psi)\downarrow_\alpha$ for all $\alpha\in\mathbb{N}$ are sound. But let consider the term (for the cut-configuration $\emptyset$ for ψ, that is not relevant for Ψ).

$$\begin{array}{lcl}
\Xi(\nu_2(k),\emptyset) & = & c_l(w^{A\vdash(\forall x)(P(x)\supset P(\hat{f}(k+1,x)))}(\mathrm{pr}^{\psi,\Omega}(k))\oplus w^{A\vdash}(\forall_r(\supset_r\ (\\
& & \quad w^{A\vdash P(\hat{f}(k+1,v(k+1)))}(P(v(k+1))\vdash P(v(k+1)))\oplus \\
& & \quad w^{P(v(k+1))\vdash}(\forall_l(P(\hat{f}(k,v(k+1)))\vdash P(\hat{f}(k,v(k+1)))\otimes_{\supset_l} \\
& & \qquad P(\hat{f}(k+1,v(k+1)))\vdash P(\hat{f}(k+1,v(k+1)))))))))
\end{array}$$

Then, for all $\alpha>0$, $|\Xi(\nu_2(k),\emptyset)\downarrow_\alpha|$ is unsound, since the application of the $\forall\colon r$ rule is unsound. We illustrate this fact for $\alpha=1$; $|\Xi(\nu_2(k),\emptyset)\downarrow_1|$ is

$$\left\{\ \dfrac{\dfrac{P(v_0)\vdash P(v_0)}{(\forall x)(P(x)\supset P(f(x))),(\forall x)(P(x)\supset P(f(x))),P(v_0)\vdash(\forall x)(P(x)\supset P(f(x))),P(v_0)}\,w\colon l,r*}{(\forall x)(P(x)\supset P(f(x))),P(v_0)\vdash(\forall x)(P(x)\supset P(f(x))),P(v_0)}\,c\colon l\ \right.\ ;$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(v_1)\vdash P(v_1)}{(\forall x)(P(x)\supset P(f(x))),P(v_1)\vdash P(f(v_1)),P(v_1)}\,w\colon l,r*}{(\forall x)(P(x)\supset P(f(x)))\vdash P(v_1)\supset P(f(v_1)),P(v_1)}\,\supset\colon r}{(\forall x)(P(x)\supset P(f(x)))\vdash(\forall x)(P(x)\supset P(f(x))),P(v_1)}\,\forall\colon r}{(\forall x)(P(x)\supset P(f(x))),(\forall x)(P(x)\supset P(f(x)))\vdash(\forall x)(P(x)\supset P(f(x))),P(v_1)}\,w\colon l}{(\forall x)(P(x)\supset P(f(x)))\vdash(\forall x)(P(x)\supset P(f(x))),P(v_1)}\,c\colon l\ ;$$

$$\left.\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(v_1)\vdash P(v_1)\qquad P(f(v_1))\vdash P(f(v_1))}{P(v_1),P(v_1)\supset P(f(v_1))\vdash P(f(v_1))}\,\supset\colon l}{P(v_1),(\forall x)(P(x)\supset P(f(x)))\vdash P(f(v_1))}\,\forall\colon l}{P(v_1),(\forall x)(P(x)\supset P(f(x))),P(v_1)\vdash P(f(v_1))}\,w\colon l}{(\forall x)(P(x)\supset P(f(x))),P(v_1)\vdash P(v_1)\supset P(f(v_1))}\,\supset\colon r}{(\forall x)(P(x)\supset P(f(x))),P(v_1)\vdash(\forall x)(P(x)\supset P(f(x)))}\,\forall\colon r}{(\forall x)(P(x)\supset P(f(x))),(\forall x)(P(x)\supset P(f(x))),P(v_1)\vdash(\forall x)(P(x)\supset P(f(x)))}\,w\colon l}{(\forall x)(P(x)\supset P(f(x))),P(v_1)\vdash(\forall x)(P(x)\supset P(f(x)))}\,c\colon l\ \right\}$$

and even all these derivations derive tautologies, the last two derivations itself are not sound.

Final step is to redefine the ACNF. The difference to propositional proof schemata is now that the clauses in the resolution refutation are instances of the clauses from the characteristic clause set, therefore we need to apply a substitution to projections as well.

Definition 5.4.7 (ACNF)**.** Let Ψ be a proof schema of the end-sequent $S(n)$, $\mathrm{PR}(\Psi)$ be its projection term schema and R be a resolution refutation schema of the characteristic clause set schema $\mathrm{CL}(\Psi)$. Additionally, let ϑ be a substitution schema used in R. Then a tuple $(\mathrm{PR}(\Psi), R, \vartheta)$ is an Atomic Cut Normal Form schema of Ψ.

Next, by ϑ_α we denote an instance of ϑ for $\alpha \in \mathbb{N}$. Then the ACNF for α is defined in the following way: for all $C \in \mathrm{CL}(\Psi)\downarrow_\alpha$, C' being an instance of C and a leaf node in $T(R\downarrow_\alpha)$, we take a corresponding proof projection $\phi_C \in \mathrm{PR}(\Psi)\downarrow_\alpha$ and compute $\phi'_C = \phi_C\vartheta_\alpha$. Then define $\mathbf{LK}_s$-proof π_α of the end-sequent $S(\alpha)$ by replacing all initial sequents C' in $T(R\downarrow_\alpha)$ by ϕ'_C and adding necessary contractions.

Example 5.4.8. Recall the proof schema Ψ from Example 5.2.13, resolution refutation schema R from Example 5.4.3 and projection terms from Example 5.4.6. Then $T(R\downarrow_0)$ is:

$$\dfrac{\vdash P(c) \qquad P(c) \vdash}{\vdash}\, cut$$

and ACNF of Ψ for $n = 0$ is ($A \vdash B_0$ denotes $S(0)\colon (\forall x)(P(x) \supset P(f(x))) \vdash (P(c) \supset P(g(0,c))) \supset (P(c) \supset P(g(0,c)))$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(c) \vdash P(c)}{P(c) \supset P(g(0,c)), P(c) \vdash P(c), P(g(0,c))}\, w\colon l,r}{P(c) \supset P(g(0,c)) \vdash P(c), P(c) \supset P(g(0,c))}\, \supset\colon r}{\vdash P(c), B_0}\, \supset\colon r}{A \vdash P(c), B_0}\, w\colon l \qquad \dfrac{\dfrac{\dfrac{\dfrac{P(c) \vdash P(c) \qquad P(g(0,c)) \vdash P(g(0,c))}{P(c) \supset P(g(0,c)), P(c) \vdash P(g(0,c))}\, \supset\colon l}{P(c), P(c) \supset P(g(0,c)), P(c) \vdash P(g(0,c))}\, w\colon l}{P(c) \supset P(g(0,c)), P(c) \vdash P(c) \supset P(g(0,c))}\, \supset\colon r}{P(c) \vdash B_0}\, \supset\colon r \quad \dfrac{}{P(c), A \vdash B_0}\, w\colon l$$

$$\dfrac{\dfrac{A, A \vdash B_0, B_0}{(\forall x)(P(x) \supset P(f(x))) \vdash (P(c) \supset P(g(0,c))) \supset (P(c) \supset P(g(0,c)))}\, c\colon l,r}{}\, cut$$

In this case there was no need to apply a substitution, so we compute the ACNF of Ψ for $n = 1$. $\mathrm{CL}(\Psi)\downarrow_1 = \{\vdash P(c);\ P(v_1) \vdash P(f(v_1));\ P(f(c)) \vdash\}$ and $T(R\downarrow_1)$ is:

$$\dfrac{\dfrac{\vdash P(c) \qquad P(c) \vdash P(f(c))}{\vdash P(f(c))}\, cut \qquad P(f(c)) \vdash}{\vdash}\, cut$$

The instance of the substitution schema, used in R, $\theta_1 = [v_0/c, v_1/c]$ and $\mathrm{PR}(\Psi)\downarrow_1$ is:

$$\left\{ \begin{array}{c} \dfrac{\dfrac{\dfrac{\dfrac{P(c) \vdash P(c)}{P(f(c)) \supset P(g(1,c)), P(c) \vdash P(c), P(g(1,c))}\, w\colon l,r}{P(f(c)) \supset P(g(1,c)) \vdash P(c), P(c) \supset P(g(1,c))}\, \supset\colon r}{\vdash (P(f(c)) \supset P(g(1,c))) \supset (P(c) \supset P(g(1,c))), P(c)}\, \supset\colon r}{(\forall x)(P(x) \supset P(f(x))) \vdash (P(f(c)) \supset P(g(1,c))) \supset (P(c) \supset P(g(1,c))), P(c)}\, w\colon l \\ ; \\ \dfrac{\dfrac{\dfrac{\dfrac{P(f(c)) \vdash P(f(c)) \qquad P(g(1,c)) \vdash P(g(1,c))}{P(f(c)) \supset P(g(1,c)), P(f(c)) \vdash P(g(1,c))}}{P(f(c)), P(f(c)) \supset P(g(1,c)), P(c) \vdash P(g(1,c))}\, w\colon l}{P(f(c)), P(f(c)) \supset P(g(1,c)) \vdash P(c) \supset P(g(1,c))}\, \supset\colon r}{P(f(c)) \vdash (P(f(c)) \supset P(g(1,c))) \supset (P(c) \supset P(g(1,c)))}\, \supset\colon r}{P(f(c)), (\forall x)(P(x) \supset P(f(x))) \vdash (P(f(c)) \supset P(g(1,c))) \supset (P(c) \supset P(g(1,c)))}\, w\colon l \\ ; \\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(v_1) \vdash P(v_1) \qquad P(f(v_1)) \vdash P(f(v_1))}{P(v_1), P(v_1) \supset P(f(v_1)) \vdash P(f(v_1))}\, \supset\colon l}{P(v_1), (\forall x)(P(x) \supset P(f(x))) \vdash P(f(v_1))}\, \forall\colon l}{P(v_1), (\forall x)(P(x) \supset P(f(x))), (\forall x)(P(x) \supset P(f(x))) \vdash P(f(v_1))}\, w\colon l}{P(v_1), (\forall x)(P(x) \supset P(f(x))) \vdash P(f(v_1))}\, c\colon l}{P(v_1), (\forall x)(P(x) \supset P(f(x))) \vdash (P(f(c)) \supset P(g(1,c))) \supset (P(c) \supset P(g(1,c))), P(f(v_1))}\, w\colon r \end{array} \right\}$$

Then ACNF of Ψ for $n = 1$ is ($A \vdash B_1$ denotes $S(1)\colon (\forall x)(P(x) \supset P(f(x))) \vdash (P(f(c)) \supset P(g(1,c))) \supset (P(c) \supset P(g(1,c)))$):

$$\dfrac{\dfrac{\dfrac{(\phi_{\vdash P(c)}) \qquad (\phi_{P(v_1)\vdash P(f(v_1))}\theta_1)}{A \vdash B_1, P(c) \qquad P(c), A \vdash B_1, P(f(c))}\, cut \qquad (\phi_{P(f(c))\vdash})}{A, A \vdash B_1, B_1, P(f(c)) \qquad P(f(c)), A \vdash B_1}\, cut}{A, A, A \vdash B_1, B_1, B_1}\, c\colon l, r*}{(\forall x)(P(x) \supset P(f(x))) \vdash (P(f(c)) \supset P(g(1,c))) \supset (P(c) \supset P(g(1,c)))}$$

5.5 Complexity of *CERES*$_s$

It is well known that cut-elimination is nonelementary in general (see [Ore82]), i.e. there is no elementary bound on the size of a cut-free proof in terms of the original one. Recall, that a function is elementary if its computation time is bounded by $e(m, n)$, for a fixed $m \in \mathbb{N}$, n being a length of the input and

$$\begin{aligned} e(0, n) &= n \\ e(k+1, n) &= 2^{e(k,n)} \end{aligned}$$

Nevertheless, in [BL10, Ruk11] subclasses of **LK**-proofs were investigated, where cut-elimination is elementary. The idea was to identify subclasses of **LK**-proofs, for which the characteristic clause set falls into a well known decidable fragment of first-order logic. Following the same idea, it is possible

those results to be extended for proof schemata as well. But finding a decidable fragment of first-order schemata is not an easy task (since unification is undecidable). Therefore this problem is left to future research.

