Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

TU UB WIEN Universitätsbibliothek

The approved original version of this thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Algorithmic Introduction of Π_2-Cuts

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Michael Peter Lettmann, MSc
Matrikelnummer 1429618

an der Fakultät für Informatik

der Technischen Universität Wien

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

Diese Dissertation haben begutachtet:

Jeremy Avigad

Dale Miller

Alexander Leitsch

Wien, 5. September 2018

Michael Peter Lettmann

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Algorithmic Introduction of Π_2 Cuts

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Michael Peter Lettmann, MSc
Registration Number 1429618

to the Faculty of Informatics

at the TU Wien

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

The dissertation has been reviewed by:

Jeremy Avigad

Dale Miller

Alexander Leitsch

Vienna, 5th September, 2018

Michael Peter Lettmann

Erklärung zur Verfassung der Arbeit

Michael Peter Lettmann, MSc
Belghofergasse 38, Top 1
1120 Wien
Österreich

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

Wien, 5. September 2018

Michael Peter Lettmann

Danksagung

An dieser Stelle möchte ich meinen Dank an Alexander Leitsch aussprechen, der mich als Betreuer durch mein Doktorratsstudium begleitete und mir stets mit seiner Expertise zur Seite stand. Ich hoffe, ich konnte seinen hohen Ansprüchen an die Qualität der Forschung sowie an die Pünktlichkeit gerecht werden.

Ferner möchte ich mich bei meinen Gutachtern Dale Miller und Jeremy Avigad bedanken, deren Vorschläge nicht nur die Qualität dieser Arbeit enorm verbesserten, sondern auch meine Sensibilität für verständliches Schreiben erhöhten.

Ein großer Dank gilt auch denen, die, bei verschiedenen Anlässen, Korrektur lasen. Namentlich sind das Vanessa Knöll, Marc Lettmann, David Michael Cerna, Francesco Antonio Genco, Timo Lang, Alexander Leitsch, Björn Lellmann und Roman Kuznets.

Nicht unerwähnt bleiben soll Gabriel Ebner, dem ich meine Programmierkenntnisse in Scala verdanke und der mir bei der Ausführung der Massentests half. Ebenso half mir Stefan Hetzl beim Verstehen der Π_1-Schnitteinführung.

Während meines Studiums hatte ich die Chance die Professoren Alan Bundy und Nicolas Peltier an deren jeweiliger Wirkungsstätte zu besuchen. Für die dort gewonnenen Eindrücke und die erhaltene Hilfe bin ich sehr dankbar.

Des Weiteren war es mir eine Freude mit den Kollegen aus dem vierten Stock zu arbeiten. Die Atmosphäre war sehr angenehm und man konnte sich auch außerhalb der Arbeitszeit treffen. Danke an Francesco alias Franz, Paolo alias Paulchen, David, Anela, Timo, Tim, Francesca, Matthias, Kees, Matteo, Michael, Revantha, Roman, Björn, Esther und Vlasta. Danke auch an die Kollegen aus dem Doktorratskolleg und aus dem Freihaus.

Ich möchte Juliane Auerböck, Beatrix Buhl, Eva Nedoma und Anna Prianichnikova für ihre gorße Hilfe danken.

Die Möglichkeit zur Dissertation verdanke ich dem Doktorratskolleg LogiCS, genauer gesagt, dem FWF Projekt W1255-N23.

Zu guter Letzt möchte ich mich bei meinen Eltern, meinem Bruderherz und meiner Verlobten bedanken, die mich trotz der großen Distanz in jeglicher Hinsicht unterstützten.

Acknowledgements

At this point, I would like to extend my thanks to Alexander Leitsch who lead me through my doctoral studies and assisted me with his great expertise. I hope that I met his high demands on the quality of research and punctuality.

Moreover, I would like to thank my reviewers Dale Miller and Jeremy Avigad. Their comments did not only improve the quality of this thesis, but also my ability to convey my thoughts to the reader through writing.

Great thanks to those who proof read what I wrote during my studies. Namely, Vanessa Knöll, Marc Lettmann, David Michael Cerna, Francesco Antonio Genco, Timo Lang, Alexander Leitsch, Björn Lellmann, and Roman Kuznets.

I would also like to mention Gabriel Ebner whom I have to thank for my programming skills in scala. He also was of great assistance for running the tests of the implementation, as well as Stefan Hetzl helped me understanding the Π_1-cut introduction method.

While studying I had the chance to visit the Professors Alan Bundy and Nicolas Peltier at their institutions. I am very thankful for the impression the experience left on me.

Furthermore, it was a great pleasure to work with the colleagues of the fourth floor. The atmosphere was very inviting and the thesis benefited also from the time we spent apart from work. Thanks to Francesco aka Franz, Paolo aka Paulchen, David, Anela, Timo, Tim, Francesca, Matthias, Kees, Matteo, Michael, Revantha, Roman, Björn, Esther, and Vlasta. Thanks to the colleagues from the doctoral college and the Freihaus, too.

I would like to thank Juliane Auerböck, Beatrix Buhl, Eva Nedoma, and Anna Prianichnikova for the great help they provided.

I have to thank the doctoral college also for the opportunity to pursue my Ph.D. studies and the FWF project W1255-N23 providing support and funding.

Last but not least, I would like to thank my parents, my brother, and my fiancée, who were supporting me despite the huge geographical distance between us.

Kurzfassung

Eines der wesentlichen Resultate innerhalb der Beweistheorie ist Gentzens 'Hauptsatz', der auch als Schnitteliminationstheorem bekannt ist. Dabei bezieht sich die Schnittelimination auf eine Technik, welche den notwendigen Teil eines Lemmas in den Beweis einbindet, um so die zusätzliche Struktur des Lemmas zu entfernen. Der daraus resultierende schnittfreie Beweis erlaubt uns, den berechenbaren Inhalt zu untersuchen, welcher wiederum von fundamentalen Interesse für das Forschungsgebiet 'Proof Mining' sowie für das automatische Generieren von Beweisen ist. Schnittfreie Beweise zeichnen sich durch ein analytisches Verhalten aus, welches hauptsächlich an der Teilformeleigenschaft deutlich wird. So tauchen im Beweis lediglich Formeln auf, die zugleich Teilformeln der zu beweisenden Aussage sind. Ein sich aus der Schnittfreiheit ergebender Nachteil ist die enorme Größe solcher Beweise, da die den Lemmas eigene Struktur den Beweis stark verkleinern kann.

Aus diesem Grund streben wir in dieser Arbeit eine Umkehrung des Schnitteliminationsalgorithmuses an, welche auf einer Verbindung zwischen der Theorie der formalen Grammatiken und der Beweistheorie basiert. So werden wir Grammatiken charakterisieren, die Beweisen mit Lemmas einer Stufe der arithmetischen Hierachie bis hin zu Π_2 Formeln zugeordnet werden. Unter der Annahme einer solchen Grammatik versuchen wir die schnittfreien Beweise so umzuschreiben, dass sie nun Schnitte enthalten und dabei ihre Größe verringert wird. Hierzu betrachten wir die bereits bekannte Einführung von Π_1 Schnitten, präsentieren einen Algorithmus zur Einführung von Π_2 Schnitten, dessen Vollständigkeit für ein relevantes Fragment und die Ergebnisse die mithilfe einer Implementierung erzielt wurden. Schlussendlich können wir zeigen, dass durch die Π_2 Schnitteinführung eine exponentielle Kompression erreicht werden kann.

Abstract

One of the crucial results in proof theory is Gentzen's 'Hauptsatz', also known as the cut-elimination theorem. Cut-elimination is a technique to incorporate only the necessary content of lemmas within a formal proof into the proof while eliminating the additional structure of the lemma. The resulting object, a cut-free proof, gives insights about the computational content of a proof and is of major interest for subjects such as proof mining and automated theorem proving. Such proofs show an analytic behaviour mainly discernible in the subformula property which tells us that within the proof only subformulas of the statement are used. A drawback is the large size of cut-free proofs, due to missing structure expressible by lemmas.

In this thesis, we propose an inversion of the cut-elimination method based on a connection of formal grammars and proof theory. We specify characteristic grammars for cut formulas according to the arithmetical hierarchy up to Π_2 formulas and discuss whether the existence of such a grammar allows us to rewrite a given cut-free proof, now with Π_2-cut formulas, in order to reduce the proof size. Thereby, we revisit Π_1-cut introduction, present an algorithm to introduce Π_2 cuts which is shown to be complete for a fragment, and discuss the decidability of the problem whether Π_2 cuts exist for a given grammar. Moreover, we show that our method of Π_2-cut introduction achieves an exponential compression of the proof size.

"Ich erblicke dagegen gerade in der Möglichkeit, solche Wahrheiten auf andere, einfachere zurückzuführen, mag die Reihe der Schlüsse noch so lang und scheinbar künstlich sein, einen überzeugenden Beweis dafür, daß ihr Besitz oder der Glaube an sie niemals unmittelbar durch innere Anschauung gegeben, sondern immer nur durch eine mehr oder weniger vollständige Wiederholung der einzelnen Schlüsse erworben ist."

Richard Dedekind, Was sind und was sollen die Zahlen? [Ded87]

Contents

CHAPTER 1

Introduction

In the history of mathematical logic, the beginning of proof theory is often dated back to Hilbert's program [Hil99, Hil00], even though the works by Frege [Fre79, Fre84], Peano [Pea89a, Pea89b], Pasch [Pas82], Fano [Fan91] and Dedekind [Ded87] established already many of its topics (see also [Ken72]). Apart from other tasks, Hilbert tried to formalize mathematics in such a way that all true statements can be proven without loosing consistency. But, as shown by Gödel [Göd31] the main obstacle cannot be overcome, i.e. theories at least as strong as Peano arithmetic cannot be shown complete. Nonetheless, a formalization of major parts of mathematics can be defined (see for example Zermelo-Fraenkel set theory [Kun14]), the completeness of first-order logic is provable [Göd29], and consistency results for theories such as Peano arithmetic [Gen35b] or even more powerful subsets of second-order logic [Tak67] exist. One of the most celebrated contribution among those is Gentzen's paper [Gen35a] in which he describes a decision method for intuitionistic logic and gives new proofs for the consistency of first-order logic and the consistency of arithmetic without an induction schema via his 'Hauptsatz', also known as the cut-elimination theorem. In order to achieve this, he developed a calculus modifying sequents, a binary relation of multisets of formulas, in which all provable and only the provable formulas of first-order logic can be derived. The rules of the calculus except the cut rule share an analytic behaviour, i.e. the premises of a rule only contain subformulas of the conclusion. In the cut rule, one can introduce formulas independently and thereby, one can model concepts such as mathematical lemmas in a formal way. By proving that every application of the cut rule can be eliminated (cut-elimination), Gentzen concluded that there is always a proof only containing subformulas of the theorem to be proven. Since a proof of inconsistency of first-order logic corresponds to a proof of a sequent only containing bottom, a symbol representing falsity, and no sequent-calculus rule apart the cut rule can be applied to the sequent only containing bottom, there is no proof of inconsistency. The same reasoning plays a major role in Gentzen's consistency proof of Peano arithmetic as shown by Takeuti [Tak87].

Cut-elimination as presented by Gentzen (also called reductive cut-elimination) is a stepwise application of reduction steps in which the cut formula, i.e. the formula introduced into the proof by the cut rule, or its complexity is simplified until it eventually vanishes. The resulting object does rarely appear in this form in mathematics but can be used for proof mining (see, e.g., [Lei15] or [Koh08] for proof mining in general) or, as already mentioned, for consistency proofs [Gen35a, Sch77].

Moreover, sequent calculus inspired many other research areas, e.g. automated theorem proving. Tableau provers originated by inverting the applications of sequent-calculus rules [Häh01] to find proofs of first-order statements. Here, the analytic behaviour of the rules allows a goal-oriented search, although the task is semidecidable. A major drawback of these provers is the large size of constructed proofs, a fact that is also explained by cut-elimination: cut-free proofs are in general much bigger than proofs with cut. For example: a single application of the cut rule with a formula of the form $\forall x \exists y C$, C being a quantifier-free formula, can reduce the size exponentially (see Section 4.9). Hereby, one of the benefits of cut rules is illustrated, namely the formalization of patterns of a proof in a compact form. Moreover, proofs with cuts are usually more structured and are therefore more human readable. Since proofs constructed by automated theorem provers are cut free (tableau methods) or contain at most simple universally closed disjunctions of literals (resolution methods), we suggest a postprocessing in which the proofs gain additional structure by the introduction of cuts.