Here we concentrate on propositional schemata and on problems of extending the results from [BL10, Ruk11] to proof schemata. The class ONEQ, a subclass of **LK**-proofs such that all cut-formulas in an **LK**-proof are closed and have at most one quantifier, was proved to be elementary in [Ruk11]. If we consider propositional (bound-linear) proof schemata, they contain at most one parameter, therefore the cut-formulas contain at most one parameter and can be seen as universally quantified formulas over the parameter. So, propositional proof schemata somehow correspond to the ONEQ class and the following proposition holds.

Proposition 5.5.1. *Cut-elimination is elementary on propositional bound-linear proof schemata.*

Proof. Consider a bound-linear proof schema Ψ. It contains at most one parameter, therefore the characteristic clause set schema contains at most one parameter. This means that it falls into the one variable class of first-order logic, which is decidable. Therefore, using the bound for the ONEQ class from [Ruk11], we obtain a bound

$$l(acnf(\Psi, \alpha)) \leq \gamma * l(\Psi\downarrow_\alpha) * 2^{2^{2^{l(\Psi\downarrow_\alpha)}}}$$

where $acnf(\Psi, \alpha)$ denotes an atomic cut normal form of Ψ for α and $\gamma \in \mathbb{N}$ is some constant independent from $l(\Psi\downarrow_\alpha)$. Note that a bound for $l(\Psi\downarrow_\alpha)$ can be computed easily: if β_1 is a number of sequents, and β_2 number of proof links, occurring in Ψ, then $l(\Psi\downarrow_\alpha) \leq \alpha * \beta_1 * \beta_2$. □

Note that the bound obtained in the proof above is not tight and can be improved. But still, while cut-elimination is exponential in usual propositional logic, it becomes at least double exponential in propositional proof schemata.

Next we discuss some problems of extending the elementary classes to first-order proof schemata. Consider the simplest class UIE (see [BL10]), i.e. the class of proofs, where all inferences going into the end-sequent are unary. The clause set of such proof contains only unit clauses, therefore it should contain clauses $\vdash A$ and $B \vdash$ such that A and B are unifiable. Then there is a one step resolution refutation, but since unification is not decidable in proof schemata, we cannot be sure that the resolution refutation exist at all. Therefore one should find a subset, where unification is decidable for term schemata and restrict the class accordingly.

The trivial subset of term schemata, where unification is decidable, is the set of usual first-order terms, but there exists a term schematization language, called *extended primal grammars*, that is a "maximal" language where the unification problem is decidable (see [ACP08]). Therefore we should find a restriction of term schemata to the equivalent class of extended primal grammars.

Usually, the undecidability of the unification problem is caused by two facts. The first is the expressibility of full Peano Arithmetic, i.e. expressing multiplication; the second is the encoding of terms with an infinite number of different variables (see e.g. [Sal93, ACP08] for more information). So, if we restrict our term schemata such that terms of type ω are only arithmetic expressions from Chapter 3 and restrict the terms on the right-hand side of rewrite rules for defined function symbols such that they do not contain nested defined function symbols (such terms are called *regular terms* in [ACP08]), then we get an equivalent class to extended primal grammars. The proof that this restrictions suffice is the subject to future research.

Another problem of extending the elementary classes to first-order proof schemata is that the conditions defining the classes should be adjusted. This is motivated with the fact, that if the $\mathbf{LK}_s$-proofs occurring in a proof schema Ψ separately satisfy the conditions being in some class, the proof schema can still be outside the class. For example, consider a proof schema Ψ with only one proof symbol ψ. Assume $\pi, \nu(k) \in \Psi$ satisfy the conditions being in the class UILM (UIRM), i.e. they contain only one monotone cut and all inferences in the left (right) cut-derivation that go into the end sequent are unary (for the full formal definition see [BL10]). But if $\nu(k)$ contains a proof link to ψ, then Ψ is outside the class UILM (UIRM). Even more, it can be outside the class G-UILM (G-UIRM), which has no restrictions about the number of monotone cuts, but on positions of these cuts in the proof (for the full formal definition see [Ruk11]). So, if there exists a binary inference ρ in $\nu(k)$, such that ρ goes into the end-sequent, and the derivation of one auxiliary formula of ρ contains a proof link to ψ and the other contains a cut, then Ψ is outside the class G-UILM (G-UIRM).

Chapter 6

Applications of $CERES_s$

Beyond theoretical importance, $CERES_s$ has practical applications as well. Here we will investigate several mathematical proofs and analyze them using the $CERES_s$ method. These proofs contain some meaningful cuts and after the analysis we will see that the refutations of characteristic clause set schemata already contain the key steps of mathematical arguments. The analysis is done semi-automatically. We start with the description of the system, implementing the $CERES_s$ method.

6.1 The GAPT Framework

GAPT (General Architecture for Proof Theory)[1] is a framework that aims at providing data-structures, algorithms and user interfaces for analyzing and transforming formal proofs. It is written in the hybrid functional object-oriented language Scala [OSV10]. The framework is very general and it implements the basic data structures for simply-typed lambda calculus, for sequent and resolution proofs, and the like.

There are different algorithms for skolemization, regularization, and the like, implemented in the GAPT framework for **LK**-proofs. The framework implements Gentzen's reductive cut-elimination method and the extraction of data structures central to the CERES method such as characteristic term (called *struct* in GAPT), characteristic clause set and projections. Although GAPT has its own theorem prover, called TAP, the search for refutations is delegated to renowned theorem provers such as Prover9[2] and Vampire[3]. Then the prover TAP is used to replay the refutations found by external provers

[1] GAPT homepage: `http://code.google.com/p/gapt/`

[2] Prover9 Homepage: `http://www.cs.unm.edu/~mccune/prover9/`

[3] Vampire Homepage: `http://www.vprover.org/`

to remove coarse-gained inferences and produce pure resolution refutations (see [DLL+12] for more information about proof replay in GAPT).

The GAPT framework has a *command line interface* (CLI) that allows a user to access the full capabilities of the framework and a *graphical user iterface*, called ProofTool, in which most of features of GAPT are integrated (see [DLL+]).

Although GAPT is a quite complex system and has many useful applications, we will not describe the framework in details here. Rather we focus on ProofTool and its features that are related to the *CERES*$_s$ method. We start our discussion from parsers and exporters.

There are several input formats parsed by the GAPT framework but the most important one (for *CERES*$_s$) is the (schematic) *proof input language*. It is designed to define (schematic) proofs in a form which is both easily human and machine readable. The grammar of the language is very simple. We describe it very briefly here from the user point of view. The full description of the grammar can be found in [Dun12].

proof *name* proves *sequent*
base {
  $id_1 : rule_1$
  ⋮
  $id_n : rule_n$
  root : $rule_{n+1}$
}
step {
  $id_1 : rule_1$
  ⋮
  $id_m : rule_m$
  root : $rule_{m+1}$
}

Figure 6.1: Proof syntax of *proof input language*.

Proof schemata can be written in any text editor with ASCII characters. To differentiate it from other text files, the extension *.lks* is used. One *.lks* file must contain at least one proof pair definition, which has the pattern given in Figure 6.1. For an inductive proof definition, the `base` block describes the base case and the `step` block describes the recursive case. The *id*s are arbitrary labels that are unique within the scope of {...} blocks (i.e. the

same labels can be used in the definition of base and step cases) and rules are tuples consisting of the rule's name, the ids of the premises and of the auxiliary formulas. The keyword `root` as an ID indicates that it is the last inference of an $\mathbf{LK}_s$-proof.

An important feature of the parser of this language is that it has a so-called *auto-propositional mode*. This means that if a user is not interested in specifying exact proofs of propositional sequents, the auto-propositional mode will prove the propositional sequents for the user.

There are also several xml formats in the GAPT framework, such as *simple xml* format, that is used to communicate with other systems. One such example is `RegSTAB` [ACP10]. It is a STAB prover that refutes regular formula schemata and can output the tableau refutation in a simple xml format, which can be read and displayed by ProofTool.

In ProofTool there are various exporters for objects of the GAPT framework. The important ones are LaTeX and *pdf exportes*. All objects from ProofTool can be exported directly into *.pdf* files and additionally, the proofs and clause sets can be exported into *.tex* files as well.

The GAPT framework contains an implementation of the basic data-structures and algorithms for the *CERES*$_s$ method such as characteristic clause-set terms, called *schematic struct* in GAPT, characteristic clause-set schema, projection term schema and their extraction from proof schemata. An important feature of ProofTool is that it allows instantiation of a proof schema for some parameter. Then the data-structures needed for the usual CERES method can be constructed and manipulated for the instance.

The characteristic and projection terms in ProofTool are displayed as trees. Therefore some screenshots taken from ProofTool, which are used in the example below, differs a bit from the usual notations used in theoretical part of the thesis.

Finally, for a more detailed description of the GAPT framework, as well on implementation details of *CERES*$_s$ we refer the interested reader to [Dun12].

6.2 The Adder Proof

Our first example is about electronic adders, which are circuits adding two numbers. A one-bit adder is a circuit, that adds two digits using a so called *carry bit* and an n-bit adder is a composition of n one-bit adders. We consider the following implementation of a one-bit adder: the sum of bits A and B, using carry bit C is $S = A \odot B \odot C$, where $\odot$ denotes an exclusive or, the initial carry bit is 0 and the output carry bit computation is $(A \wedge B) \odot (C \wedge (A \odot B))$.

Now, we analyze the following proposition.