In this thesis, we present algorithmic techniques to introduce cuts into cut-free proofs in order to compress the proof size, structure the proof, and improve human readability. In particular, we consider cut formulas of the shape $\forall x C$ (see Chapter 3) and formulas of the shape $\forall x \exists y D$ (see Chapter 4) where C and D are quantifier free. Both techniques are based on a connection of proof theory and formal grammars [Het11, AHL15]. Proofs with cut can be translated into tree grammars such that the production rules of the grammar represent the term instantiations of the cut formula within the proof. By computing the language of the grammar, one receives a Herbrand term set, i.e. a term representation of a Herbrand sequent of the proven statement. A Herbrand sequent is a propositional tautology consisting only of instances of the statement in consideration. Therefore, Herbrand sequents contain all the necessary information of a cut-free proof. This means that cut elimination can be done via grammars, i.e. cut-elimination can be interpreted as the computation of a language by a grammar. This leads to the question whether the method can be inverted. Assume a cut-free proof of a statement and that we found a grammar whose language covers a Herbrand sequent of the cut-free proof. Can we construct a proof with cut? Or in different words: Is cut elimination invertible?

In this work, we present all methods to invert cut elimination that appear in literature which consist of the mentioned variants for formulas of the shape $\forall x C$ and $\forall x \exists y D$. Nonetheless, there are various methods introducing cuts or similarly compressing structures into proofs. Closest to this work are other approaches which abbreviate or structure given input proofs: in [WP10] an algorithm for the introduction of atomic cuts being capable of exponential proof compression is developed. There exist several contributions

to proof compression by cut introduction in propositional logic: a method defined in [FG07] is shown to never increase the size of proofs more than polynomially. Another approach to the compression of first-order proofs is based on the introduction of definitions for abbreviating terms and can be found in [VSU10].

Apart from inverting cut elimination, the introduction method presented here is in a sense capable of creating non-analytic deduction steps, a tool required for automated induction provers [Bun01]. Therefore, the generation of cuts or lemmas (to stress the connection to mathematics) can also be motivated in a bigger framework, i.e. the analysis of induction proofs and, in particular, their induction invariants. In fact, the computation of non-analytic cuts may yield induction invariants for automated induction provers. Similar work in this fashion is done in [BBHI05] where rippling, a method based on failed proof attempts, is defined. Furthermore, [Col01] and [Col02] adopt an eager approach to lemma generation in automated theory formation.

Occurring notations are consistent throughout the chapters. The most frequently used symbols have an entry in Table 2.1 in Section 2.1. Moreover, there is an index chapter attached which refers to the page of the corresponding definition. If the definition of a concept has a corresponding symbol then there is a subentry referring to all occurrences of this symbol. The structure of the work is as follows: In Chapter 2, we introduce basic concepts that are well-known in the area of formal logic. Chapter 3 presents the results known for Π_1-cut introduction that are the basis for the present work. In Chapter 4, we discuss Π_2-cut introduction. Thereby, we define the main obstacle (Section 4.4) analogously to the main obstacle of Π_1-cut introduction (Section 3.3). In contrast to the Π_1 case, we show that the Π_2-cut introduction problem is not always solvable (Section 4.5) and proceed with a characterization of its solvability (Section 4.6). Afterwards, we define a method to find solutions for a fragment via a unification method (Section 4.7). Since the former discussion was restricted to a Π_2-cut formula without blocks of quantifiers, we show in Section 4.8 that the results can be extended to more general cut formulas. We conclude the chapter by proving the maximal possible compression of a single Π_2 cut which can be found with our method (Section 4.9), i.e. a exponential compression. Chapter 5 presents an implementation of the unification method defined in Section 4.7, the grammar generation algorithm used for computing $\mathbf{S}\Pi_2$-$\mathbf{G}$s (see Section 4.3 and Section 5.2), and the corresponding experiments (Section 5.3). The work is concluded by a discussion of the achieved results and the arisen questions (Chapter 6).

CHAPTER 2

Preliminaries

The following chapter is a collection of all necessary but well-known concepts of formal logic that are required for the understanding of the present work. Readers, that are familiar with the subject, might skip this chapter and use it as a reference to look up definitions and notations.

More precisely, the chapter is structured as follows: Section 2.1 consists of a list of notations. Whenever the reader might encounter a symbol in the present work he or she cannot recognize, he or she might use Table 2.1 or the Index. While the Index contains all defined concepts and a reference to the corresponding definition, Table 2.1 contains items that are definitions itself (hence, the Index refers also to Table 2.1) or gives the symbols of units of often used terminology such as: arbitrary terms are represented as r, s, t, or r_i. In Section 2.2, we present the ***G3c**-calculus* which is used to represent classical first-order proofs. In classical first-order logic, there are many normal forms allowing a simpler reasoning. In Section 2.3, we define the normal forms that are relevant for the present work. Section 2.4 serves as a survey of all complexity measures. Some of them will also be defined in the course of the following chapters, whereas the basic concepts are defined only in Section 2.4. In Section 2.5, we briefly discuss Herbrand's theorem and how we are going to use it. In contrast to the other sections of the current chapter, Section 2.6 is not of a proof theoretic character. It gives an introduction to formal tree grammars and presents some necessary properties/definitions.

2.1 Notations

In this section, we give a list of all basic notations used in the present work. Table 2.1 contains four columns where the first column shows the name, the second column shows the symbol for the concept, the third gives a short explanation or some remarks (if necessary), and the fourth column shows the symbols used for representatives. The

lines are in alphabetical order of the names. Note that in some cases, the used symbols for different concepts overlap. This will only occur if it is not misleading.

Name	**Symbol: Concept**	**Explanation/Remarks**	**Symbols: Representatives**
$F\vec{r}$		F is a formula of arity $\mathbf{l}(\vec{r})$ which is instantiated with $\vec{r}$. Represents $F(\vec{r})$.	
l-ary		A corresponding object with arity l.	
l-tuple		A tuple of length l.	
$\vec{r}\|_i$		The i-th term in a tuple of terms $\vec{r}$.	
$\forall x.F$		$\forall x.F$ is a shorthand for $\forall x(F)$.	
$\exists x.F$		$\exists x.F$ is a shorthand for $\exists x(F)$.	
Arity	$\mathbf{a}(\cdot)$	Gives the arity of an object, i.e. a natural number.	
Clauses			C, D, R, C_i
Clause sets			$\mathcal{C}, \mathcal{D}, \mathcal{C}_i$
Concatenation of Sequents	$S \circ T$	A new sequent consisting of the merged antecedents and succedents of the old sequents.	
Constant (natural number)		Natural numbers used for representing fixed parameters.	l, m, n, p, q, a, b
Constants		A constant in the term language.	a, b, c, d
Context of a sequent			$\Gamma, \Delta, \Lambda, \Pi$
Dual	$\overline{(\cdot)}$	If it is applied to a single object without a negation in front it adds a negation. Otherwise, it drops the negation in front. If it is applied to a set of objects, it is applied to every single element and outputs the set of results.	
Eigenvariables		Variables introduced by a strong quantifier rules ($r\colon\forall$,$l\colon\exists$). See also the entry "Variables".	$\alpha, \beta, \gamma, \alpha_i$

Formulas			$F, G, A, B, C, D, E, H, X$
Functions		A function in the term language	f, g, h, f_i
Indices		Natural numbers used for representing indices.	i, j, k
Length	$\mathbf{l}(\cdot)$	Gives the length of a tuple, i.e. maps the tuple to the number of objects in the tuple.	
Literals			L, M, N, K, H, W
Natural numbers	$\mathbb{N}$	See also the entries "Constant (natural number)" and "Indices".	$l, m, n, p, q, a, b, i, j, k$
Positions			p, q
Predicates			P, Q, R
Proofs			φ, χ, ψ
Set of free variables	$\mathcal{F}(\cdot)$	Gives the set of variables not bounded by a quantifier within an object.	
Set of natural numbers	$\mathbb{N}_l$	The set of natural numbers $\{1, \ldots, l\}$	
Set of productions	$\alpha \to r_1, \ldots, r_l$	A shorthand for $\alpha \to r_1, \ldots, \alpha \to r_l$.	
Set of variables	$\mathcal{V}(\cdot)$	Gives the set of variables occurring in an object.	
Sets	$\{\cdot\}$		S, I, J, K
Sequents	$\Gamma \vdash \Delta$	A pair of multisets of formulas separated by $\vdash$ where the left multiset is referred to as *antecedent* and the right multiset is referred to as *succedent*.	$R, S, T, H, \mathcal{H}, \mathcal{S}$
Strong quantifiers		Quantifiers introduced by the $r\colon\forall$-rule or the $l\colon\exists$-rule.	
Substitutions	$[x\backslash r], [\vec{x}\backslash\vec{r}]$	Applied to an object, it replaces a variable or tuple of variables by a term or set of terms.	$\sigma, \tau, \upsilon, \sigma_i$
Terms			r, s, t, r_i
Tuples of terms			$\vec{r}, \vec{s}, \vec{t}, \vec{r}_i$
Variables		See also the entry "Eigenvariables".	x, y, z, x_i

Weak quantifiers		Quantifiers introduced by the $l:\forall$-rule or the $r:\exists$-rule.	

Table 2.1: Notations

Remark 1 *In the definition of a sequent (see Table 2.1), we could also use sets instead of multisets, since the **G3c**-calculus does not contain structural rules.*

2.2 G3c-Calculus

We will work with the **G3c**-calculus defined in Table 2.2. It is a system in which first-order proofs for classical statements can be represented and consists of a collection of rules. These rules allow us to manipulate sequents (see Table 2.1) that represent two multisets of formulas. Sequents can be interpreted as formulas itself, i.e. the formula that corresponds to a sequent $\Gamma \vdash \Delta$ is the conjunction of all formulas in Γ implying the disjunction of all formulas in Δ

$$\bigwedge_{A\in\Gamma} A \to \bigvee_{B\in\Delta} B.$$

In order to understand a sequent more intuitively, we can also read it as: In the presence/context of Γ, Δ is provable/derivable. In contrast to the more common **LK**-calculus (see [Gen35a] and [TS96, Chapter 3]), all the rules of the **G3c**-calculus are invertible (Proposition 1). This is a necessary condition for parts of our algorithm (cf. Sections 4.6 and 4.7).

Note that there is a slight difference between the original version of Kleene [Kle09] and the here presented version of Troelstra and Schwichtenberg inspired by Dragalin [Dra87, Part 1, §3]. In Kleene's system, all formulas that appear in the conclusion must also appear in the premise while, for instance, the formula $F \wedge G$ in the conclusion of $l:\wedge$ does not appear in the premise (see also the discussion in [TS96, Subsection 3.5.11.]).

For every rule we call the sequent at the bottom the *conclusion* and for every rule except Ax and $l:\bot$ we call the sequents on the top *premises*. In the following, we will think of proofs as trees. This will facilitate the description of our approach to find a solution for the schematic extended Herbrand sequent for Π_2 cuts (see Chapter 4). Hence, the leaves of a proof represent tautological axioms.