Proposition 6.2.1. *A circuit n-bit adder is commutative.*

Proof. The initial carry bits for computing $A + B$ and $B + A$ are 0, then by the computation of output carry bits, all the corresponding carry bits are equal for each step of computation. Since all carry bits are equal, the sums are also equal. □

Even though the proof is mathematically simple and short, its formalization in propositional proof schemata is quite big, (the *adder.lks* file contains about 3500 lines). But it has several nice properties, which are mentioned below. We start with some formula definitions ($\hat{\mathbf{S}}$ denotes the sum and $\hat{\mathbf{C}}$ the carry bit computation):

$$\begin{array}{rcl}
A \odot B & =_{def} & (A \wedge \neg B) \vee (\neg A \wedge B) \\
A \Leftrightarrow B & =_{def} & (\neg A \vee B) \wedge (\neg B \vee A) \\
\hat{\mathbf{S}}_i & =_{def} & S_i \Leftrightarrow (A_i \odot B_i) \odot C_i \\
\hat{\mathbf{S}}'_i & =_{def} & S'_i \Leftrightarrow (B_i \odot A_i) \odot C'_i \\
\hat{\mathbf{C}}_i & =_{def} & C_{i+1} \Leftrightarrow (A_i \wedge B_i) \vee (C_i \wedge A_i) \vee (C_i \wedge B_i) \\
\hat{\mathbf{C}}'_i & =_{def} & C'_{i+1} \Leftrightarrow (B_i \wedge A_i) \vee (C'_i \wedge B_i) \vee (C'_i \wedge A_i) \\
Adder_n & =_{def} & \bigwedge_{i=0}^{n} \hat{\mathbf{S}}_i \wedge \bigwedge_{i=0}^{n} \hat{\mathbf{C}}_i \wedge \neg C_0 \\
Adder'_n & =_{def} & \bigwedge_{i=0}^{n} \hat{\mathbf{S}}'_i \wedge \bigwedge_{i=0}^{n} \hat{\mathbf{C}}'_i \wedge \neg C'_0 \\
EqC_n & =_{def} & \bigwedge_{i=0}^{n} (C_i \Leftrightarrow C'_i) \\
EqS_n & =_{def} & \bigwedge_{i=0}^{n} (S_i \Leftrightarrow S'_i)
\end{array}$$

Then the lemma *the carry bits are equal* is represented by the formula EqC_n and the proposition *a circuit n-bit adder is commutative* by the sequent schema $Adder_n, Adder'_n \vdash EqS_n$.

The formalization of the proof above is the proof schema $\Psi = \langle \psi, \varphi, \phi, \chi \rangle$, where ψ is associated with the pair $(\pi_1, \nu_1(k))$, $\nu_1(k)$ is:[4]

$$\dfrac{\dfrac{\overset{(\varphi(k+1))}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k+1} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k+1} \hat{\mathbf{C}}'_i \vdash EqC_{k+1}} \qquad \overset{(\chi(k+1))}{EqC_{k+1}, \bigwedge_{i=0}^{k+1} \hat{\mathbf{S}}_i, \bigwedge_{i=0}^{k+1} \hat{\mathbf{S}}'_i \vdash EqS_{k+1}}}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k+1} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}'_i, \bigwedge_{i=0}^{k+1} \hat{\mathbf{S}}_i, \bigwedge_{i=0}^{k+1} \hat{\mathbf{S}}'_i \vdash EqS_{k+1}}\, cut}{Adder_{k+1}, Adder'_{k+1} \vdash EqS_{k+1}}\, \wedge : l*$$

φ is associated with the pair $(\pi_2, \nu_2(k))$ and $\nu_2(k)$ is:

[4]Since the proofs are huge, we omit all base cases and the purely propositional parts of the proofs. A full formal proof (the *adder.lks* file) can be found at `http://www.logic.at/asap/`

$$\dfrac{\dfrac{\overset{(\varphi(k))}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}'_i \vdash EqC_k} \qquad \overset{(\phi(k))}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}'_i \vdash C_{k+1} \Leftrightarrow C'_{k+1}}}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}'_i \vdash EqC_{k+1}} \wedge\colon r, c\colon l*}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k+1} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k+1} \hat{\mathbf{C}}'_i \vdash EqC_{k+1}} w\colon l, \wedge\colon l*$$

ϕ is associated with the pair $(\pi_3, \nu_3(k))$ and $\nu_3(k)$ is:

$$\dfrac{\dfrac{\overset{(\phi(k))}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}'_i \vdash C_{k+1} \Leftrightarrow C'_{k+1}} \qquad \begin{array}{c}\vdots\\ C_{k+1} \Leftrightarrow C'_{k+1}, \hat{\mathbf{C}}_{k+1}, \hat{\mathbf{C}}'_{k+1} \vdash C_{k+2} \Leftrightarrow C'_{k+2}\end{array}}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{C}}'_i, \hat{\mathbf{C}}_{k+1}, \hat{\mathbf{C}}'_{k+1} \vdash C_{k+2} \Leftrightarrow C'_{k+2}} cut}{\neg C_0, \neg C'_0, \bigwedge_{i=0}^{k+1} \hat{\mathbf{C}}_i, \bigwedge_{i=0}^{k+1} \hat{\mathbf{C}}'_i \vdash C_{k+2} \Leftrightarrow C'_{k+2}} \wedge\colon l*$$

and finally, χ is associated with the pair $(\pi_4, \nu_4(k))$ and $\nu_4(k)$ is:

$$\dfrac{\dfrac{\overset{(\chi(k))}{EqC_k, \bigwedge_{i=0}^{k} \hat{\mathbf{S}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{S}}'_i \vdash EqS_k} \qquad \begin{array}{c}\vdots\\ C_{k+1} \Leftrightarrow C'_{k+1}, \hat{\mathbf{S}}_{k+1}, \hat{\mathbf{S}}'_{k+1} \vdash S_{k+1} \Leftrightarrow S'_{k+1}\end{array}}{EqC_k, \bigwedge_{i=0}^{k} \hat{\mathbf{S}}_i, \bigwedge_{i=0}^{k} \hat{\mathbf{S}}'_i, C_{k+1} \Leftrightarrow C'_{k+1}, \hat{\mathbf{S}}_{k+1}, \hat{\mathbf{S}}'_{k+1} \vdash EqS_{k+1}} \wedge\colon r}{EqC_{k+1}, \bigwedge_{i=0}^{k+1} \hat{\mathbf{S}}_i, \bigwedge_{i=0}^{k+1} \hat{\mathbf{S}}'_i \vdash EqS_{k+1}} \wedge\colon l*$$

It is easy to see that for each proof symbol there is only one cut-configuration, namely: $\emptyset$ for ψ and

$$\begin{aligned}\Omega_\varphi &= \{\vdash EqC_n\}\\ \Omega_\phi &= \{\vdash C_{n+1} \Leftrightarrow C'_{n+1}\}\\ \Omega_\chi &= \{EqC_n \vdash\}\end{aligned}$$

The characteristic and projection terms for (some of) these cut-configurations are given in Appendix B. The (simplified[5]) characteristic clause set schema $\mathrm{CL}(\Psi) = ((\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4), \mathcal{R})$, where $\mathcal{R}$ is:

$$\begin{aligned}\mathcal{C}_1(0) &\to \{C_0 \vdash;\quad C'_0 \vdash;\quad \vdash C_0, C'_0;\quad C_0, C'_0 \vdash\},\\ \mathcal{C}_1(k+1) &\to \mathcal{C}_2(k+1) \oplus \mathcal{C}_4(k+1),\\ \mathcal{C}_2(0) &\to \{C_0 \vdash;\quad C'_0 \vdash\},\\ \mathcal{C}_2(k+1) &\to \mathcal{C}_2(k) \oplus \mathcal{C}_3(k),\\ \mathcal{C}_3(0) &\to \{C_1 \vdash C'_1;\quad C'_1 \vdash C_1\},\\ \mathcal{C}_3(k+1) &\to \mathcal{C}_3(k) \oplus \{C_{k+2} \vdash C'_{k+2}, C_{k+1};\quad C'_{k+2} \vdash C_{k+2}, C'_{k+1};\\ &\qquad\qquad C'_{k+1}, C_{k+2} \vdash C'_{k+2};\quad C_{k+1}, C'_{k+2} \vdash C_{k+2}\},\\ \mathcal{C}_4(0) &\to \{\vdash C_0, C'_0;\quad C_0, C'_0 \vdash\},\\ \mathcal{C}_4(k+1) &\to \mathcal{C}_4(k) \otimes \{\vdash C_{k+1}, C'_{k+1};\quad C_{k+1}, C'_{k+1} \vdash\}.\end{aligned}$$

[5]For the better readability, factorization is done on the clauses and tautologies are deleted.

Note that, the clauses in $\mathcal{C}_4$ have parameter dependent number of literals, namely, for all $\alpha \in \mathbb{N}$, the clauses in $\mathcal{C}_4{\downarrow_\alpha}$ contain $2{*}(\alpha{+}1)$ literals. Therefore, for the specification of a corresponding resolution refutation schema, the use of clause variables is needed. A resolution refutation schema of $\mathrm{CL}(\Psi)$ is $R = ((\varrho, \delta, \eta), \mathcal{R})$ where $\mathcal{R}$ is the following rewrite system:

$$\begin{array}{rcl}
\varrho(0, X) & \to & r(r((\vdash C_0, C_0') \circ X;\ C_0 \vdash;\ C_0);\ C_0' \vdash;\ C_0'), \\
\varrho(k+1, X) & \to & r(r(\varrho(k, (\vdash C_{k+1}, C_{k+1}') \circ X);\ \eta(k);\ C_{k+1}'); \\
 & & \quad r(\delta(k);\ \varrho(k, (C_{k+1}, C_{k+1}' \vdash) \circ X);\ C_{k+1}'); \\
 & & \quad C_{k+1}), \\
\delta(0) & \to & C_1 \vdash C_1', \\
\delta(k+1) & \to & r(C_{k+2} \vdash C_{k+2}', C_{k+1}; \\
 & & \quad r(\delta(k);\ C_{k+1}', C_{k+2} \vdash C_{k+2}';\ C_{k+1}'); \\
 & & \quad C_{k+1}), \\
\eta(0) & \to & C_1' \vdash C_1, \\
\eta(k+1) & \to & r(C_{k+2}' \vdash C_{k+2}, C_{k+1}'; \\
 & & \quad r(\eta(k);\ C_{k+1}, C_{k+2}' \vdash C_{k+2};\ C_{k+1}); \\
 & & \quad C_{k+1}')
\end{array}$$

The resolution refutation of $\mathrm{CL}(\Psi){\downarrow_\alpha}$ is defined by $\varrho(n, \vdash){\downarrow_\alpha}$, for all $\alpha \in \mathbb{N}$.

Note that the derivations $\delta(n)$ and $\eta(n)$, for all $\alpha \in \mathbb{N}$, derive respectively the clauses $C_{\alpha+1} \vdash C_{\alpha+1}'$ and $C_{\alpha+1}' \vdash C_{\alpha+1}$ modulo factoring.