Note that in literature even trees where the leaves are non-tautological might be considered as proofs since the non-tautological leaves can be seen as additional assumptions of the proof. This implies a misleading usage of the word axiom. On the one hand the **G3c**-calculus contains the rule Ax, a shorthand for axiom. On the other hand every leaf of a proof tree is called an axiom, although it does not necessarily belongs to an application of the Ax rule. In this work, the notion axiom will always denote a leaf of a proof tree, whereas Ax and $l:\bot$, if necessary, will be called ***G3c**-axiom*. Moreover, we will introduce

$$Ax\ \frac{}{P, \Gamma \vdash \Delta, P\ (P \text{ atomic})} \qquad l{:}\bot\ \frac{}{\bot, \Gamma \vdash \Delta}$$

$$l{:}\wedge\ \frac{F, G, \Gamma \vdash \Delta}{F \wedge G, \Gamma \vdash \Delta} \qquad r{:}\wedge\ \frac{\Gamma \vdash \Delta, F \qquad \Gamma \vdash \Delta, G}{\Gamma \vdash \Delta, F \wedge G}$$

$$l{:}\vee\ \frac{F, \Gamma \vdash \Delta \qquad G, \Gamma \vdash \Delta}{F \vee G, \Gamma \vdash \Delta} \qquad r{:}\vee\ \frac{\Gamma \vdash \Delta, F, G}{\Gamma \vdash \Delta, F \vee G}$$

$$l{:}\to\ \frac{\Gamma \vdash \Delta, F \qquad G, \Gamma \vdash \Delta}{F \to G, \Gamma \vdash \Delta} \qquad r{:}\to\ \frac{F, \Gamma \vdash \Delta, G}{\Gamma \vdash \Delta, F \to G}$$

$$l{:}\forall\ \frac{\forall x F, F[x\backslash r], \Gamma \vdash \Delta}{\forall x F, \Gamma \vdash \Delta} \qquad r{:}\forall\ \frac{\Gamma \vdash \Delta, F[x\backslash\alpha]}{\Gamma \vdash \Delta, \forall x F}$$

$$l{:}\exists\ \frac{F[x\backslash\alpha], \Gamma \vdash \Delta}{\exists F, \Gamma \vdash \Delta} \qquad r{:}\exists\ \frac{\Gamma \vdash \Delta, \exists x F, F[x\backslash r]}{\Gamma \vdash \Delta, \exists x F}$$

where in $r{:}\forall$ and $l{:}\exists$ the α is not free in the conclusion.

Table 2.2: **G3c**-Calculus [TS96]

the notions of a **G3c**-derivation and a **G3c**-proof. The first of which can contain axioms apart form **G3c**-axioms and the latter will only contain **G3c**-axioms as leaves. For this reason, we will speak about tautological and non-tautological leaves. In a potential extension to a calculus containing equality, we would have to use the term quasi-tautology because a sequent of the form $\vdash r = r$ is not a tautology but a quasi-tautology.

Definition 1 *Let S be a sequent. We call an arbitrary tree a* **G3c**-derivation *of S if it has only sequents as nodes, has S as lowest element such that each edge corresponds to a rule of **G3c**, i.e. each node is an axiom or a conclusion of a rule of **G3c**, and the immediate successor(s) is/are the premise(s) of that rule.*

*Furthermore, we call a **G3c**-derivation a* **G3c**-proof *if it does not contain premises that are not as well conclusions of another rule, i.e. each edge to a leaf of the corresponding tree belongs to an instance of Ax or $l{:}\bot$.*

We call the sequent that corresponds to the root node the end sequent*.*

Definition 2 *Let S be a quantifier-free sequent. We call a **G3c**-derivation of S* maximal *if the leaves of the tree cannot be conclusions of rules.*

As already mentioned, the **G3c**-calculus is invertible. This is a useful property for proof search since we can apply the rules backwards without loosing completeness, i.e. if a sequent is provable and it is a conclusion of a rule all its premises are provable, too.

Proposition 1 (Inversion Lemma [TS96, Proposition 3.5.4.]) *Let $\Rightarrow$ denote deducibility in **G3c**.*

1. *If $\Rightarrow F \wedge G, \Gamma \vdash \Delta$ then $\Rightarrow F, G, \Gamma \vdash \Delta$.*
2. *If $\Rightarrow \Gamma \vdash \Delta, F \vee G$ then $\Rightarrow \Gamma \vdash \Delta, F, G$.*
3. *If $\Rightarrow F \vee G, \Gamma \vdash \Delta$ then $\Rightarrow F, \Gamma \vdash \Delta$ and $\Rightarrow G, \Gamma \vdash \Delta$.*
4. *If $\Rightarrow \Gamma \vdash \Delta, F \wedge G$ then $\Rightarrow \Gamma \vdash \Delta, F$ and $\Rightarrow \Gamma \vdash \Delta, G$.*
5. *If $\Rightarrow \Gamma \vdash \Delta, F \to G$ then $\Rightarrow F, \Gamma \vdash \Delta, G$.*
6. *If $\Rightarrow F \to G, \Gamma \vdash \Delta$ then $\Rightarrow \Gamma \vdash \Delta, F$ and $\Rightarrow G, \Gamma \vdash \Delta$.*
7. *If $\Rightarrow \Gamma \vdash \Delta, \forall x F$ then $\Rightarrow \Gamma \vdash \Delta, F[x\backslash\alpha]$, for any α such that $\alpha \notin \mathcal{F}(\Gamma, \Delta, F)$.*
8. *If $\Rightarrow \exists x F, \Gamma \vdash \Delta$ then $\Rightarrow F[x\backslash\alpha], \Gamma \vdash \Delta$, for any α such that $\alpha \notin \mathcal{F}(\Gamma, \Delta, F)$.*

We refer to quantifiers that are introduced by $r\colon\forall$ or $l\colon\exists$ as *strong quantifiers* while quantifiers that are introduced by $l\colon\forall$ or $r\colon\exists$ are referred to as *weak quantifiers*. We will make use of the symbol $\neg$ to denote $\to \bot$. Moreover, we will abbreviate

$$l\colon\to \frac{\Gamma \vdash \Delta, F \qquad Ax\,\dfrac{}{\bot, \Gamma \vdash \Delta}}{F \to \bot, \Gamma \vdash \Delta} \qquad \text{and} \qquad r\colon\to \frac{F, \Gamma \vdash \Delta, \bot}{\Gamma \vdash \Delta, F \to \bot}$$

by

$$l\colon\neg \frac{\Gamma \vdash \Delta, F}{\neg F, \Gamma \vdash \Delta} \qquad \text{and} \qquad r\colon\neg \frac{F, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg F}$$

respectively. If $\bot$ occurs on the right of a sequent, we are allowed to drop it. Whenever we refer to the **G3c**-calculus from now on, we mean the version in which $l\colon\neg$ and $r\colon\neg$ are included. Moreover, we call the rules with two premises, i.e. $r\colon\wedge$, $l\colon\vee$, and $l\colon\to$, *binary rules* and the rules with a single premise, i.e. $l\colon\wedge$, $r\colon\vee$, $r\colon\to$, $l\colon\forall$, $r\colon\forall$, $l\colon\exists$, $r\colon\exists$, $l\colon\neg$, and $r\colon\neg$, *unary rules*.

We denote the extension of **G3c**-calculus with the *Cut* rule defined as

$$Cut\ \frac{\Gamma \vdash \Delta, C \qquad C, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}$$

the **G3c**$^+$-calculus and in this context, we call C the cut formula. Other than all other rules, the *Cut* rule is not analytic as its premises contain a formula that might be unrelated to the conclusion. Moreover, C is not necessarily a subformula of any formula in Γ or Δ. Intuitively, the *Cut* rule can be seen as the application of a lemma. While

a proof of the left premise is a proof of the lemma in the current context, a proof of the right premise is allowed to make use of the proven lemma C. This might result in a simplification of the proof itself as known from mathematics where lemmas play a crucial role. Of course, mathematical proofs are usually far beyond classical first-order logic, but in Section 3.6 and in Section 4.9 we provide examples for classical first-order logic that demonstrate the enormous compression that can be achieved by lemmas with a certain structure.

For each $\mathbf{G3c}^{+}$-rule, we call the multisets Γ and Δ the *context* while the other formulas are called *main formulas*.

2.3 Normal Forms

It is well known that logical connectives can be simulated by others. In classical logic, the formula $F \wedge G$ is equivalent to $\neg(\neg F \vee \neg G)$. This allows us to define normal forms for formulas in which we ask for certain properties such as: only the connectives $\vee$ and $\neg$ shall occur. Of course, there are restrictions on how a normal form can be defined. It is impossible to find for each classical formula an equivalent variant only using the connective $\wedge$.

In the present work, we will make use of the disjunctive normal form. In order to give a formal definition, we look at the negation normal form in which the negation only occurs in front of atoms. Note that we cannot define a negation normal form in intuitionistic logic. If we want to transform an arbitrary formula into negation normal form we have to apply the double negation elimination, i.e. two negations $\neg$ that appear directly after each other can be dropped. This is only valid in classical logic.

Definition 3 (Negation Normal Form) *A formula F is in* negation normal form *(shorthand:* **NNF***) if only the connectives $\forall$, $\exists$, $\wedge$, $\vee$, and $\neg$ occur in F and all occurrences of $\neg$ are in front of atoms.*

Example 1 *The formulas $F =_{def} P \wedge (Q \vee \neg R)$ and $G =_{def} (P \wedge Q) \vee (P \wedge \neg R)$ are in* ***NNF*** *while the formula $P \wedge \neg\neg P$ is not in* ***NNF****. Note that the formulas F and G are equivalent. Hence,* ***NNF*** *is not a canonical normal form.*

Let F, G be formulas of first-order logic. Consider the rewrite rules

$$\begin{aligned} F \to G &\Rightarrow \neg F \vee G \\ \neg\forall x F &\Rightarrow \exists x \neg F \\ \neg\exists x F &\Rightarrow \forall x \neg F \\ \neg(F \vee G) &\Rightarrow \neg F \wedge \neg G \\ \neg(F \wedge G) &\Rightarrow \neg F \vee \neg G \\ \neg\neg F &\Rightarrow F \end{aligned}$$

where $\Rightarrow$ is the rewriting operation. We can apply a rewrite rule to a formula if it contains a subformula of one of the forms appearing on the left of $\Rightarrow$. During the application of a rewrite rule, we replace the subformula according to the rule by the formula on the right of $\Rightarrow$. Then every formula C can be transformed into an equivalent formula D in **NNF** by an exhaustive application of the defined rewrite rules. The last rewrite rule is the mentioned double negation elimination.

A formula in **NNF** can easily be transformed into a formula in disjunctive normal form (or conjunctive normal form). The drawback is an exponential blow up of the formula size. This can be omitted if we transform into satisfiability-equivalent formulas instead of equivalent formulas (cf. Tseitin transformation [Tse68]). The standard transformation uses the distributivity of $\vee$ and $\wedge$ as well as rules to shift the quantifiers in front:

$$\begin{aligned} F \wedge (G \vee H) &\Rightarrow (F \wedge G) \vee (F \wedge H)\,, \\ (\forall x F) \wedge G &\Rightarrow \forall x\,(F \wedge G)\,, \\ (\exists x F) \wedge G &\Rightarrow \exists x\,(F \wedge G)\,, \\ (\forall x F) \vee G &\Rightarrow \forall x\,(F \vee G)\,, \text{ and} \\ (\exists x F) \vee G &\Rightarrow \exists x\,(F \vee G) \end{aligned}$$

for formulas F, G, and H. When shifting quantifiers in front, we have to take care that the quantified variable x does not occur in G. This can be achieved by renaming.

Definition 4 (Disjunctive Normal Form) *A formula F is in* disjunctive normal form *(shorthand:* **DNF***) if all quantifiers are in front (prenexed)*

$$F = \forall \vec{x}_1 \exists \vec{y}_1 \ldots \forall \vec{x}_n \exists \vec{y}_n\,(F')$$

with possibly empty tuples of variables $\vec{x}_j, \vec{y}_k$ and the quantifier free body is of the form

$$F' = G_1 \vee \ldots \vee G_l$$

where for all $i \in \mathbb{N}_l$

$$G_i \stackrel{def}{=} L^i_1 \wedge \ldots \wedge L^i_{m^i}$$

with L^i_j being an atom or a negated atom. Furthermore, we call L^i_j a literal *and G_k a* clause *of F for arbitrary i's, j's, and k's.*

In literature, clauses correspond usually to formulas in conjunctive normal form. The normal form is analogously defined as the **DNF**, but $\wedge$ is exchanged with $\vee$ and vice versa. Since we only consider formulas in **DNF**, we are allowed to use the term clause to denote subformulas of a formula in **DNF** as in Definition 4. As well as the **NNF**, the **DNF** is not a canonical normal form, since different formulas in **DNF** can be logically equivalent.

2.4 Complexity Measurements

On several occasions, we will measure the complexity of different objects with respect to changing parameters. To preserve a better overview, this section contains all complexity measurements occurring in this work. Therefore, we will also mention definitions of complexity which should be read first in the context of their chapter, but can be looked up here. In these cases, we refer to the corresponding chapter.

2.4.1 Size of Sets

The size of a set is defined in the standard way, as the number of elements occurring in the set. The concept can be defined without specifying the type of the elements within the set.