It is easy to verify that, for all $\alpha \in \mathbb{N}$, $\varrho(n, \vdash){\downarrow_\alpha}$ is a resolution refutation of $\mathrm{CL}(\Psi)\downarrow_\alpha$. For $\varrho(n, \vdash)\downarrow_0$ this is trivial. Assume that $\varrho(n, \vdash)\downarrow_\alpha$ is a resolution refutation of $\mathrm{CL}(\Psi)\downarrow_\alpha$, for $\alpha > 0$. Then, using this induction hypothesis, $\varrho(n, \vdash){\downarrow_{\alpha+1}}$ is a resolution refutation if

$$\begin{array}{l}
r(r(\vdash C_{\alpha+1}, C_{\alpha+1}';\ C_{\alpha+1}' \vdash C_{\alpha+1};\ C_{\alpha+1}'); \\
\quad r(C_{\alpha+1} \vdash C_{\alpha+1}';\ C_{\alpha+1}, C_{\alpha+1}' \vdash;\ C_{\alpha+1}'); \\
\quad C_{\alpha+1})
\end{array}$$

is a resolution refutation, which it obviously is. Clearly, all clauses at the leaf nodes of $\varrho(n, \vdash){\downarrow_{\alpha+1}}$ are in $\mathrm{CL}(\Psi){\downarrow_{\alpha+1}}$.

Let us illustrate this with some instances. For $\alpha = 0$,

$$\mathrm{CL}(\Psi)\downarrow_0 \ = \ \{C_0 \vdash; \quad C_0' \vdash; \quad \vdash C_0, C_0'; \quad C_0, C_0' \vdash\}$$

and $\varrho(n, \vdash)\downarrow_0$ is:

$$\dfrac{\dfrac{\vdash C_0, C_0' \qquad C_0 \vdash}{\vdash C_0'} \qquad C_0' \vdash}{\vdash}$$

For $\alpha = 1$, $\mathrm{CL}(\Psi)\downarrow_1$ is:

$$\{C_0 \vdash; \quad C_0' \vdash; \quad C_1 \vdash C_1'; \quad C_1' \vdash C_1; \quad \vdash C_0, C_0', C_1, C_1';$$
$$C_0', C_0 \vdash C_1, C_1'; \quad C_1', C_1 \vdash C_0, C_0'; \quad C_0, C_0', C_1, C_1' \vdash \}$$

and $\varrho(n, \vdash)\downarrow_0$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash C_0, C_0', C_1, C_1' \qquad C_0 \vdash}{\vdash C_0', C_1, C_1'} \qquad C_0' \vdash}{\vdash C_1, C_1'} \qquad C_1' \vdash C_1}{\vdash C_1} \qquad \dfrac{C_1 \vdash C_1' \qquad \dfrac{\dfrac{C_1, C_1' \vdash C_0, C_0' \qquad C_0 \vdash}{C_1, C_1' \vdash C_0'} \qquad C_0' \vdash}{C_1, C_1' \vdash}}{C_1 \vdash}}{\vdash}$$

Since the proof schema is too big, it is not possible to give the ACNF here, but still we can see that the proof schema has some interesting properties. First, we observe that for all $\alpha \in \mathbb{N}$, the number of clauses in $\mathrm{CL}(\Psi)\downarrow_\alpha$ is exponentially growing, namely $\mathrm{CL}(\Psi)\downarrow_\alpha$ contains $2^{\alpha+2}$ clauses. Second, the resolution refutation is also exponential in α.

Finally, while the original proof schema (with cuts) used the lemma that all carry bits are equal, from the resolution refutation (which is a skeleton of the essential cut-free proof) we can see that the (essential) cut-free proof now derives the equality of the carry bits one-by-one.

6.3 The Exponential Proof

Our next example is taken from [BL99], where a sequence of proofs was given as a worst case example for the so called QMON class, on which cut-elimination is exponential. A meta-level proof schema of the sequents

$$S_n \colon T, (\forall x) P(f(x), x) \vdash P(f^{2^n}(a), a)$$

was defined, where T is transitivity of P, i.e.

$$T \colon (\forall x)(\forall y)(\forall z)(P(x, y) \wedge P(y, z) \supset P(x, z))$$

and proved via the pigeonhole principle that every cut-free proof of S_n must contain initial sequents $P(f^{i+1}(a), f^i(a)) \vdash P(f^{i+1}(a), f^i(a))$ for all $i = 0, \ldots, 2^n - 1$ (and thus is exponential in n).

Below we will formalize that meta-level schema within our language and prove the statement above via cut-elimination. We start from the term definitions. Let $\hat{f}$ be a binary defined function symbol with the rewrite rules:

$$\begin{aligned} \hat{f}(0, x) &\to x, \\ \hat{f}(k+1, x) &\to f(\hat{f}(k, x)). \end{aligned}$$

Then $\hat{f}(\alpha, x)$ represents $f^\alpha(x)$ for all $\alpha \in \mathbb{N}$. So it remains to define a function representing 2^n. We define the function $\hat{e}(n)$, for a unary defined function symbol $\hat{e}$, with the rewrite rules:

$$\begin{aligned} \hat{e}(0) &\to 1, \\ \hat{e}(k+1) &\to \hat{e}(k) + \hat{e}(k). \end{aligned}$$

Then the term $f^{2^n}(x)$ is represented by $\hat{f}(\hat{e}(n), x)$ and we should construct a proof schema of the sequent

$$S(n)\colon T, (\forall x)P(f(x), x) \vdash P(\hat{f}(\hat{e}(n), a), a).$$

A proof schema Ψ of $S(n)$, is the tuple: $\langle \psi, \varphi \rangle$, where ψ is associated with the pair $(\pi_1, \nu_1(k))$; π_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(\hat{e}(0), a), a) \vdash P(\hat{f}(\hat{e}(0), a), a)}{P(f(a), a) \vdash P(\hat{f}(\hat{e}(0), a), a)}\, def\colon l}{(\forall x)P(f(x), x) \vdash P(\hat{f}(\hat{e}(0), a), a)}\, \forall\colon l}{T, (\forall x)P(f(x), x) \vdash P(\hat{f}(\hat{e}(0), a), a)}\, w\colon l$$

and $\nu_1(k)$ is:

$$\dfrac{\dfrac{(\varphi(k+1))}{T, (\forall x)P(f(x), x) \vdash (\forall x)P(\hat{f}(\hat{e}(k+1), x), x)} \qquad \dfrac{P(\hat{f}(\hat{e}(k+1), a), a) \vdash P(\hat{f}(\hat{e}(k+1), a), a)}{(\forall x)P(\hat{f}(\hat{e}(k+1), x), x) \vdash P(\hat{f}(\hat{e}(k+1), a), a)}\, \forall\colon l}{T, (\forall x)P(f(x), x) \vdash P(\hat{f}(\hat{e}(k+1), a), a)}\, cut$$

where φ is associated with the pair $(\pi_2, \nu_2(k))$; π_2 is ($u\colon \omega \to \iota$ is a schematic variable):

$$\dfrac{\dfrac{\dfrac{\dfrac{P(\hat{f}(\hat{e}(0), u(0)), u(0)) \vdash P(\hat{f}(\hat{e}(0), u(0)), u(0))}{P(f(u(0)), u(0)) \vdash P(\hat{f}(\hat{e}(0), u(0)), u(0))}\, def\colon l}{(\forall x)P(f(x), x) \vdash P(\hat{f}(\hat{e}(0), u(0)), u(0))}\, \forall\colon l}{(\forall x)P(f(x), x) \vdash (\forall x)P(\hat{f}(\hat{e}(0), x), x)}\, \forall\colon r}{T, (\forall x)P(f(x), x) \vdash (\forall x)P(\hat{f}(\hat{e}(0), x), x)}\, w\colon l$$

and $\nu_2(k)$ is:

$$\dfrac{\dfrac{\dfrac{(\varphi(k))}{T, (\forall x)P(f(x), x) \vdash (\forall x)P(\hat{f}(\hat{e}(k), x), x)} \qquad \dfrac{(1)}{(\forall x)P(\hat{f}(\hat{e}(k), x), x), T \vdash (\forall x)P(\hat{f}(\hat{e}(k+1), x), x)}\, cut}{T, T, (\forall x)P(f(x), x) \vdash (\forall x)P(\hat{f}(\hat{e}(k+1), x), x)}\, c\colon l}{T, (\forall x)P(f(x), x) \vdash (\forall x)P(\hat{f}(\hat{e}(k+1), x), x)}\, cut$$

where (1) is ($A(i, j)$ is an abbreviation of $P(\hat{f}(i, u(k+1)), \hat{f}(j, u(k+1)))$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A(\hat{e}(k+1),\hat{e}(k)) \vdash A(\hat{e}(k+1),\hat{e}(k)) \qquad A(\hat{e}(k),0) \vdash A(\hat{e}(k),0)}{A(\hat{e}(k+1),\hat{e}(k)), A(\hat{e}(k),0) \vdash A(\hat{e}(k+1),\hat{e}(k)) \wedge A(\hat{e}(k),0)} \wedge : r \qquad A(\hat{e}(k+1),0) \vdash A(\hat{e}(k+1),0)}{(A(\hat{e}(k+1),\hat{e}(k)) \wedge A(\hat{e}(k),0)) \supset A(\hat{e}(k+1),0), A(\hat{e}(k+1),\hat{e}(k)), A(\hat{e}(k),0) \vdash A(\hat{e}(k+1),0)} \supset : l}{T, A(\hat{e}(k+1),\hat{e}(k)), A(\hat{e}(k),0) \vdash A(\hat{e}(k+1),0)} \forall : l*}{T, P(\hat{f}(\hat{e}(k),\hat{f}(\hat{e}(k),u(k+1))),\hat{f}(\hat{e}(k),u(k+1))), A(\hat{e}(k),0) \vdash A(\hat{e}(k+1),0)} def : l}{T, (\forall x)P(\hat{f}(\hat{e}(k),x),x), A(\hat{e}(k),0) \vdash A(\hat{e}(k+1),0)} \forall : l}{T, (\forall x)P(\hat{f}(\hat{e}(k),x),x), (\forall x)P(\hat{f}(\hat{e}(k),x),x) \vdash A(\hat{e}(k+1),0)} c : l}{T, (\forall x)P(\hat{f}(\hat{e}(k),x),x) \vdash A(\hat{e}(k+1),0)} \forall : r}{T, (\forall x)P(\hat{f}(\hat{e}(k),x),x) \vdash (\forall x)P(\hat{f}(\hat{e}(k+1),x),x)}$$

There are two relevant cut-configurations for Ψ, namely $\emptyset$ for ψ and $\Omega = \{\vdash (\forall x)P(\hat{f}(\hat{e}(k),x),x)\}$ for φ. The characteristic terms for these cut-configurations are:

$$\begin{array}{lcl}
\Theta(\pi_1,\emptyset) & = & \vdash \\
\Theta(\nu_1(k),\emptyset) & = & \mathrm{cl}_{k+1}^{\varphi,\Omega} \oplus P(\hat{f}(\hat{e}(k+1),a),a) \vdash \\
\Theta(\pi_2,\Omega) & = & \vdash P(\hat{f}(\hat{e}(0),u(0)),u(0)) \\
\Theta(\nu_2(k),\Omega) & = & \mathrm{cl}_{k}^{\varphi,\Omega} \oplus (P(\hat{f}(\hat{e}(k+1),u(k+1)),\hat{f}(\hat{e}(k),u(k+1))) \vdash \otimes \\
 & & \qquad P(\hat{f}(\hat{e}(k),u(k+1)),u(k+1)) \vdash \otimes \\
 & & \qquad \vdash P(\hat{f}(\hat{e}(k+1),u(k+1)),u(k+1)) \ )
\end{array}$$