Definition 5 (Size of a Set) *Let S be a set of arbitrary elements then the size of a set $|S|$ is given by the number of elements in S.*

2.4.2 Complexity of Sets of Tuples of Terms

In comparison to the size of a set, the definition of the *complexity* of a set deviates. Indeed, we will only define the complexity of sets of tuples of terms. This definition will play a role when considering a minimal **G3c**-derivation that instantiates a block of quantifiers in order to get ground formulas which are instantiated with the tuples of the set. Hence, the complexity of such a set corresponds to the minimal number of quantifier inferences necessary to ground the formula.

Definition 6 (Instantiation Complexity) *Let $T =_{def} \{\vec{r}_1, \ldots, \vec{r}_l\}$ be a set of m-tuples of ground terms. Assume an m-ary predicate symbol P. Then we define the* instantiation complexity of T*, denoted by $\sharp T$, as the minimal number of quantifier inferences in a cut-free proof of the sequent*

$$\forall x_1 \ldots x_m P(x_1, \ldots, x_m) \vdash P(\vec{r}_1) \wedge \ldots \wedge P(\vec{r}_l).$$

Example 2 *Let $\vec{r} = (a, b, c)$ and $\vec{s} = (a, b, b)$ be tuples of terms r, s, t and $T = \{\vec{r}, \vec{s}\}$. Figure 2.1 shows a proof of $\forall x_1, x_2, x_3 P(x_1, x_2, x_3) \vdash P(\vec{r}) \wedge P(\vec{s})$ with a minimal number of quantifier instantiations. Thus, $\sharp T = 4$.*

2.4.3 Complexity of Sequents

The definitions of this subsection should be read first in their corresponding sections. They are concerned with the complexity of sequents depending on whether the sequents represent the main content of a cut-free proof, a proof with Π_1 cuts, or a proof with a Π_2 cut.

The following definition appears first in Section 2.5.

Figure 2.1: Cut-free proof of $\forall x_1, x_2, x_3 P(x_1, x_2, x_3) \vdash P(\vec{r}) \wedge P(\vec{s})$; Example 2

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Ax\ \dfrac{}{P(\vec{r}), P(\vec{s}), \Gamma_3 \vdash P(\vec{r})} \quad Ax\ \dfrac{}{P(\vec{r}), P(\vec{s}), \Gamma_3 \vdash P(\vec{s})}}{r:\wedge\ P(\vec{r}), P(\vec{s}), \Gamma_3 \vdash P(\vec{r}) \wedge P(\vec{s})}}{l:\forall\ P(\vec{r}), \forall x_3 P(a, b, x_3), \Gamma_2 \vdash P(\vec{r}) \wedge P(\vec{s})}}{l:\forall\ \forall x_3 P(a, b, x_3), \Gamma_2 \vdash P(\vec{r}) \wedge P(\vec{s})}}{l:\forall\ \forall x_2, x_3 P(a, x_2, x_3), \Gamma_1 \vdash P(\vec{r}) \wedge P(\vec{s})}}{l:\forall\ \forall x_1, x_2, x_3 P(x_1, x_2, x_3) \vdash P(\vec{r}) \wedge P(\vec{s})}$$

with

$$\Gamma_1 \stackrel{def}{=} \{\forall x_1, x_2, x_3 P(x_1, x_2, x_3)\}$$
$$\Gamma_2 \stackrel{def}{=} \Gamma_1 \cup \{\forall x_2, x_3 P(a, x_2, x_3)\}$$
$$\Gamma_3 \stackrel{def}{=} \Gamma_2 \cup \{\forall x_3 P(a, b, x_3)\}$$

Definition 7 (Complexity of Herbrand sequents. See Definition 14.) *Let*

$$H \stackrel{def}{=} F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_l] \vdash G[\vec{y}\backslash\vec{r}_{l+1}], \ldots, G[\vec{y}\backslash\vec{r}_m].$$

be a Herbrand sequent. Then the complexity of H is defined as

$$|H| \stackrel{def}{=} \sharp\{\vec{r}_1, \ldots, \vec{r}_l\} + \sharp\{\vec{r}_{l+1}, \ldots, \vec{r}_m\}$$

where $\sharp\{\cdot\}$ denotes the instantiation complexity (Definition 6).

The following definition appears first in Section 3.1.

Definition 8 (Complexity of Π_1-EHSs. See Definition 21.) *Let*

$$\mathcal{H} \stackrel{def}{=} F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_l], G_1 \to \bigwedge_{j=1}^{n_1} G_1[\alpha_1\backslash s_{1,j}], \ldots, G_m \to \bigwedge_{j=1}^{n_m} G_m[\alpha_m\backslash s_{m,j}] \vdash$$

*be a Π_1-**EHS**. Then the complexity of $\mathcal{H}$ is defined as $|\mathcal{H}|_{\Pi_1} =_{def} \sharp\{\vec{r}_1, \ldots, \vec{r}_l\} + \sum_{j=1}^{m} n_j$.*

The following definition appears first in Section 4.2.

Definition 9 (Complexity of Π_2-EHSs. See Definition 28.) *Let*

$$\mathcal{H} \stackrel{def}{=} F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}, \bigvee_{i=1}^{q} C[(x, y) \setminus (\alpha, r_i)] \to \bigwedge_{j=1}^{p} C[(x, y) \setminus (s_j, \beta_j)] \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}$$

*be a Π_2-**EHS**. Then the complexity of $\mathcal{H}$ is defined as*

$$|\mathcal{H}|_{\Pi_2} \stackrel{def}{=} \sharp\{\vec{r}_1, \ldots, \vec{r}_l\} + \sharp\{\vec{s}_1, \ldots, \vec{s}_m\} + p + q.$$

2.4.4 Complexity of Proofs in G3c

In the course of this thesis, we will show that proofs containing cuts can be much smaller than proofs without cuts (see Section 3.6 and Section 4.9). Apart from the question when and how this is possible, we need terminology to speak about the size/complexity of a proof. Depending on how concisely we want to compare the sizes, different measurements are used. The *quantifier complexity* counts the number of weak quantifier instantiations in a proof. Thereby, we ignore the propositional part of a proof and hence, logically complex formulas are not treated differently than simple predicates as long as they have the same quantification. Especially, if the proof size mainly depends on the occurring terms, the quantifier complexity gives a easily computable and sufficient approximation of the size of a proof. It plays a major role in cut introduction, since the propositional structure of the context in such problems does not change, but its instantiations. Only the cut formula changes the logical structure considerably.

Definition 10 (Quantifier Complexity) *Let φ be a **G3c**-proof. We define the* quantifier complexity $|\varphi|_q$ *as the number of weak quantifier inferences in φ.*

As mentioned before, the quantifier complexity ignores the propositional part of a proof. For this reason, we introduce the *inference complexity* which counts the number of all occurring **G3c**-rules that are not **G3c**-axioms. This allows us to check whether an introduced cut formula increases the proof size due to its logical structure. Even though the computation of the inference complexity is not as simple as the computation of the quantifier complexity, it is sufficiently easy to compute.

Definition 11 (Inference Complexity) *Let φ be a **G3c**-proof. If φ is of the form*

$$Ax\,\frac{}{\Gamma \vdash \Delta} \qquad \text{or} \qquad l\colon\bot\,\frac{}{\Gamma \vdash \Delta}\ ,$$

then the inference complexity $|\varphi|_i$ *is defined to be* 0*. If φ is of the form*

$$\textit{Binary rule}\,\frac{\chi \qquad \psi}{\Gamma \vdash \Delta}$$

*with an arbitrary binary rule with **G3c**-subproofs χ and ψ, then $|\varphi|_i =_{def} |\chi|_i + |\psi|_i + 1$. If φ is of the form*

$$\textit{Unary rule}\,\frac{\chi}{\Gamma \vdash \Delta}$$

*with an arbitrary unary rule with a **G3c**-subproof χ, then $|\varphi|_i =_{def} |\chi|_i + 1$.*

Even though the Inference complexity considers the propositional part of a proof, it might miss some part of the proof that is relevant for its size: the term structure. While the representation of a single term on an actual computer is dependent on the underlying framework (and will not be discussed here), the nesting of terms and the number of occurrences are entities relevant to proof size (independent of their symbolic representation). In order to incorporate this as well, we define the *symbol complexity* which counts every symbol occurrence of every sequent and the number of rules. Usually, we do not compute the concise symbol complexity, but give upper bounds.

Definition 12 (Symbol Complexity) *Let φ be a **G3c**-proof and Σ the corresponding signature. Let $S_1, \ldots, S_m$ be the sequents occurring in φ. The* symbol complexity *$|S_i|_s$ of a sequent S_i for $i \in \mathbb{N}_m$ is equal to the number of occurrences of the symbols of the set $\Sigma \cup \{\wedge, \vee, \rightarrow, \neg, \exists, \forall, \vdash, \bot\}$ and of the variables occurring in S_i. The symbol complexity $|\varphi|_s$ of the proof is defined as*

$$|\varphi|_s \stackrel{def}{=} |\varphi|_i + \sum_{i \in \mathbb{N}_m} S_i.$$

It is easy to see that the different measurements for **G3c**-proofs follow an order. While the quantifier complexity is the most coarse one, the symbol complexity is the finest.

Proposition 2 (See Proposition 12 of [LL18]) *Let φ be a **G3c**-proof. Then the following inequalities hold:*

$$|\varphi|_q \leq |\varphi|_i \leq |\varphi|_s.$$

Proof:
The claim trivially holds. □

Example 3 *Let a, b, and c be constants and let P be a unary predicate, i.e. the signature Σ is $\{a, b, c, P\}$. Consider the proof φ of Figure 2.2. Then the complexity measures are as follows:*

$$\begin{aligned} |\varphi|_q &= 1, \\ |\varphi|_i &= 3, \text{ and} \\ |\varphi|_s &= 87. \end{aligned}$$

2.4.5 Complexity of Grammars

The following definition appears first in Section 3.2 and Example 9 shows the computation of the complexity of a schematic Π_1 grammar.

Figure 2.2: Simple proof used for the computation of complexity measures; Example 3

$$\varphi \overset{def}{=} \; l:\to \dfrac{Ax\dfrac{}{\Gamma, P(b), P(c) \vdash P(b)} \qquad r:\vee \dfrac{Ax\dfrac{}{\Gamma, P(c) \vdash P(c), P(a), P(b)}}{\Gamma, P(c) \vdash P(c) \vee P(a), P(b)}}{l:\forall \dfrac{\Gamma, P(c) \vee P(a) \to P(b), P(c) \vdash P(b)}{\forall x (P(x) \vee P(a) \to P(b)), P(c) \vdash P(b)}}$$

with $\Gamma =_{def} \forall x (P(x) \vee P(a) \to P(b))$

Definition 13 (Complexity of SΠ_1-Gs. See Definition 25.) *Let $\mathcal{G}$ be an $\boldsymbol{S\Pi_1}$-$\boldsymbol{G}$ of the form $\langle \tau, N, \Sigma, \Pr \rangle$ and T be the set of all tuples of terms $\vec{r}$ such that $\tau \to h_F \vec{r}$ is a production rule in* Pr *and h_F is the freshly introduced function. Then the complexity $|\mathcal{G}|$ of $\mathcal{G}$ is defined as*

$$\sharp T - l + m$$

where l is the number of tuples in T, $\sharp T$ is the instantiation complexity of T (see Definition 6), and m is the number of production rules in $\mathcal{G}$.

2.5 Herbrand's Theorem

One of the most prominent theorems in mathematical logic is Herbrand's theorem (see [Her30]). It tells us that for every provable sequent S in classical first-order logic exists a tautological sequent T consisting only of grounded formulas (the formulas do not contain variables) occurring in S. Therefore, it can be seen as a projection of first-order logic into propositional logic, of course only for the provable fragment.

Despite the strength of Herbrand's original theorem, the literature typically states just a weak variant. A discussion about this issue can be found in the paper [Bus95] which formulates the statement in its strongest form and elaborates the connection between the theorem and cut-elimination. Note that also in this work we will only use an easy consequence: For every provable sequent without strong quantifiers, there exists a Herbrand sequent. The construction of a sequent without strong quantifiers is provability preserving and was already implicitly contained in Herbrand's theorem, although the concept of sequents was introduced by Gentzen [Gen35a] years after Herbrand's thesis.

Definition 14 (Herbrand Sequent) *A tautological sequent H of the form*

$$F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_l] \vdash G[\vec{y}\backslash\vec{r}_{l+1}], \ldots, G[\vec{y}\backslash\vec{r}_m]$$

where $F[\vec{x}\backslash\vec{r}_i]$ for $i \in \mathbb{N}_l$ are instances of F and $G[\vec{y}\backslash\vec{r}_j]$ for $j \in \{l+1, \ldots, m\}$ are instances of G is called a Herbrand sequent *of $\forall \vec{x} F \vdash \exists \vec{y} G$. The complexity of H is defined as*

$$|H| \overset{def}{=} \sharp\{\vec{r}_1, \ldots, \vec{r}_l\} + \sharp\{\vec{r}_{l+1}, \ldots, \vec{r}_m\}$$

where $\sharp\{\cdot\}$ denotes the instantiation complexity (Definition 6). For abbreviation, we will write

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{l} \vdash [G[\vec{y}\backslash\vec{r}_i]]_{i=l+1}^{m}$$

which reduces in Chapter 3 to

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{l} \vdash .$$

Considering the complexity of Herbrand sequents, we took into account that Herbrand sequents are used as representation of cut-free proofs (see Corollary 1 and Theorem 3). Hence, the complexity is the minimal number (instantiation complexity) of weak quantifier inferences that is needed to introduce the tuples of terms. So we will always find a cut-free proof with an equal quantifier complexity.

Corollary 1 (Corollary of Herbrand's theorem) *Let the sequent $S =_{def} \forall\vec{x}F \vdash \exists\vec{y}G$ be provable. Then there exists a Herbrand sequent for S.*

Moreover, we can find a Herbrand sequent preserving the quantifier complexity of the proof. The following version is taken from [LL18] which is based on a version proven in [HLRW14].

Theorem 3 *Assume a sequent $S =_{def} \forall\vec{x}F \vdash \exists\vec{y}G$. There is a Herbrand sequent H of S with $|H| = l$ iff there exists a cut-free proof φ of S such that $|\varphi|_q = l$.*

Proof Sketch:
A Herbrand sequent describes exactly the terms we have to introduce by weak quantifier inferences. Let H be a Herbrand sequent of S with $|H| = l$. Then a cut-free proof φ with $|\varphi|_q = l$ can be constructed in the following way: apply all propositional inferences first and afterwards all quantifier rules.

Let φ be a cut-free proof of S. Then different terms for a given position of an atom can only be produced by weak quantifier inferences. Hence, the number of weak quantifier inferences in φ is equal to the number of different terms obtained by substitution, and therefore $|\varphi|_q = |H|$ for H being the Herbrand sequent obtained from φ. □

Every cut-free proof φ of a prenex end-sequent S can be transformed into a cut-free proof χ of S (without increase of proof length) s.t. χ contains a midsequent S^*, i.e. a sequent in χ such that all quantifier inferences in χ are below S^* and all propositional ones above [Gen35b].

Example 4 *Let $S =_{def} P(a) \vee P(b) \vee P(c) \vdash \exists x P(x)$ be a sequent where P is a unary predicate symbol and a, b, c are terms. The proof φ of Figure 2.3 is cut free. The*

Figure 2.3: Cut-free proof φ; Example 4

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Ax\ \dfrac{}{P(a) \vdash P(a), P(b), P(c), \exists x P(x)}}{l:\vee\ \dfrac{P(a) \vee P(b) \vee P(c) \vdash P(a), P(b), P(c), \exists x P(x)}{}}\ \chi}{l:\exists\ \dfrac{P(a) \vee P(b) \vee P(c) \vdash P(a), P(b), \exists x P(x)}{}}}{l:\exists\ \dfrac{P(a) \vee P(b) \vee P(c) \vdash P(a), \exists x P(x)}{}}}{l:\exists\ \dfrac{}{P(a) \vee P(b) \vee P(c) \vdash \exists x P(x)}}$$

with $\chi =_{def}$

$$\dfrac{Ax\ \dfrac{}{P(b) \vdash P(a), P(b), P(c), \exists x P(x)} \qquad Ax\ \dfrac{}{P(c) \vdash P(a), P(b), P(c), \exists x P(x)}}{l:\vee\ \dfrac{}{P(b) \vee P(c) \vdash P(a), P(b), P(c), \exists x P(x)}}$$

midsequent of φ is

$$P(a) \vee P(b) \vee P(c) \vdash P(a), P(b), P(c), \exists x P(x)$$

and therefore,

$$P(a) \vee P(b) \vee P(c) \vdash P(a), P(b), P(c)$$

is a Herbrand sequent of S. The complexity of the Herbrand sequent is

$$\sharp\{a, b, c\} = 3$$

which is equal to the quantifier complexity $|\varphi|_q$ of φ.

As shown in the previous example, the Herbrand sequent serves as a representation of the term instantiations of a cut-free proof. In the course of this thesis, we will elaborate a connection between formal language theory and proof theory. Therefore, it will be necessary to translate Herbrand sequents into formal language theory.

Definition 15 (Herbrand Term Set) *Let H be a Herbrand sequent of the form*

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{l} \vdash [G[\vec{y}\backslash\vec{r}_i]]_{i=l+1}^{m}$$

of $\forall\vec{x}F \vdash \exists\vec{y}G$ (see Definition 14). Let h_F and h_G be fresh functions such that $\boldsymbol{a}(h_F) = \boldsymbol{a}(F)$ and $\boldsymbol{a}(h_G) = \boldsymbol{a}(G)$. Then the set

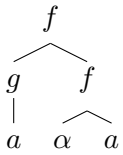
$$H_s \overset{def}{=} \{h_F\vec{r}_i \mid i \in \mathbb{N}_l\} \cup \{h_G\vec{r}_i \mid i \in \mathbb{N}_m \backslash \mathbb{N}_l\}$$

is called a Herbrand term set *of H.*

Since we consider in Chapter 3 only Herbrand sequents of the form

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{l} \vdash,$$

Figure 2.4: A tree representation of a term



the Herbrand term set reduces in this chapter to

$$H_s \overset{def}{=} \{h_F \vec{r}_i \mid i \in \mathbb{N}_l\}.$$

Example 5 *Let us consider the Herbrand sequent of Example 4, i.e.*

$$P(a) \vee P(b) \vee P(c) \vdash P(a), P(b), P(c).$$

Then the Herbrand term set is defined as

$$\{h_{P(a)\vee P(b)\vee P(c)}, h_P a, h_P b, h_P c\}$$

where $\boldsymbol{a}\left(h_{P(a)\vee P(b)\vee P(c)}\right) = 0$ *and* $\boldsymbol{a}(h_P) = 1$.

2.6 Grammars

The results considered in this work are based on relations between formal language theory and proof theory [AHL15]. The way variables are replaced in the procedure of cut-elimination can be defined by tree grammars modelling substitutions of terms. [Het12] presents a characterization of the substitutions defining the Herbrand instances of a proof after the elimination of a Π_1 cut. In this section, we give some necessary definitions for tree grammars in general , whereas the grammars for introducing cuts are defined later (cf. Definitions 23, 24, 29, 30, and 45). A good overview of tree grammars can be found in [CDG$^+$08] and [GS97].

While usual grammars consider terms as strings, tree grammars are more closely related to proof theory as they consider terms as labelled trees. A labelled tree, as we consider it in tree grammars, is a tree where every node is labelled with a function of the considered signature or with a variable; the number of ancestors of a node corresponds to the arity of the label of the node. A simple example of a tree representing a term is shown in Figure 2.4. Here, f is a binary function, g is a unary function, a a nullary function, and α a variable. The term represented by the tree is $f(g(a), f(\alpha, a))$.

In general, we distinguish between terminals and nonterminals. The terminals denote those symbols that cannot be removed or changed in an already constructed tree (except in an unrestricted tree grammar). The nonterminals might change during the construction.

Therefore, nonterminals behave like first-order variables, even though, first-order variables do not have to be represented as nonterminals. The distinction between terminals and nonterminals is necessary to decide whether a term is in the language of a tree grammar. The language only contains terms without nonterminals (similar to grounded terms in first-order logic).

A tree grammar consists basically of a signature, rewriting rules, and a starting symbol. Additionally, there are nonterminals that are separated from the terms of the signature and denote the terms to which the rewriting rules can be applied (the unrestricted tree grammars are special, since they allow the rewriting rules to be applied to all terms). The starting symbol is considered to be a nonterminal and provides, as the name suggests, a starting point from which terms can be derived. A single step derivation exchanges a subterm according to a rewriting rule. Consider again the term $f\left(g\left(a\right), f\left(\alpha, a\right)\right)$ and assume the rewriting rule $\alpha \to g\left(\alpha\right)$. Then we can derive the term $f\left(g\left(a\right), f\left(g\left(\alpha\right), a\right)\right)$ in a single step and the terms $\{f\left(g\left(a\right), f\left(r, a\right)\right) \mid r = g^l\left(\alpha\right) \wedge l \in \mathbb{N}\}$ in an arbitrary number of derivations. A term r belongs to the language of a tree grammar if there is a derivation of r according to the rewriting rules of the grammar beginning with the starting symbol such that r does not contain nonterminals. Hence, the terms of $\{f\left(g\left(a\right), f\left(r, a\right)\right) \mid r = g^l\left(\alpha\right) \wedge l \in \mathbb{N}\}$ cannot be part of a language. If we also assume the rewriting rule $\alpha \to b$ and the start production $\tau \to f\left(g\left(a\right), f\left(\alpha, a\right)\right)$, the language contains the terms $\{f\left(g\left(a\right), f\left(r, a\right)\right) \mid r = g^l\left(b\right) \wedge l \in \mathbb{N}\}$.

The most general formal grammars, according to the Chomsky-hierarchy [Cho56], are the unrestricted grammars (also called: type-0-grammars). For defining the $\mathcal{G}^*$-unifiability (cf. Section 4.7), we need a tree grammar without restrictions on the productions, i.e. tree grammars that correspond to unrestricted grammars. For this reason, we introduce the concept of unrestricted tree grammars.

Definition 16 (Unrestricted Tree Grammar) *An* unrestricted tree grammar $\mathcal{G}$ *is a tuple* $\langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ *where* N *is a finite set of nonterminal symbols with arity* 0 *such that* $\tau \in N$*. Furthermore,* Σ *is a finite set of function symbols of arbitrary arities, i.e. a term signature, satisfying* $N \cap \Sigma = \emptyset$*. The productions (or production rules)* Pr *consist of a finite set of rules of the form* $r \to s$ *where* $r, s \in \mathcal{T}\left(\Sigma \cup N\right)$*, where* $\mathcal{T}\left(\Sigma \cup N\right)$ *denotes the set of all terms definable from symbols in* $\Sigma \cup N$*. As usual* $\mathcal{L}\left(\mathcal{G}\right)$*, the* language *defined by* $\mathcal{G}$ *is the set of all terminal strings (ground terms) derivable in* $\mathcal{G}$*.*

As mentioned before, the term *unrestricted* in the definition of an unrestricted tree grammar refers to the form of the rewriting rules. These are able to exchange arbitrary subterms of $\mathcal{T}\left(\Sigma \cup N\right)$, for example terms such as $f\left(\alpha, a\right)$ where f is a binary function of Σ, a is a zeroary function of Σ, and α is a nonterminal. Therefore, unrestricted tree grammars can be used to derive abstractions of terms, while most other grammars derive only instances.

A special case of unrestricted tree grammars are the regular tree grammars, that restrict the form of the productions. They are similar to context-free grammars (also called: type-2-grammars) of the Chomsky-hierarchy [Cho56], but not equivalent.

Definition 17 (Regular Tree Grammar) *A* regular tree grammar $\mathcal{G}$ *is an unrestricted tree grammar where the productions* Pr *are of the form* $\alpha \to r$ *with* $\alpha \in N$ *and* $r \in \mathcal{T}(\Sigma \cup N)$.

In general, tree grammars might contain cyclic productions that derive after a number of applications the nonterminal they started with. A simple example is a production rule that replaces a nonterminal with a term containing the same nonterminal, for instance $\alpha \to g(\alpha)$. By disallowing this behaviour (cf. Definition 18), the language of a regular tree grammar becomes finite, since after a finite number of derivations all nonterminals are replaced and we are not able to apply any other production rule.

The languages of grammars specifying Herbrand instances are finite (see [HLRW14]) and therefore their productions may be assumed to be acyclic. Indeed, if the language of a grammar $\mathcal{G}$ is finite and $\mathcal{G}$ is cyclic, there exists a grammar $\mathcal{G}'$ such that $\mathcal{G}'$ is acyclic and its language is the same as the language of $\mathcal{G}$, i.e. $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$.