Then the characteristic clause set schema $\mathrm{CL}(\Psi) = ((\mathcal{C}_1,\mathcal{C}_2),\mathcal{R})$, where $\mathcal{R}$ is:

$$\begin{array}{rcl}
\mathcal{C}_1(0) & \to & \{\vdash\} \\
\mathcal{C}_1(k+1) & \to & \mathcal{C}_2(k+1) \oplus \{P(\hat{f}(\hat{e}(k+1),a),a) \vdash\} \\
\mathcal{C}_2(0) & \to & \{\vdash P(\hat{f}(\hat{e}(0),u(0)),u(0))\} \\
\mathcal{C}_2(k+1) & \to & \mathcal{C}_2(k) \oplus \{C(k+1,u)\}
\end{array}$$

where $C(k+1,u)$ is the following clause:

$$P(\hat{f}(\hat{e}(k+1),u(k+1)),\hat{f}(\hat{e}(k),u(k+1))),\ P(\hat{f}(\hat{e}(k),u(k+1)),u(k+1)) \vdash P(\hat{f}(\hat{e}(k+1),u(k+1)),u(k+1))$$

The resolution refutation schema of $\mathrm{CL}(\Psi)$, $R = ((\varrho,\delta),\mathcal{R})$, where $\mathcal{R}$ is:

$$\begin{array}{rcl}
\varrho(0,u,v) & \to & \vdash \\
\varrho(k+1,u,v) & \to & r(\delta(k+1,u,v);\ P(\hat{f}(\hat{e}(k+1),a),a) \vdash;\ P(\hat{f}(\hat{e}(k+1),a),a)) \\
\delta(0,u,v) & \to & \vdash P(\hat{f}(\hat{e}(0),u(0)),u(0)) \\
\delta(k+1,u,v) & \to & r(\delta(k,u',v); \\
 & & \qquad r(\delta(k,u,v);\ C(k+1,v);\ P(\hat{f}(\hat{e}(k+1),v(k+1)),\hat{f}(\hat{e}(k),v(k+1)))); \\
 & & \qquad P(\hat{f}(\hat{e}(k),v(k+1)),v(k+1)) \ )
\end{array}$$

Note that $\delta(k, u', v)$ is just a renaming of the schematic variable u with u' in the end-sequent of δ. This renaming is necessary for defining a substitution schema for R, which is:

$$\begin{aligned}\vartheta &= [u/\lambda k.\hat{f}(\hat{e}(k), v(k+1)),\ u'/\lambda k.v(k+1)][v/\lambda k.a] \\ &= [u/\lambda k.\hat{f}(\hat{e}(k), a),\ u'/\lambda k.a,\ v/\lambda k.a]\end{aligned}$$

Then the resolution refutation of $\mathrm{CL}(\Psi)$ for all $\alpha \in \mathbb{N}$ is $\varrho(n, u, v)\vartheta{\downarrow_\alpha}$.

It remains to compute the projection terms and the ACNF. The projection terms for the cut-configurations defined above are (the abbreviation $A(i, j)$ from above is used here again):

$$\begin{aligned}
\Xi(\pi_1, \emptyset) &= w_l(\forall_l(P(\hat{f}(\hat{e}(0), a), a) \vdash P(\hat{f}(\hat{e}(0), a), a))) \\
\Xi(\nu_1(k), \emptyset) &= w^{\vdash P(\hat{f}(\hat{e}(k+1),a),a)}(\mathrm{pr}^{\varphi,\Omega}(k+1)) \oplus \\
&\quad\ w^{T,(\forall x)P(f(x),x)\vdash}(P(\hat{f}(\hat{e}(k+1), a), a) \vdash P(\hat{f}(\hat{e}(k+1), a), a)\ ) \\
\Xi(\pi_2, \Omega) &= w_l(\forall_l(P(\hat{f}(\hat{e}(0), u(0)), u(0)) \vdash P(\hat{f}(\hat{e}(0), u(0)), u(0))) \\
\Xi(\nu_2(k), \Omega) &= c_l(w^{T\vdash}(\mathrm{pr}^{\varphi,\Omega}(k)) \oplus w^{T,(\forall x)P(f(x),x)\vdash}(\forall_l(\forall_l(\forall_l(\\
&\qquad (A(\hat{e}(k+1), \hat{e}(k)) \vdash A(\hat{e}(k+1), \hat{e}(k)) \otimes_{\wedge_r} \\
&\qquad\ A(\hat{e}(k), 0) \vdash A(\hat{e}(k), 0)\ ) \otimes_{\supset_l} \\
&\qquad A(\hat{e}(k+1), 0) \vdash A(\hat{e}(k+1), 0)\ ))))))
\end{aligned}$$

Now, the ACNF schema is the tuple $(\mathrm{PR}(\Psi), R, \vartheta)$. Let us compute some instances. For $\alpha = 0$, $R{\downarrow_0}$ is just $\vdash$ and $\mathrm{PR}(\Psi){\downarrow_0}$ is:

$$\left\{ \quad \dfrac{\dfrac{P(f(a), a) \vdash P(f(a), a)}{(\forall x)P(f(x), x) \vdash P(f(a), a)}\,\forall\colon l}{T, (\forall x)P(f(x), x) \vdash P(f(a), a)}\,w\colon l \quad \right\}$$

Therefore the ACNF of Ψ for $\alpha = 0$ is simply the projection from $\mathrm{PR}(\Psi){\downarrow_0}$. For $\alpha = 1$, $R{\downarrow_1}$ is

$$\dfrac{\vdash P(f(u_0'), u_0') \qquad \dfrac{\dfrac{\vdash P(f(u_0), u_0) \qquad P(f^2(v_1), f(v_1)), P(f(v_1), v_1) \vdash P(f^2(v_1), v_1)}{P(f(v_1), v_1) \vdash P(f^2(v_1), v_1)}\,u_0/f(v_1)}{\vdash P(f^2(v_1), v_1)}\,u_0'/v_1 \qquad P(f^2(a), a) \vdash}{\vdash}\,v_1/a$$

and $T(R{\downarrow_1})$ is

$$\dfrac{\vdash P(f(a), a) \qquad \dfrac{\dfrac{\vdash P(f^2(a), f(a)) \qquad P(f^2(a), f(a)), P(f(a), a) \vdash P(f^2(a), a)}{P(f(a), a) \vdash P(f^2(a), a)}\,cut}{\vdash P(f^2(a), a)}\,cut \qquad P(f^2(a), a) \vdash}{\vdash}\,cut$$

Now, the set of projections $\mathrm{PR}(\Psi)\downarrow_1$, contains exactly the projections $\phi_{C_1}, \phi_{C_2}, \phi_{C_3}$, for

$$\begin{aligned} C_1 &= \vdash P(f(u_0), u_0) \\ C_2 &= P(f^2(u_1), f(u_1)), P(f(u_1), u_1) \vdash P(f^2(u_1), u_1) \\ C_3 &= P(f^2(a), a) \vdash \end{aligned}$$

which are respectively:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(f(u_0),u_0)\vdash P(f(u_0),u_0)}{(\forall x)P(f(x),x)\vdash P(f(u_0),u_0)}\forall\colon l}{T,(\forall x)P(f(x),x)\vdash P(f(u_0),u_0)}w\colon l}{T,T,(\forall x)P(f(x),x)\vdash P(f(u_0),u_0)}w\colon l}{T,(\forall x)P(f(x),x)\vdash P(f(u_0),u_0)}c\colon l}{T,(\forall x)P(f(x),x)\vdash P(f(u_0),u_0),P(f^2(a),a)}w\colon r \quad ;$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(f^2(u_1),f(u_1))\vdash P(f^2(u_1),f(u_1)) \qquad P(f(u_1),u_1)\vdash P(f(u_1),u_1)}{P(f^2(u_1),f(u_1)),P(f(u_1),u_1)\vdash P(f^2(u_1),f(u_1))\wedge P(f(u_1),u_1)}\wedge\colon r \qquad P(f^2(u_1),u_1)\vdash P(f^2(u_1),u_1)}{(P(f^2(u_1),f(u_1))\wedge P(f(u_1),u_1)\supset P(f^2(u_1),u_1),P(f^2(u_1),f(u_1)),P(f(u_1),u_1)\vdash P(f^2(u_1),u_1)}\supset\colon l}{T,P(f^2(u_1),f(u_1)),P(f(u_1),u_1)\vdash P(f^2(u_1),u_1)}\forall\colon l*}{T,T,(\forall x)P(f(x),x),P(f^2(u_1),f(u_1)),P(f(u_1),u_1)\vdash P(f^2(u_1),u_1)}w\colon l*}{T,(\forall x)P(f(x),x),P(f^2(u_1),f(u_1)),P(f(u_1),u_1)\vdash P(f^2(u_1),u_1)}c\colon l}{T,(\forall x)P(f(x),x),P(f^2(u_1),f(u_1)),P(f(u_1),u_1)\vdash P(f^2(u_1),u_1),P(f^2(a),a)}w\colon r \quad ;$$

$$\dfrac{P(f^2(a),a)\vdash P(f^2(a),a)}{T,(\forall x)P(f(x),x),P(f^2(a),a)\vdash P(f^2(a),a)}w\colon l* \quad ;$$

Next, we denote $A = (\forall x)P(f(x),x)$ and $B = P(f^2(a),a)$, then the the ACNF of Ψ for $\alpha = 1$ is

$$\dfrac{\dfrac{\dfrac{(\phi_{C_1}[u_0/a])}{T,A\vdash B,P(f(a),a)} \qquad \dfrac{(\phi_{C_1}[u_0/f(a)]) \qquad (\phi_{C_2}[u_1/a])}{T,A\vdash B,P(f^2(a),f(a)) \qquad T,A,P(f^2(a),f(a)),P(f(a),a)\vdash B,P(f^2(a),a)}cut}{T,T,A,A,P(f(a),a)\vdash B,B,P(f^2(a),a)}cut \qquad (\phi_{C_3})\ T,A,P(f^2(a),a)\vdash B}{T,T,T,A,A,A\vdash B,B,B,P(f^2(a),a)}cut}{\dfrac{T,T,T,T,A,A,A,A\vdash B,B,B,B}{T,(\forall x)P(f(x),x)\vdash P(f^2(a),a)}c\colon l,r*}$$

Finally, from the resolution refutation schema, which is a skeleton of an (essentially) cut-free proof, and from the projection terms, namely from $\Xi(\nu_2(k),\Omega)$, it is easy to see that indeed every cut-free proof of S_n must contain initial sequents $P(f^{i+1}(a),f^i(a)) \vdash P(f^{i+1}(a),f^i(a))$ for all $i = 0,\ldots,2^n-1$.

Chapter 7

Conclusion and Future Work

It is well known that cut-elimination is not in general possible in the presence of induction. In the literature attempts were made to find restrictions which would allow cut-elimination. For example in [MM00] a reductive cut-elimination method was given for intuitionistic proof systems with induction. The other way was to avoid the use of the induction rule by so called cyclic proofs. But in [Bro05] it was mentioned that there is no cut-elimination theorem for classical cyclic proof systems in the literature. The reason is that the reductive cut-elimination methods cannot be extended to proofs with cycles (shifting cuts over cycles is a major problem).