Definition 18 (Acyclic Tree Grammar) *We call a regular tree grammar* acyclic *if there is a strict total order* $<$ *on the nonterminals* N *such that for each rule* $\alpha \to r$ *in* Pr *only nonterminals smaller than* α *occur in* r.

Moreover, we are interested in grammars specifying substitutions. As substitutions are homomorphic mappings on terms, variables have only to be replaced by single terms within a derivation. Therefore, we need a restriction of derivations, rigid derivations (see [JKV09]).

Definition 19 (Rigid Derivation) *We call a derivation* rigid *with respect to a nonterminal* α *if only a single rule for* α *is allowed to occur in the derivation.*

The following example is a variant of Example 2 of [HLRW14]. It shows that allowing only rigid derivations with respect to a nonterminal makes regular tree grammars more concise. There are languages $\mathcal{L}$ which cannot be generated by a regular tree grammar, but such a grammar generating $\mathcal{L}$ exists when only rigid derivations are admitted.

Example 6 *Let* $\Sigma =_{def} \{0, f\}$ *where* 0 *is a constant and* f *is a unary function. On the one hand, a simple pumping argument shows that the language* $\mathcal{L} =_{def} \{g(r, r) \mid r \in \mathcal{T}(\Sigma)\}$ *where* $\mathcal{T}(\Sigma)$ *are all terms constructable with the terms of* Σ *is not regular. On the other hand,* $\mathcal{L}$ *is generated by the tree grammar* $\mathcal{G} =_{def} \langle \tau, \{\tau, \alpha\}, \{0, f, g\}, \mathrm{Pr} \rangle$ *where* $\mathrm{Pr} =_{def} \{\tau \to g(\alpha, \alpha), \alpha \to 0 \mid f(\alpha)\}$ *if all derivations in* $\mathcal{G}$ *are rigid with respect to* α.

Definition 20 (Totally Rigid Acyclic Tree Grammar) *We call an acyclic regular tree grammar* $\langle \tau, N, \Sigma, \Pr \rangle$ *a* totally rigid acyclic tree grammar *if all derivations with respect to all* $\alpha \in N$ *are rigid.*

A totally rigid acyclic tree grammar is a special case of the rigid (acyclic) tree grammars (see [JKV09]) in which only the derivations with respect to all $\alpha \in N'$ for $N' \subseteq N$ have to be rigid. In the course of this thesis, we will only make use of totally rigid acyclic tree grammars.

CHAPTER 3

Revisiting Π_1-Cut Introduction

In this section, we present the basic methods for introducing Π_1 cuts in classical first-order proofs. Thereby, we mainly follow the paper [HLRW14]. The interested reader might also read [HLW12, HLR$^+$14, EH15] where the same methodology is used to introduce Π_1 cuts in systems with equality or to introduce Π_1-induction invariants.

Work on cut-introduction can be found at various places in the literature. Closest to our work are those approaches which abbreviate or structure given input proofs: in [WP10] an algorithm for the introduction of atomic cuts is developed that is capable of exponential proof compression. There exist several contributions to proof compression by cut-introduction in propositional logic: a method defined in [FG07] is shown to never increase the size of proofs more than polynomially, [DFG08] describes compression by cut-introduction in the more general context of cut-based abduction; the paper [DFG13] presents a general framework for theorem proving with analytic and bounded cut. Another approach to the compression of first-order proofs is based on introduction of definitions for abbreviating terms and can be found in [VSU10].

In general, cut introduction can also be called lemma generation which itself is performed in many different ways. In [Bun01, BBHI05, IB96], methods are presented to find proofs with induction where the construction of lemmas is necessary. In automated theory formation [Col01, Col02], an eager approach to lemma generation is adopted. One common approach most of them share is that they try to define well performing heuristics, either to find short proofs with cuts, to find theorems in theories, etc. Π_1-cut introduction, as we present it in this chapter, differs fundamentally from these methodologies. Instead, it directly inverts Gentzen's cut-elimination procedure. Therefore, we represent the Herbrand sequent by a term language (Section 3.1) which itself will be covered by a schematic Π_1 grammar (Section 3.2). Then we compute a cut formula and a proof with cut (Section 3.4). Section 3.5 presents how the method performs in practice and further research topics.

For simplicity, this chapter will only consider proofs that have an end sequent of the form $\forall \vec{x} F \vdash$. This is also no theoretical restriction, since every sequent is equivalent to a sequent with formulas only occurring on the left. By applying $l\colon\wedge$ and prenexing, we receive a sequent of the described form. After this chapter, we will again consider formulas of the form $\forall \vec{x} F \vdash \exists \vec{y} G$. Moreover, we consider only cut formulas with a single quantified variable instead of blocks of quantified variables. This is indeed a restriction, but can easily be fixed (see Section 3.5).

3.1 Analysis of Π_1 Cuts in Sequent Calculus

In Section 2.5, we determined that the main information of a cut-free proof, i.e. all necessary term instantiations, can be stored in a Herbrand sequent. When cuts appear in the proof, the existence of a Herbrand sequent is guaranteed by Gentzen's Hauptsatz (cut-elimination theorem, [Gen35b]). But after applying cut-elimination, the information of the cut is lost, i.e. the Herbrand sequent does not suffice to represent proofs with cut. Therefore, the notion of an *extended Herbrand sequent* was developed. In order to explain this notion, we will regard a proof with a single cut which appears bottommost in the corresponding proof tree. In such a proof, we can consider the two subproofs of the premises of the cut-rule and derive their midsequents. These can be transformed into Herbrand sequents that contain the whole information of the proof with cut, but in two sequents. By applying $l\colon\rightarrow$, we receive a single sequent still containing the whole information (see Example 7).

In general, we can extract an extended Herbrand sequent for Π_1 cuts from a proof with Π_1 cuts and an extended Herbrand sequent for Π_1 cuts allows us to construct a proof with Π_1 cuts. But note that both proofs, the one we constructed using the extended Herbrand sequent for Π_1 cuts and the one that was used for extracting an extended Herbrand sequent for Π_1 cuts, are not necessarily the same. Only the quantifier inferences are shared.

Example 7 *Let P be a unary predicate symbol, a a constant, and f a unary function. Let $f^l a$ be an abbreviation for $f \dots f a$ where f appears l times,*

$$\forall x F \stackrel{def}{=} \forall x \left(Pa \wedge (Px \rightarrow Pfx) \wedge \neg Pf^4 a \right),$$

and $\forall y G =_{def} \forall y \left(\neg Py \vee Pf^2 y \right)$. The formula F is unsatisfiable, since the first two conjuncts Pa and $Px \rightarrow Pfx$ tell us that $Pf^l a$ is true for all natural numbers l. This contradicts the last conjunct $\neg Pf^4 a$. Hence we can prove the sequent $\forall x F \vdash$. In such a proof, we would have to show that Pz is true for z being $fa, f^2 a, f^3 a$ up to $f^4 a$, i.e. for every instance between fa and $f^4 a$ starting with the given instance for z being a. If we look closer on $\forall x F$, we see that in its presence $\forall y G$ is provable. This gives us the possibility to take two steps at once, i.e. we can prove directly $Pf^2 a$ starting from Pa and in a second step $Pf^4 a$ starting from $Pf^2 a$. Then

$$\dfrac{\dfrac{\dfrac{\vdots}{\forall xF, F[x\backslash\alpha], F[x\backslash f\alpha] \vdash G[y\backslash\alpha]}\, l:\forall}{\dfrac{\vdots}{\forall xF \vdash \forall yG}}\quad \dfrac{\dfrac{\dfrac{\vdots}{\forall xF, F, \forall yG, G[y\backslash a], G[y\backslash f^2 a] \vdash}\, l:\forall}{\dfrac{\vdots}{\forall xF, \forall yG \vdash}}}{\forall xF \vdash}\, Cut$$

is a proof of $\forall xF \vdash$ *with a single cut. The sequents*

$$\forall xF, F[x\backslash\alpha], F[x\backslash f\alpha] \vdash G[y\backslash\alpha]$$

and

$$\forall xF, F, \forall yG, G[y\backslash a], G[y\backslash f^2 a] \vdash$$

are midsequents of the subproofs of the end sequents $\forall xF \vdash \forall yG$ *and* $\forall xF, \forall yG \vdash$*, respectively. Note that we need the subformulas* Pa *and* $\neg P f^4 a$ *of* F *to get a midsequent in the right branch of the proof. Therefore, we applied once the* $l:\forall$ *rule with the substitution* $[x\backslash x]$ *to* $\forall xF$*, apart from the applications to* $\forall yG$*. Now, we can drop the quantified formulas to get the corresponding Herbrand sequents, i.e.*

$$\begin{aligned} F[x\backslash\alpha], F[x\backslash f\alpha] &\vdash G[y\backslash\alpha] \text{ and} \\ F, G[y\backslash a], G[y\backslash f^2 a] &\vdash . \end{aligned}$$

These represent the content of cut-free proofs of the two sequents $\forall xF \vdash \forall yG$ *and* $\forall xF, \forall yG \vdash$*. For technical reasons, we extend the sequents such that they have a shared context for an application of a* $l:\to$*-inference:*

$$\begin{aligned} F[x\backslash\alpha], F[x\backslash f\alpha], F &\vdash G[y\backslash\alpha] \text{ and} \\ F[x\backslash\alpha], F[x\backslash f\alpha], F, G[y\backslash a], G[y\backslash f^2 a] &\vdash . \end{aligned}$$

These are still Herbrand sequents, albeit they are not minimal. In order to summarize the information in one sequent, we apply an $l:\wedge$*-inference and an* $l:\to$*-inference*

$$l:\to \dfrac{F[x\backslash\alpha], F[x\backslash f\alpha], F \vdash G[y\backslash\alpha] \qquad \varphi}{F[x\backslash\alpha], F[x\backslash f\alpha], F, G[y\backslash\alpha] \to (G[y\backslash a] \wedge G[y\backslash f^2 a]) \vdash}$$

with

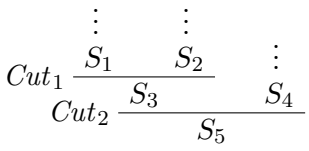
$$\varphi \overset{def}{=} \qquad l:\wedge \dfrac{F[x\backslash\alpha], F[x\backslash f\alpha], F, G[y\backslash a], G[y\backslash f^2 a] \vdash}{F[x\backslash\alpha], F[x\backslash f\alpha], F, G[y\backslash a] \wedge G[y\backslash f^2 a] \vdash}$$

Finally, we get the extended Herbrand sequent for Π_1 cuts

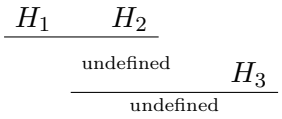
$$F[x\backslash\alpha] \wedge F[x\backslash f\alpha] \wedge F, G[y\backslash\alpha] \to \Big(G[y\backslash a] \wedge G[y\backslash f^2 a]\Big) \vdash .$$

Note that F *in the conjunction is actually superfluous. This single sequent represents the content of a proof with a single* Π_1 *cut of the sequent* $\forall xF \vdash$*.*

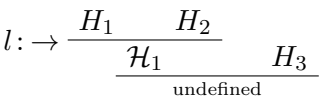
Figure 3.1: Extracting an extended Herbrand sequent of a proof with several Π_1 cuts



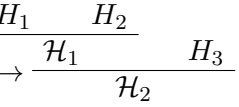
(a) A schema of a proof with two Π_1 cuts



(b) Replacing the subproofs with their propositional content, i.e. their Herbrand sequents



(c) Creation of the first Π_1-**EHS**



(d) Creation of the second Π_1-**EHS**

The following definition gives us the general form of extended Herbrand sequents for Π_1 cuts, even for proofs with more than a single Π_1 cut. The general idea is the same as shown in Example 7 and depicted in Figure 3.1. For a proof with Π_1 cuts of a sequent S_5 (see Figure 3.1a) we define a single tautological sequent $\mathcal{H}_2$ (see Figure 3.1d) containing only propositional formulas representing the content of the proof. The first part of $\mathcal{H}_2$ consists of instantiations of the formulas in S_5. This corresponds to the instantiations of F in Example 7. The second part consists of formulas of the form $G \to \bigwedge_{i \in I} G\sigma_i$ where σ_i are substitutions. G represents a cut formula and the substitutions σ_i tell us how the cut formula is instantiated when occurring on the left side of the sequent (Note that a Π_1-cut formula on the right side of a sequent is only instantiated with an eigenvariable). The logical connectives $\to$ and $\bigwedge$ merely are a product of writing all information of the two branches of a cut rule into one sequent, as in the last steps of Example 7. As soon as we consider more than one Π_1 cut, we have to take care of the eigenvariable conditions. Assume a proof with many Π_1 cuts for which we want to construct a Π_1-**EHS**, for instance as in Figure 3.1a. The topmost cut (Cut_1 of Figure 3.1a) can be treated as in Example 7 giving us a representation, i.e. a Π_1-**EHS** (cf. $\mathcal{H}_1$ in Figure 3.1c), of a subproof which might occur in a branch of another Π_1 cut (S_3 is a premise of Cut_2, cf. Figure 3.1a). Let us assume this cut occurs below and there is no cut in between. Then for one branch of this cut rule (Cut_2) the representation is not longer a Herbrand sequent (more precisely a midsequent) but a Π_1-**EHS** ($\mathcal{H}_1$ of Figure 3.1c) which can be treated exactly the same way. The resulting Π_1-**EHS** ($\mathcal{H}_2$ of Figure 3.1d) contains the information of the topmost (Cut_1) and the second topmost cut (Cut_2). Moreover, the Π_1-**EHS** for the second topmost Π_1 cut (Cut_2) contains the eigenvariables of the topmost Π_1 cut (Cut_1) whereas the converse does not hold. The different occurrences of the eigenvariables is expressed in the additional variable conditions of Definition 21.