We presented a cut-elimination method $CERES_s$ for some kind of cyclic proofs, which we call proof schemata and hence have shown that cut-elimination is possible also in proofs with cycles. To the best of our knowledge no other proof of cut-elimination theorem exists for cyclic proofs.

$CERES_s$ is an extension of the cut-elimination method CERES (which is cut-elimination method by resolution), removing all cuts at once, avoiding the problem of shifting cuts over cycles. It allows extractions of the characteristic clause set schema and of the projection schema from a proof schema. Then the resolution refutation schema of the characteristic clause set schema together with the projection schema describes a sequence of atomic cut normal forms.

The $CERES_s$ method, besides its theoretical importance, has also very useful practical applications. Therefore the method was implemented and applied to some schematic problems, which was illustrated in the previous chapter.

As it usually happens in research, solving one problem gives rise to several other problems. Therefore there are many potential improvements of $CERES_s$ left to future research. Here we present the most important open questions.

Extraction of Herbrand Sequent and Interpolants. There are procedures which extract valuable information from cut-free proofs, such as Herbrand sequent or interpolants. Considering the fact, that $CERES_s$ does not produce a cut-free sequence of proofs in usual notation, it is worthy to have such algorithms for proof schemata as well, which will allow the extraction of valuable information form the ACNF schema.

Skolemization and Deskolemization. As was discussed in Chapter 2 and Chapter 5, skolemization is vital for CERES, as well for $CERES_s$. Therefore the algorithm for skolemizing and deskolemizing a proof schema should be found.

Investigate the Resolution Calculus $\mathcal{R}_s$. There are numerous problems that can be investigated in $\mathcal{R}_s$. The first is to find a general algorithm, verifying that given resolution proof schema is indeed a resolution refutation schema of the given clause set schema. Moreover, the term language should be restricted such that the unification problem becomes decidable (and a unification algorithm can be obtained). This will allow to investigate decidable fragments of first-order schemata.

Investigate Fast Classes. Once we have decidable fragments of first-order schemata, fast cut-elimination classes can be investigated for proof schemata and the results from [BL10, Ruk11] can be extended to it.

Extend the Method to Multiple parameters. Having only one parameter is a major restriction of our language, since nested inductions cannot be handled. Therefore extending the calculus and the method to multiple parameters is of major importance.

Appendix A

Cut Transformation Rules

We will list below transformation rules for the Gentzen's method. Unlike Gentzen's original proof of cut-elimination theorem, we will use a *cut* rule instead of a *mix* rule. We divide rules into two parts, grade reduction rules and rank reduction rules. Note that these rules are complete for cut-elimination, provided an uppermost cut is selected.

A.1 Grade Reduction Rules

1. Cut rule premises are lower sequents of the $\wedge\colon r$ and $\wedge\colon l$ rules:

$$\dfrac{\dfrac{\begin{matrix}\phi_1\\ \Gamma_1 \vdash \Delta_1, A\end{matrix} \quad \begin{matrix}\phi_2\\ \Gamma_2 \vdash \Delta_2, B\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B}\wedge\colon r \quad \dfrac{\begin{matrix}\phi_r\\ A, B, \Gamma \vdash \Delta\end{matrix}}{A \wedge B, \Gamma \vdash \Delta}\wedge\colon l}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_1, \Delta_2, \Delta}\, cut$$

   transforms to

$$\dfrac{\begin{matrix}\phi_2\\ \Gamma_2 \vdash \Delta_2, B\end{matrix} \quad \dfrac{\dfrac{\begin{matrix}\phi_1\\ \Gamma_1 \vdash \Delta_1, A\end{matrix} \quad \begin{matrix}\phi_r\\ A, B, \Gamma \vdash \Delta\end{matrix}}{B, \Gamma_1, \Gamma \vdash \Delta_1, \Delta}\, cut}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_1, \Delta_2, \Delta}\, cut$$

2. Cut rule premises are lower sequents of the $\vee\colon r$ and $\vee\colon l$ rules:

$$\dfrac{\dfrac{\begin{matrix}\phi_l\\ \Gamma \vdash \Delta, A, B\end{matrix}}{\Gamma \vdash \Delta, A \vee B}\vee\colon r \quad \dfrac{\begin{matrix}\phi_1\\ A, \Gamma_1 \vdash \Delta_1\end{matrix} \quad \begin{matrix}\phi_2\\ B, \Gamma_2 \vdash \Delta_2\end{matrix}}{A \vee B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\vee\colon l}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\overset{\phi_l}{\Gamma \vdash \Delta, A, B} \qquad \overset{\phi_1}{A, \Gamma_1 \vdash \Delta_1}}{\Gamma, \Gamma_1 \vdash \Delta, \Delta_1, B}\, cut \qquad \overset{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, cut$$

3. Cut rule premises are lower sequents of the $\supset\colon r$ and $\supset\colon l$ rules:

$$\dfrac{\dfrac{\overset{\phi_l}{A, \Gamma \vdash \Delta, B}}{\Gamma \vdash \Delta, A \supset B} \supset\colon r \qquad \dfrac{\overset{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \qquad \overset{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{A \supset B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \supset\colon l}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\overset{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\overset{\phi_l}{A, \Gamma \vdash \Delta, B} \qquad \overset{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{A, \Gamma, \Gamma_2 \vdash \Delta, \Delta_2}\, cut}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, cut$$

4. Cut rule premises are lower sequents of the $\neg\colon r$ and $\neg\colon l$ rules:

$$\dfrac{\dfrac{\overset{\phi_l}{A, \Gamma_1 \vdash \Delta_1}}{\Gamma_1 \vdash \Delta_1, \neg A} \neg\colon r \qquad \dfrac{\overset{\phi_r}{\Gamma_2 \vdash \Delta_2, A}}{\neg A, \Gamma_2 \vdash \Delta_2} \neg\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\overset{\phi_r}{\Gamma_2 \vdash \Delta_2, A} \qquad \overset{\phi_l}{A, \Gamma_1 \vdash \Delta_1}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

5. Cut rule premises are lower sequents of the $\forall\colon r$ and $\forall\colon l$ rules:

$$\dfrac{\dfrac{\overset{\phi_l}{\Gamma_1 \vdash \Delta_1, A(x/\alpha)}}{\Gamma_1 \vdash \Delta_1, (\forall x)A} \forall\colon r \qquad \dfrac{\overset{\phi_r}{A(x/t), \Gamma_2 \vdash \Delta_2}}{(\forall x)A, \Gamma_2 \vdash \Delta_2} \forall\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_l(\alpha/t)}{\Gamma_1 \vdash \Delta_1, A(x/t)} \quad \dfrac{\phi_r}{A(x/t), \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

6. Cut rule premises are lower sequents of the $\exists\colon r$ and $\exists\colon l$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A(x/t)}\, \exists\colon r}{\Gamma_1 \vdash \Delta_1, (\exists x)A} \quad \dfrac{\dfrac{\phi_r}{A(x/\alpha), \Gamma_2 \vdash \Delta_2}\, \exists\colon l}{(\exists x)A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A(x/t)} \quad \dfrac{\phi_r(\alpha/t)}{A(x/t), \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

A.2 Rank Reduction Rules

1. Cut rule left premise is an axiom:

$$\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\phi_r}{A, \Gamma \vdash \Delta}}{A, \Gamma \vdash \Delta}\, cut \qquad \text{transforms to} \qquad \dfrac{\phi_r}{A, \Gamma \vdash \Delta}$$

2. Cut rule right premise is an axiom:

$$\dfrac{\dfrac{\phi_l}{\Gamma \vdash \Delta, A} \quad \dfrac{}{A \vdash A}}{\Gamma \vdash \Delta, A}\, cut \qquad \text{transforms to} \qquad \dfrac{\phi_l}{\Gamma \vdash \Delta, A}$$

3. Cut rule left premise is a lower sequent of the $w\colon r$ rule:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1}\, w\colon r}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1}\, w\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1}\, w\colon r*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}$$

If Γ_2 is empty and/or Δ_2 is empty, then corresponding weakening rules are omitted.

4. Cut rule right premise is a lower sequent of the $w\colon l$ rule:

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\dfrac{\phi_r}{\Gamma_2 \vdash \Delta_2}}{A, \Gamma_2 \vdash \Delta_2}\, w\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\phi_r}{\Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_2}\, w\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, w\colon r*$$

If Γ_1 is empty and/or Δ_1 is empty, then corresponding weakening rules are omitted.

5. Cut rule left premise is a lower sequent of the $c\colon r$ rule:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A, A}}{\Gamma_1 \vdash \Delta_1, A}\, c\colon r \quad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A, A} \quad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A}\, cut \quad \dfrac{\phi_r^1}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2, \Gamma_2 \vdash \Delta_1, \Delta_2, \Delta_2}\, cut}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, \Delta_2}\, c\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, c\colon r*$$

Where ϕ_r^1 is variant of ϕ_r, such that all eigenvariables are renamed. If Γ_2 and/or Δ_2 is empty, then corresponding contraction rules are omitted.

6. Cut rule right premise is a lower sequent of the $c\colon l$ rule:

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\dfrac{\phi_r}{A, A, \Gamma_2 \vdash \Delta_2}}{A, \Gamma_2 \vdash \Delta_2}\, c\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\phi_l^1}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\phi_r}{A, A, \Gamma_2 \vdash \Delta_2}}{A, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut}{\Gamma_1, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_1, \Delta_2}\, cut}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_1, \Delta_2}\, c\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, c\colon r*$$

Where ϕ_l^1 is variant of ϕ_l, such that all eigenvariables are renamed. If Γ_1 and/or Δ_1 is empty, then corresponding contraction rules are omitted.