Definition 21 (Extended Herbrand Sequent for Π_1 Cuts) *Let $\vec{r}_1, \ldots, \vec{r}_l$ be tuples of terms, let $G_1, \ldots, G_m$ be quantifier-free formulas, let $\alpha_1, \ldots, \alpha_m$ be variables, and let*

$s_{i,j}$ for $1 \leq i \leq m, 1 \leq j \leq n_j$ be terms such that

$$\mathcal{V}(G_i) \subseteq \{\alpha_i, \ldots, \alpha_m\} \text{ for all } i \text{ and}$$
$$\mathcal{V}(s_{i,j}) \subseteq \{\alpha_{i+1}, \ldots, \alpha_m\} \text{ for all } i, j.$$

Then the sequent

$$\mathcal{H} \stackrel{def}{=} F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_l], G_1 \to \bigwedge_{j=1}^{n_1} G_1[\alpha_1\backslash s_{1,j}], \ldots, G_m \to \bigwedge_{j=1}^{n_m} G_m[\alpha_m\backslash s_{m,j}] \vdash$$

is called an extended Herbrand sequent for Π_1 cuts *(shorthand: Π_1-**EHS**) of $\forall\vec{x}F \vdash$ if $\mathcal{H}$ is a tautology.*

*The complexity of a Π_1-**EHS** is defined as $|\mathcal{H}|_{\Pi_1} =_{def} \sharp\{\vec{r}_1, \ldots, \vec{r}_l\} + \sum_{j=1}^{m} n_j$.*

While the first part, i.e.

$$F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_l] \vdash,$$

looks similar to the usual Herbrand sequent for end-sequents of the form $\forall\vec{x}F \vdash$, the second part is completely new. In Example 7, we extracted a Π_1-**EHS** for a given proof. Now, we want to construct a proof for a given Π_1-**EHS**. In the example below, we consider again only a single Π_1 cut. The extension to many Π_1 cuts can be achieved in two ways: a naive one leading to unnecessary duplications of formulas and an elaborated one using interpolation. After the example, we will briefly discuss the naive way. For the advanced techniques see [HLRW14].

Example 8 *Let P, a, f, $f^l a$, $\forall x F$, and $\forall y G_1$ be as in Example 7. Then*

$$\begin{aligned} S \stackrel{def}{=}\ & Pa \wedge (P\alpha \to Pf\alpha) \wedge \neg Pf^4 a, \\ & Pa \wedge \left(Pf\alpha \to Pf^2\alpha\right) \wedge \neg Pf^4 a, \\ & \neg P\alpha \vee Pf^2\alpha \to \left(\neg Pa \vee Pf^2 a\right) \wedge \left(\neg Pf^2 a \vee Pf^4 a\right) \vdash \end{aligned}$$

*is a Π_1-**EHS** with $m = 1$ and $\alpha_1 = \alpha$. This is almost the same Π_1-**EHS** as in Example 7, but we dropped the superfluous conjunct F. By inverting the steps, we can reconstruct a proof with a single Π_1 cut. Since S is a tautology, also*

$$\begin{aligned} & Pa \wedge (P\alpha \to Pf\alpha) \wedge \neg Pf^4 a, \\ & Pa \wedge \left(Pf\alpha \to Pf^2\alpha\right) \wedge \neg Pf^4 a \ \vdash \neg P\alpha \vee Pf^2\alpha \end{aligned}$$

and

$$\begin{aligned} & Pa \wedge (P\alpha \to Pf\alpha) \wedge \neg Pf^4 a, \\ & Pa \wedge \left(Pf\alpha \to Pf^2\alpha\right) \wedge \neg Pf^4 a, \\ & \left(\neg Pa \vee Pf^2 a\right) \wedge \left(\neg Pf^2 a \vee Pf^4 a\right) \ \vdash \end{aligned}$$

are tautologies which are the premises of an $l\colon\to$-rule with S as conclusion. This implies that we are able to prove the two sequents

$$\forall xF \vdash \forall yG_1[\alpha\backslash y] \text{ and}$$
$$\forall xF, \forall yG_1[\alpha\backslash y] \vdash$$

which leads to the proof

$$\dfrac{\dfrac{\dfrac{\vdots}{\forall xF, F[x\backslash\alpha], F[x\backslash f\alpha] \vdash G_1}\, l\colon\forall}{\dfrac{\vdots}{\forall xF \vdash \forall yG_1[\alpha\backslash y]}} \qquad \dfrac{\dfrac{\dfrac{\vdots}{\forall xF, \forall yG_1[\alpha\backslash y], G_1[\alpha\backslash a], G_1[\alpha\backslash f^2a] \vdash}\, l\colon\forall}{\dfrac{\vdots}{\forall xF, \forall yG_1[\alpha\backslash y] \vdash}}}{\forall xF \vdash}\, Cut$$

*This example shows how proofs with single Π_1 cuts can be obtained from Π_1-**EHS**s. In order to see that we are able to extend this procedure in a naive way to many Π_1 cuts, let us sketch the case with two Π_1 cuts. Assume the Π_1-**EHS***

$$\Gamma, A \to \bigwedge_{i\in I} A\sigma_i, B \to \bigwedge_{j\in J} B\tau_j \vdash$$

where Γ consists of formulas $D[x\backslash r_i]$ for a quantifier-free formula D, A corresponds to G_1 and B corresponds to G_2 of Definition 21, and the variable conditions are also according to the definition. Let $\mathcal{V}(A) \subseteq \{y, z\}$ and $\mathcal{V}(B) \subseteq \{z\}$. Then also the sequents

$$\Gamma, \bigwedge_{j\in J} B\tau_j \vdash A,$$
$$\Gamma, \bigwedge_{i\in I} A\sigma_i \vdash, B,$$
$$\Gamma \vdash A, B, \text{ and}$$
$$\Gamma, \bigwedge_{i\in I} A\sigma_i, \bigwedge_{j\in J} B\tau_j \vdash$$

are provable, and moreover, the sequents

$$\forall xD, \forall zB \vdash \forall yA,$$
$$\forall xD, \forall yA \vdash B,$$
$$\forall xD \vdash \forall yA, B, \text{ and}$$
$$\forall xD, \forall yA, \forall zB \vdash$$

are provable. This allows us to derive $\forall xD \vdash$ with three applications of the cut rule:

$$\dfrac{\dfrac{\dfrac{\vdots}{\forall xD \vdash \forall yA, B} \quad \dfrac{\vdots}{\forall xD, \forall yA \vdash B}}{r\colon\forall\dfrac{\forall xD \vdash B}{\forall xD \vdash \forall zB}} \quad \dfrac{\dfrac{\vdots}{\forall xD, \forall zB \vdash \forall yA} \quad \dfrac{\vdots}{\forall xD, \forall yA, \forall zB \vdash}}{\forall xD, \forall zB \vdash}}{\forall xD \vdash}\,.$$

Note that the $r:\forall$-rule occurs necessarily after the cut with $\forall yA$ as cut formula since the corresponding eigenvariable might occur in A.

*The drawback of the naive way is that we increase the number of Π_1 cuts in comparison to the number of implications in the Π_1-**EHS**. This can be avoided by ensuring a linear structure of the proof with Π_1 cuts. In [HLRW14], the authors show via interpolation that we do not have to increase the number of Π_1 cuts.*

As already mentioned, we do not show here how a proof with many Π_1 cuts can be obtained from a Π_1-**EHS** without redundant duplications. This can be found in [HLRW14]. Even though, the result is an essential improvement of the described method, it is not necessary for the understanding of the course of this thesis. For completeness, we only mention the following result guaranteeing the exact relationship between Π_1-**EHS**s and proofs with Π_1 cuts. Note that the result below with its one to one complexity relations requires a proof via interpolation. The definitions of the complexity measures $|\cdot|_q$ and $|\cdot|_{\Pi_1}$ can be found in Definition 10 and Definition 21.

Theorem 4 (See [HLRW14]) *$\forall xF \vdash$ has a proof φ with Π_1 cuts such that $|\varphi|_q = l$ iff it has an Π_1-**EHS** $\mathcal{H}$ with $|\mathcal{H}|_{\Pi_1} = l$.*

The proof relies on Craig's interpolation theorem (see [Cra57] and the presentation in [Tak87, Chapter 4 Paragraph 23]) and is in this form a generalization of Proposition 2 in [HLW12] which shows the result for a single Π_1 cut.

3.2 Schematic Π_1 Grammars

In [Het12, AHL15, Het11], Π_1-**EHS**s are investigated via totally rigid acyclic tree grammars (see Definition 20). This is part of the aim to define a direct correspondence of cut-elimination and cut-introduction in formal grammar theory. The whole picture is depicted in Figure 3.2 taken from [HLRW14]. It shows six objects, either from proof theory or from formal grammar theory, and their relationship. While the objects on the right are in some sense uncompressed, i.e. proofs without cuts, a list of all grounded formulas (Herbrand sequent), or a list of all words in a language, the objects on the left are their compressed analogues: A proof with Π_1 cuts can be transformed into a cut-free proof via cut-elimination which usually increases the proof size (line 1 of Figure 3.2); In Corollary 2, we will show that a Π_1-**EHS** can be transformed into a Herbrand sequent (line 2 of Figure 3.2); Moreover, a schematic Π_1 grammar ($\mathbf{S\Pi_1\text{-}G}$) is a tree grammar that by definition has a corresponding language which might cover the Herbrand term set of a Herbrand sequent (line 3 of Figure 3.2). The horizontal relationships show that all lines do basically the same, i.e. a correspondence of cut-elimination. Theorem 4 proves that we can translate every proof with Π_1 cuts into a Π_1-**EHS** and vice versa preserving the complexity. Later, we will see that every Π_1-**EHS** can be represented by a schematic Π_1 grammar and vice versa (cf. Theorem 6). Since the relationships hold analogously on

Figure 3.2: Proof-theoretic setting of Π_1-cut introduction

proof φ with Π_1 cuts	$\overset{\text{cut-elimination}}{\longrightarrow}$	cut-free proof χ
$\updownarrow$ Theorem 4		$\updownarrow$ Theorem 3
Π_1-**EHS**	$\overset{\text{Corollary 2}}{\longrightarrow}$	Herbrand sequent
$\updownarrow$ Theorem 6		$\updownarrow$ trivial
SΠ_1-**G**s $\mathcal{G}$	$\overset{\text{defines}}{\longrightarrow}$	language $\mathcal{L}(\mathcal{G})$

the right side, we could perform cut-elimination via defining schematic Π_1 grammars, computing its language, and constructing a propositional proof of the corresponding Herbrand sequent.