7. Cut rule left premise is a lower sequent of an arbitrary unary rule:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma \vdash \Delta, A}\, unary}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_l}{\Gamma \vdash \Delta, A} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma, \Gamma_2 \vdash \Delta, \Delta_2}\, cut}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, unary$$

8. Cut rule right premise is a lower sequent of an arbitrary unary rule:

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\dfrac{\phi_r}{A, \Gamma \vdash \Delta}\, unary}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\phi_r}{A, \Gamma \vdash \Delta}\, cut}{\Gamma_1, \Gamma \vdash \Delta_1, \Delta}\, unary}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}$$

9. Cut rule left premise is a lower sequent of an arbitrary binary rule:

$$\dfrac{\dfrac{\overset{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \quad \overset{\phi_2}{\Gamma_2 \vdash \Delta_2}}{\Gamma \vdash \Delta, A}\, binary \quad \overset{\phi_r}{A, \Pi \vdash \Lambda}}{\Gamma, \Pi \vdash \Delta, \Lambda}\, cut$$

transforms to

$$\dfrac{\dfrac{\overset{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \quad \overset{\phi_r}{A, \Pi \vdash \Lambda}}{\Gamma_1, \Pi \vdash \Delta_1, \Lambda}\, cut \quad \overset{\phi_2}{\Gamma_2 \vdash \Delta_2}}{\Gamma, \Pi \vdash \Delta, \Lambda}\, binary$$

Or if cut formula comes from right premise of a binary rule:

$$\dfrac{\dfrac{\overset{\phi_1}{\Gamma_1 \vdash \Delta_1} \quad \overset{\phi_2}{\Gamma_2 \vdash \Delta_2, A}}{\Gamma \vdash \Delta, A}\, binary \quad \overset{\phi_r}{A, \Pi \vdash \Lambda}}{\Gamma, \Pi \vdash \Delta, \Lambda}\, cut$$

transforms to

$$\dfrac{\overset{\phi_1}{\Gamma_1 \vdash \Delta_1} \quad \dfrac{\overset{\phi_2}{\Gamma_2 \vdash \Delta_2, A} \quad \overset{\phi_r}{A, \Pi \vdash \Lambda}}{\Gamma_2, \Pi \vdash \Delta_2, \Lambda}\, cut}{\Gamma, \Pi \vdash \Delta, \Lambda}\, binary$$

10. Cut rule right premise is a lower sequent of an arbitrary binary rule:

$$\dfrac{\overset{\phi_l}{\Pi \vdash \Lambda, A} \quad \dfrac{\overset{\phi_1}{A, \Gamma_1 \vdash \Delta_1} \quad \overset{\phi_2}{\Gamma_2 \vdash \Delta_2}}{A, \Gamma \vdash \Delta}\, binary}{\Pi, \Gamma \vdash \Lambda, \Delta}\, cut$$

transforms to

$$\dfrac{\dfrac{\overset{\phi_l}{\Pi \vdash \Lambda, A} \quad \overset{\phi_1}{A, \Gamma_1 \vdash \Delta_1}}{\Pi, \Gamma_1 \vdash \Lambda, \Delta_1}\, cut \quad \overset{\phi_2}{\Gamma_2 \vdash \Delta_2}}{\Pi, \Gamma \vdash \Lambda, \Delta}\, binary$$

Or if cut formula comes from right premise of a binary rule:

$$\dfrac{\dfrac{\phi_l}{\Pi \vdash \Lambda, A} \qquad \dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1} \qquad \dfrac{\phi_2}{A, \Gamma_2 \vdash \Delta_2}\, binary}{A, \Gamma \vdash \Delta}}{\Pi, \Gamma \vdash \Lambda, \Delta}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1} \qquad \dfrac{\dfrac{\phi_l}{\Pi \vdash \Lambda, A} \qquad \dfrac{\phi_2}{A, \Gamma_2 \vdash \Delta_2}\, cut}{\Pi, \Gamma_2 \vdash \Lambda, \Delta_2}}{\Pi, \Gamma \vdash \Lambda, \Delta}\, binary$$

Appendix B

Characteristic Terms of the Adder Proof

We will list below the characteristic terms and (some of) the projections terms of the Adder proof, obtained from ProofTool. First, note that in ProofTool all these terms are displayed as trees upside-down, where the leaf nodes are clauses and the other nodes are the symbols $\oplus, \otimes, \sim$ (represents the negation) for characteristic terms and $\oplus, \otimes_\sigma, \rho$ for projection terms, where σ, ρ are the binary and unary rule names respectively.

In ProofTool the notation of cut-configurations also differs from the one that we used in the thesis. The cut-configurations are represented as lists of formulas, separated by |, that separates formulas from the antecedent of a sequent from the ones from the succedent. Here we list the cut-configurations of the proof schema Ψ from Section 6.2 in the notation of ProofTool:[1]

$$\begin{aligned}
\emptyset &= (|) \\
\Omega_\varphi &= \Big(\mid \bigwedge_{i=0}^{k} ((\neg C_i \vee Cp_i) \wedge (\neg Cp_i \vee C_i)) \Big) \\
\Omega_\phi &= (\mid ((\neg C_{k+1} \vee Cp_{k+1}) \wedge (\neg Cp_{k+1} \vee C_{k+1}))) \\
\Omega_\chi &= \Big(\bigwedge_{i=0}^{k} ((\neg C_i \vee Cp_i) \wedge (\neg Cp_i \vee C_i)) \mid \Big)
\end{aligned}$$

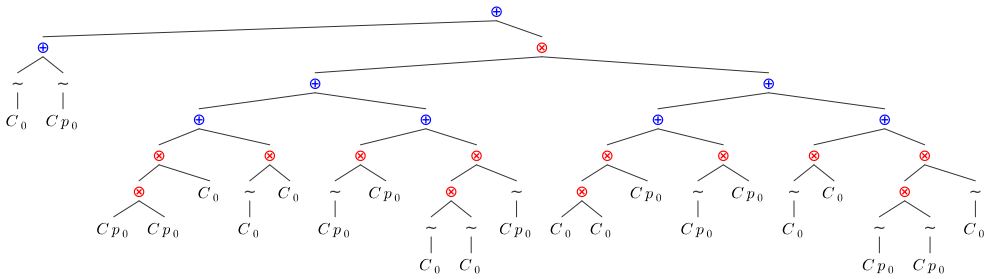
B.1 Characteristic Terms

Below we list the characteristic terms of the proof schema Ψ from Section 6.2. Note that for the better readability, in ProofTool the empty nodes of the

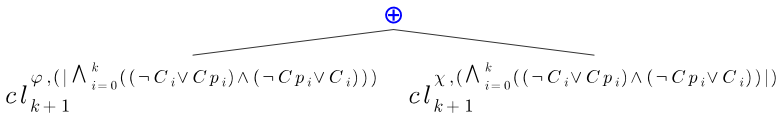
[1] The atom Cp corresponds to the atom C'.

characteristic terms are pruned.

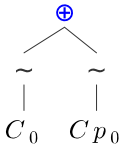
The characteristic term $\Theta(\pi_1, \emptyset)$:



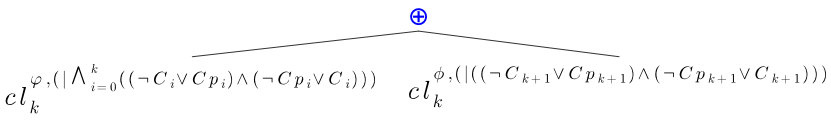
The characteristic term $\Theta(\nu_1(k), \emptyset)$:



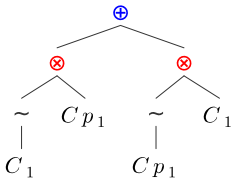
The characteristic term $\Theta(\pi_2, \Omega_\varphi)$:



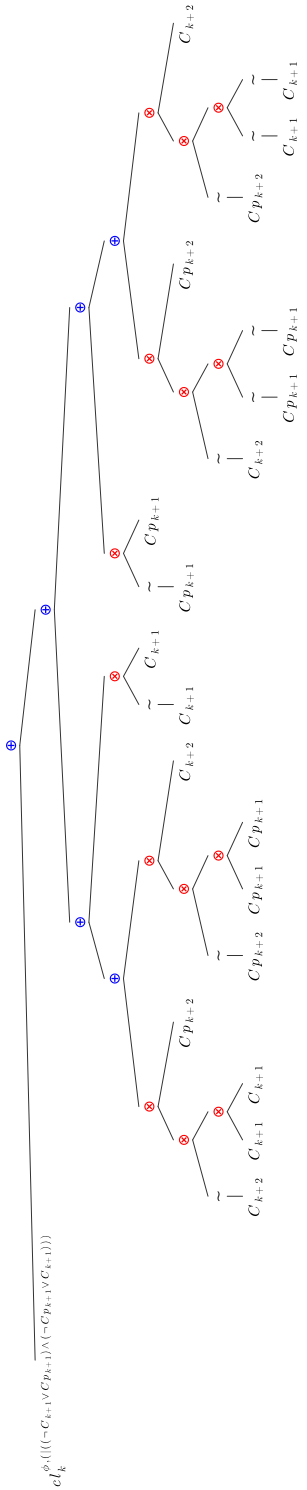
The characteristic term $\Theta(\nu_2(k), \Omega_\varphi)$:



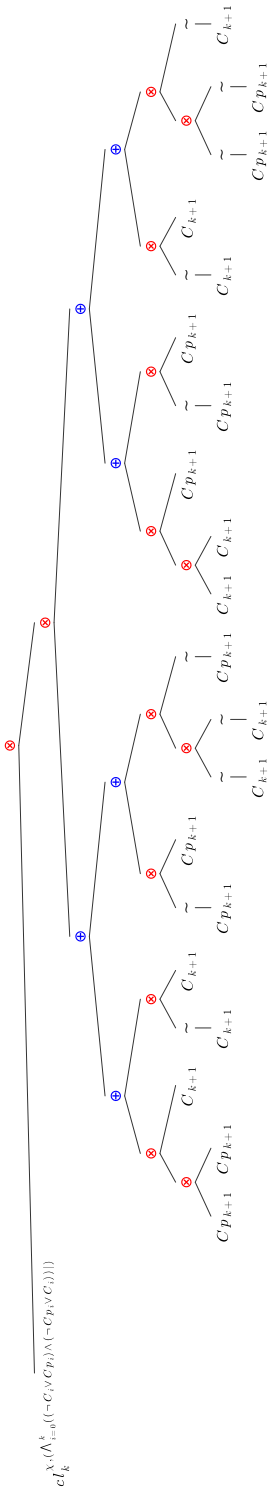
The characteristic term $\Theta(\pi_3, \Omega_\phi)$:



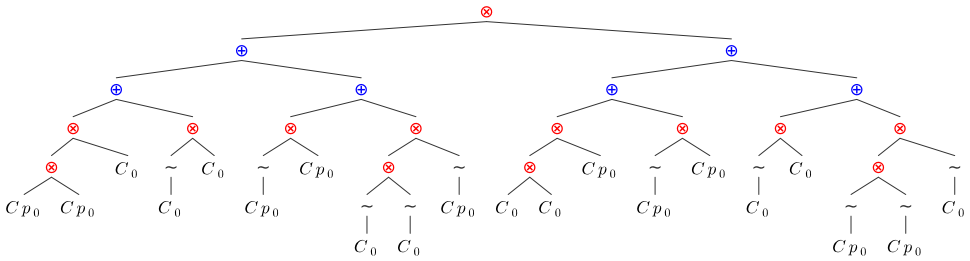
The characteristic term $\Theta(\nu_3(k), \Omega_\phi)$:



The characteristic term $\Theta(\nu_4(k), \Omega_\chi)$:



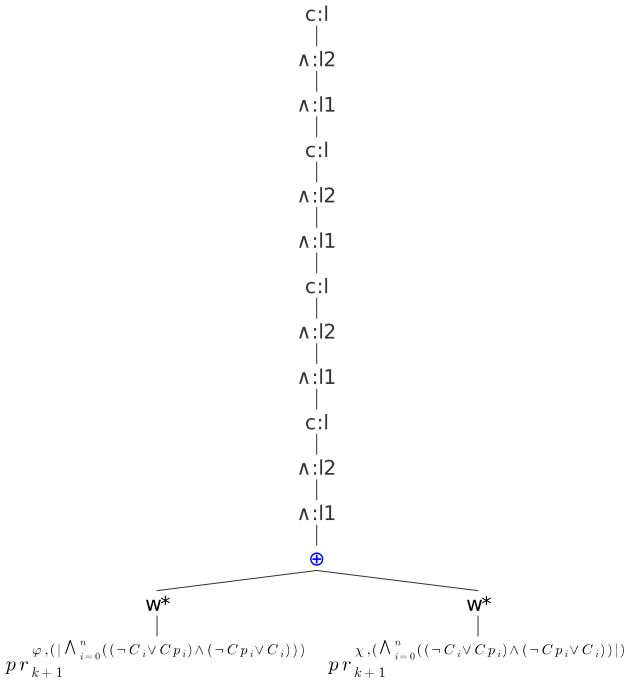
The characteristic term $\Theta(\pi_4, \Omega_\chi)$



B.2 Projection Terms

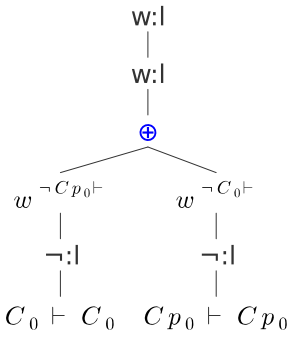
Below we list (some of[2]) the projection terms of the proof schema Ψ from Section 6.2. Note that the calculus of the GAPT framework is a bit different: our $\wedge\colon l$ and $\vee\colon r$ rules are represented as a chain of several unary rules, like for example the chains $\wedge\colon l1, \wedge\colon l2, c\colon l$ and $\vee\colon r1, \vee\colon r2, c\colon r$; therefore these rules appear in the projection terms instead of $\wedge\colon l$ and $\vee\colon r$.

The projection term $\Xi(\nu_1(k), \emptyset)$:

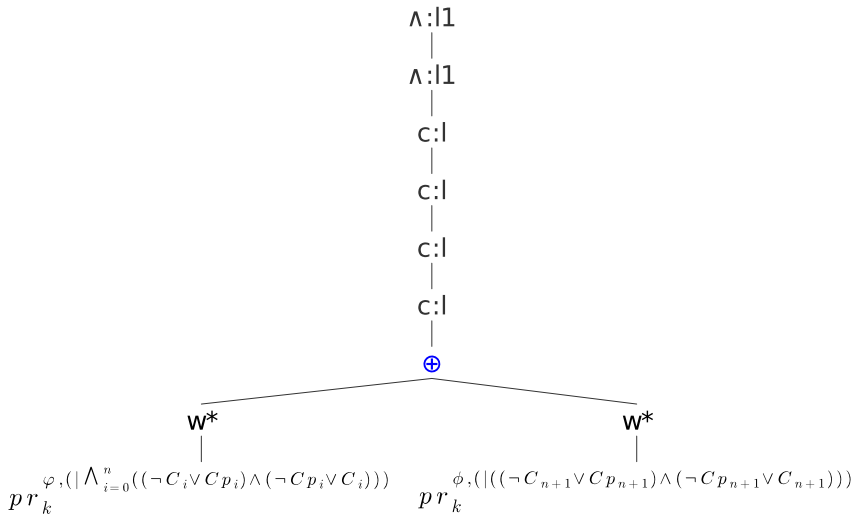


[2]The projection terms which are too big to be put on a page can be found at `http://www.logic.at/asap/`

The projection term $\Xi(\pi_2, \Omega_\varphi)$:



The projection term $\Xi(\nu_2(k), \Omega_\varphi)$:



Bibliography

[ACP08] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. More Flexible Term Schematisations via Extended Primal Grammars. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM 2008)*, 2008.

[ACP09] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A Schemata Calculus for Propositional Logic. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 5607 of *Lecture Notes in Computer Science*, pages 32–46, 2009.

[ACP10] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. RegSTAB: A SAT-Solver for Propositional Iterated Schemata. In *International Joint Conference on Automated Reasoning*, pages 309–315, 2010.

[ACP11] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Decidability and Undecidability Results for Propositional Schemata. *Journal of Artificial Intelligence Research*, 40:599–656, 2011.

[AEP] Vincent Aravantinos, Mnacho Echenim, and Nicolas Peltier. A Resolution Calculus for First-Order Schemata. To appear.

[AP11] Vincent Aravantinos and Nicolas Peltier. Generating Schemata of Resolution Proofs. In Martin Giese and Roman Kuznets, editors, *TABLEAUX 2011 Workshops, Tutorials, and Short Papers*, pages 16–30, 2011.

[AZ99] Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer, 1999.

[BHL+05] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Cut-Elimination: Experiments with CERES. In Franz Baader and Andrei Voronkov, editors, *Logic*

for Programming, Artificial Intelligence, and Reasoning (LPAR) 2004, volume 3452 of *Lecture Notes in Computer Science*, pages 481–495. Springer, 2005.

[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*, 403:160–175, 2008.

[BHW12] Matthias Baaz, Stefan Hetzl, and Daniel Weller. On the Complexity of Proof Deskolemization. *Journal of Symbolic Logic*, 77(2):669–686, 2012.

[BL94] Matthias Baaz and Alexander Leitsch. Skolemization and proof complexity. *Fundamenta Informaticae*, 20(4):353–379, 1994.

[BL99] Matthias Baaz and Alexander Leitsch. Cut Normal Forms and Proof Complexity. *Annals of Pure and Applied Logic*, 97:127–177, 1999.

[BL00] Matthias Baaz and Alexander Leitsch. Cut-Elimination and Redundancy-Elimination by Resolution. *Journal of Symbolic Computation*, 29(2):149–176, 2000.

[BL06] Matthias Baaz and Alexander Leitsch. Towards a Clausal Analysis of Cut-Elimination. *Journal of Symbolic Computation*, 41(3-4):381–410, 2006.

[BL10] Matthias Baaz and Alexander Leitsch. Fast Cut-Elimination by CERES. In S. Feferman and W. Sieg, editors, *Proofs, Categories and Computations*, pages 31–49. College Publications, London, 2010.

[BL11] Matthias Baaz and Alexander Leitsch. *Methods of Cut-Elimination*, volume 34 of *Trends in Logic*. Springer, 2011.

[Bro05] James Brotherston. Cyclic Proofs for First-Order Logic with Inductive Definitions. In B. Beckert, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 3702 of *Lecture Notes in Computer Science*, pages 78–92, 2005.

[CL08] Agata Ciabattoni and Alexander Leitsch. Towards an Algorithmic Construction of Cut-Elimination Procedures. *Mathematical Structures in Computer Science*, 18:81–105, 2008.

[DLL+] Cvetan Dunchev, Alexander Leitsch, Tomer Libal, Martin Riener, Mikheil Rukhaia, Daniel Weller, and Bruno Woltzenlogel-Paleo. ProofTool: GUI for the GAPT Framework. To appear.

[DLL+10] Tsvetan Dunchev, Alexander Leitsch, Tomer Libal, Daniel Weller, and Bruno Woltzenlogel Paleo. System Description: The Proof Transformation System CERES. In *Automated Reasoning*, pages 427–433. Springer Berlin / Heidelberg, 2010.

[DLL+12] Cvetan Dunchev, Alexander Leitsch, Tomer Libal, Martin Riener, Mikheil Rukhaia, Daniel Weller, and Bruno Woltzenlogel-Paleo. System Feature Description: Importing Refutations into the GAPT Framework. In David Pichardie and Tjark Weber, editors, *Second International Workshop on Proof Exchange for Theorem Proving (PxTP 2012)*, volume 878 of *CEUR Workshop Proceedings*, pages 51–57, 2012.

[DLRW] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. CERES for First-Order Schemata. To appear.

[DLRW12] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. CERES for Propositional Proof Schemata. Technical report, Vienna University of Technology, 2012.

[Dun12] Cvetan Dunchev. *Automation of Cut-Elimination in Proof Schemata*. PhD thesis, Vienna University of Technology, 2012.

[FMWP10] Pascal Fontaine, Stephan Merz, and Bruno Woltzenlogel Paleo. Exploring and Exploiting Algebraic and Graphical Properties of Resolution. In *8th International Workshop on Satisfiability Modulo Theories - SMT 2010*, Edinburgh, Royaume-Uni, 2010.

[Het08] Stefan Hetzl. *Proof Profiles. Characteristic Clause Sets and Proof Transformations*. VDM-Verlag, Saarbrücken, 2008.

[HLW11] Stefan Hetzl, Alexander Leitsch, and Daniel Weller. CERES in Higher-order Logic. *Annals of Pure and Applied Logic*, 162(12):1001–1034, 2011.

[HLWP09] Stefan Hetzl, Alexander Leitsch, Daniel Weller, and Bruno Woltzenlogel Paleo. A Clausal Approach to Proof Analysis in Second-order Logic. In Sergei Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*,

volume 5407 of *Lecture Notes in Computer Science*, pages 214–229. Springer Berlin, 2009.

[Lei97] Alexander Leitsch. *The resolution calculus*. Texts in theoretical computer science. Springer-Verlag Inc., New York, NY, USA, 1997.

[Lib08] Tomer Libal. Cut Elimination in Inductive Proofs of Weakly Quantified Theorems. Master's thesis, Vienna University of Technology, 2008.

[LRP12] Alexander Leitsch, Giselle Reis, and Bruno Woltzenlogel Paleo. Towards CERES in Intuitionistic Logic. In Patrick Cégielski and Arnaud Durand, editors, *Computer Science Logic (CSL'12)*, volume 16 of *LIPIcs*, pages 485–499. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[MM00] Raymond McDowell and Dale Miller. Cut-Elimination for a Logic with Definitions and Induction. *Theoretical Computer Science*, 232(1–2):91–119, 2000.

[MR67] Albert R. Meyer and Dennis M. Ritchie. The Complexity of Loop Programs. In *Proceedings of the 1967 22nd national conference*, ACM '67, pages 465–469, New York, NY, USA, 1967.

[Ore82] Vladimir P. Orevkov. Lower Bounds for Increasing Complexity of Derivations After Cut-Elimination. *Journal of Mathematical Sciences*, 20:2337–2350, 1982.

[OSV10] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala: A Comprehensive Step-by-step Guide*. Artima, Inc., 2nd edition, 2010.

[Pal09] Bruno Woltzenlogel Paleo. *A General Analysis of Cut-Elimination by CERes*. PhD thesis, Vienna University of Technology, 2009.

[Rob65] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Ruk11] Mikheil Rukhaia. *CERES and Fast Cut-Elimination. A monograph*. VDM-Verlag, Saarbrücken, 2011.

[Sal93] Gernot Salzer. On the Relationship between Cycle Unification and the Unification of Infinite Sets of Terms. In Franz Baader and Wayne Snyder, editors, *7th Workshop on Unification (UNIF'93)*, Boston, Mass., USA, 1993.

[Tak87] Gaisi Takeuti. *Proof Theory*. North Holland, second edition, 1987.

[Tsi70] Dionysis Tsichritzis. The Equivalence Problem of Simple Programs. *Journal of the ACM*, 17(4):729–738, 1970.

Index