In this section, we establish the relationship between Π_1-**EHS**s and schematic Π_1 grammars and thereby the relationship between proofs with Π_1 cuts and tree grammars. Nonetheless, note the general aim of the chapter which is to invert cut-elimination for Π_1 cuts. This consists of two major parts. On one hand, we have to show that every schematic Π_1 grammar allows us to construct a Π_1-**EHS** which is part of this section. On the other hand, we need to find a tree grammar whose language corresponds to a Herbrand sequent. Then we could move clockwise through the objects of Figure 3.2 starting from a cut-free proof of an end sequent S over a Herbrand sequent, a schematic Π_1 grammar, a Π_1-**EHS** to a proof with Π_1 cuts of the same sequent S. In the whole work presented here, we focus on the relationship between tree grammars, extended Herbrand sequents, and proofs with cuts which is of theoretical nature. For the problem of computing a grammar, which requires more practical evaluation, we give the necessary background, discuss occurring issues, and give a prototype algorithm (cf. Section 3.5 and Chapter 5).

In order to define tree grammars that correspond to Π_1-**EHS**s, we have to translate objects of first-order logic to symbols of a tree grammar and vice versa. Since formulas do not have a direct opponent in tree grammars, we define *term representations*: h_F is the representation of a formula F in a tree grammar. In order to translate such term representations back into formulas we introduce the **-operator* which replaces each term representation by its original formula ($h_F{}^* =_{def} F$). Note that we can only translate back into first-order logic, i.e. the *-operator does not apply to an arbitrary tree grammar. There has to be a designated term that represents a formula. In the Definition 24 of general schematic Π_1 grammars, we refer to this functions as freshly introduced.

Definition 22 *Let F be a quantifier-free l-ary formula and $\mathcal{G}$ be an arbitrary tree grammar. Then the* term representation *h_F of F in $\mathcal{G}$ is an l-ary function not occurring in the signature of $\mathcal{G}$ (since we consider in this work always at most one grammar at the time we do not denote its dependency on $\mathcal{G}$).*

Let h_F be an l-ary term representation of some formula F and $\vec{r}$ be an l-ary tuple of terms. Then we denote the back translation of $h_F\vec{r}$ by the *-operator*, i.e. $h_F\vec{r}^*$ being defined as $h_F\vec{r}^* =_{def} F\vec{r}$.*

This allows us to define tree grammars that directly correspond to a Π_1-**EHS**. For this reason, we introduce the term representation of the formula in the context h_F, collect all its term instantiations $\vec{r}_1, \ldots, \vec{r}_l$, and define the starting points of every derivation in the grammar by creating the productions $\{\tau \to h_F\vec{r}_i \mid 1 \leq i \leq l\}$. The terms $h_F\vec{r}_i$ contain the eigenvariables $\alpha_1, \ldots, \alpha_m$ which are considered as nonterminals. In the original proof, the eigenvariables occur in Π_1-cut formulas that are also instantiated with the terms $s_{i,j}$ for $1 \leq i \leq l, 1 \leq j \leq n_i$. For Π_1 cuts the underlying intuition is quite natural: Proving a formula (the cut-formula) once for an independent value (for the eigenvariable α_i), we can add it to the context (the left side of the sequent) and instantiate it with the terms $s_{i,1}, \ldots, s_{i,n_i}$. As a consequence, if the instantiations of the context contain an eigenvariable α_i (in a proof with Π_1 cut), the proof after cut-elimination contains none of the eigenvariables α_i, and all its occurrences should be replaced by the instantiations $s_{i,1}, \ldots, s_{i,n_i}$. In the schematic Π_1 grammar, we express this via the productions $\{\alpha_i \to s_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n_m\}$.

Definition 23 *Let*

$$\mathcal{H} = F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_l], G_1 \to \bigwedge_{j=1}^{n_1} G_1[\alpha_1\backslash s_{1,j}], \ldots, G_m \to \bigwedge_{j=1}^{n_m} G_m[\alpha_m\backslash s_{m,j}] \vdash$$

*be an Π_1-**EHS**. Then the* schematic Π_1 grammar $\mathcal{G}(\mathcal{H})$ *corresponding to $\mathcal{H}$ is defined as the totally rigid acyclic tree grammar $\langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ with $N = \{\tau, \alpha_1, \ldots, \alpha_m\}$, Σ being the signature of $\mathcal{H}$ plus the term representation h_F of F with $\boldsymbol{a}(h_F) = \boldsymbol{a}(F)$, and $\mathrm{Pr} = \{\tau \to h_F\vec{r}_i \mid 1 \leq i \leq l\} \cup \{\alpha_i \to s_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n_m\}$.*

This definition yields an object that is related to a Π_1-**EHS**, since we used a Π_1-**EHS** to construct the grammar. It remains to show that the relationship can be reversed. This is not obvious, since there is *no information about the cut formula* in the schematic Π_1 grammar, whereas the context and all instantiations have a one-to-one correspondence. For this reason, we have to define schematic Π_1 grammars independent from Π_1-**EHS**s.

Indeed, the main problem of Π_1-cut introduction is to show that the left side of Figure 3.2 is invertible. If we are able to prove this, then the cut-elimination procedure described above is also invertible whenever there is a schematic Π_1 grammar that covers the language of a Herbrand term set: Assume a cut-free proof χ. There is always a Herbrand sequent which can be translated into a Herbrand term set H_s. If we find a schematic Π_1 grammar whose language is a superset of H_s, then this superset is as well a Herbrand term set, potentially containing some term representations of redundant formulas (for practice it might be beneficial to allow supersets). The existence of a schematic Π_1 grammar then implies the existence of a proof with Π_1 cuts.

Definition 24 (Schematic Π_1 Grammar) *Let $\vec{r}_1, \ldots, \vec{r}_l$ be terms with $\boldsymbol{l}(\vec{r}_k) = n$ for all $k \in \mathbb{N}_l$, let $\alpha_1, \ldots, \alpha_m$ be variables, and let $s_{i,j}$ for $1 \leq i \leq m, 1 \leq j \leq n_j$ be terms such that*

$$\mathcal{V}(s_{i,j}) \subseteq \{\alpha_{i+1}, \ldots, \alpha_m\} \text{ for all } i, j.$$

Then we call the totally rigid acyclic tree grammar $\langle \tau, N, \Sigma, \Pr \rangle$ with $N = \{\tau, \alpha_1, \ldots, \alpha_m\}$ and $\Pr = \{\tau \to h_F \vec{r}_i \mid 1 \leq i \leq l\} \cup \{\alpha_i \to s_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n_m\}$ where h_F is a fresh function with $\boldsymbol{a}(h_F) = n$ a schematic Π_1 grammar *(shorthand: $\mathbf{S}\Pi_1$-$\mathbf{G}$).*

The definition is a simple abstraction of Definition 23. Only the direct correspondence to first-order logic, i.e. the formula corresponding to h_F, is missing. As we will see later (cf. Theorem 6), if the language of a $\mathbf{S}\Pi_1$-**G** covers a Herbrand term set, there is a corresponding Π_1-**EHS**.

In order to ensure, that the relationship between $\mathbf{S}\Pi_1$-**G**s and Π_1-**EHS**s is meaningful, we have to establish a measure. Since we considered so far mainly the non-propositional part of proofs, it is natural to define a complexity measure of the grammar that is related to the quantifier complexity of proofs. In the motivation of Definition 23, we explained the relationship between the terms of the production rules and the instantiations within a proof with Π_1 cuts. Hence, we define the complexity of a $\mathbf{S}\Pi_1$-**G** as the number of weak quantifier inferences necessary to introduce the terms occurring as an argument of h_F ($\sharp T$) plus the number of production rules with an eigenvariable on its left side ($m - l$). Theorem 6 guarantees that the quantifier complexity of a proof constructed according to a $\mathbf{S}\Pi_1$-**G** is equal to the complexity of the grammar.

Definition 25 (Complexity of $\mathbf{S}\Pi_1$-Gs) *Let $\mathcal{G}$ be an $\boldsymbol{S}\Pi_1$-$\boldsymbol{G}$ of the form $\langle \tau, N, \Sigma, \Pr \rangle$ and T be the set of all tuples of terms $\vec{r}$ such that $\tau \to h_F \vec{r}$ is a production rule in* Pr *and h_F is the freshly introduced function. Then the complexity $|\mathcal{G}|$ of $\mathcal{G}$ is defined as*

$$\sharp T - l + m$$

where l is the number of tuples in T, $\sharp T$ is the instantiation complexity of T (see Definition 6), and m is the number of production rules in $\mathcal{G}$.

Remark 2 *Note that we subtract l in Definition 25 to avoid counting the production rules with the starting symbol on the left twice.*

Example 9 *Consider again the Π_1-$\boldsymbol{EHS}$ defined in Example 8. Then $\mathcal{G} = \langle \tau, N, \Sigma, \Pr \rangle$ with $N = \{\tau, \alpha\}$, $\Sigma = \{f, a, h_F\}$ where h_F is a fresh unary function, and $\Pr = \{\tau \to h_F\alpha, \tau \to h_F f\alpha, \alpha \to a, \alpha \to f^2 a\}$ is the $\boldsymbol{S}\Pi_1$-$\boldsymbol{G}$ corresponding to S. The fresh function h_F serves as an encoding of the formula F which was instantiated once with α and once with $f\alpha$. The other productions correspond to instantiations of the cut formula; we proved $\forall x.\neg Px \vee Pf^2x$ (with the eigenvariable α) and used two instantiations of it, i.e. with the*

substitutions $[x\backslash a]$ *and* $[x\backslash f^2 a]$. *Note that there is no information about the cut formula itself in the grammar, but there is information about the quantifier rules applied to the cut formula. As we will see in the course of this chapter, the latter suffices to deduce a cut formula.*

The complexity of the grammar is

$$|\mathcal{G}| = \sharp\{h_F\alpha, h_F f\alpha\} - 2 + 4 = 2 - 2 + 4 = 4.$$

The language is

$$\mathcal{L}(\mathcal{G}) = \{h_F a, h_F f^2 a, h_F f a, h_F f f^2 a\}$$

which is a Herbrand term set of $\forall x F \vdash$ *(see also Example 10).*

As mentioned above, one can perform cut-elimination based on $\mathbf{S}\Pi_1$-**G**s by computing its language and constructing the corresponding Herbrand sequent as in Example 9.

Theorem 5 (See [Het12, AHL15, Het11]) *If* $\mathcal{H}$ *is an* Π_1-***EHS*** *of* $\forall \vec{x} F \vdash$*, then* $\mathcal{L}(\mathcal{G}(\mathcal{H}))$ *is a Herbrand term set* H_s *of* $\forall \vec{x} F \vdash$ *where the fresh function (see Definition 15) is the term representation* h_F *of* F.

Proof:
See [Het12, AHL15, Het11]. □

Corollary 2 *If* $\mathcal{H}$ *is an* Π_1-***EHS*** *of* $\forall \vec{x} F \vdash$*, then* $\mathcal{L}(\mathcal{G}(\mathcal{H}))^* \vdash$ *is a Herbrand sequent of* $\forall \vec{x} F \vdash$ *where* $(\cdot)^*$ *is defined as in Definition 22.*

Proof:
This is a direct consequence of 5. □

In Example 9, we defined a $\mathbf{S}\Pi_1$-**G** for the running example of this chapter and computed its language. Theorem 5 states that the language is a Herbrand term set. In order to check this, we need to translate back to first-order logic via $(\cdot)^*$. Since the tree grammar was defined according to a proof, ${h_F}^*$ is defined and we can proceed as in the following example.

Example 10 *Let $\mathcal{G}$ be the $\boldsymbol{S\Pi_1}$-$\boldsymbol{G}$ defined in Example 9. The function h_F encodes the formula $F = Pa \wedge (\neg Px \rightarrow Pfx) \wedge \neg Pf^4a$ and therefore, $\mathcal{L}(\mathcal{G}(\mathcal{H}))^* \vdash$ is*

$$\begin{aligned} Pa \wedge (Pa \rightarrow Pfa) \wedge \neg Pf^4a, \\ Pa \wedge \left(Pf^2a \rightarrow Pf^3a\right) \wedge \neg Pf^4a, \\ Pa \wedge \left(Pfa \rightarrow Pf^2a\right) \wedge \neg Pf^4a, \\ Pa \wedge \left(Pf^3a \rightarrow Pf^4a\right) \wedge \neg Pf^4a \vdash \end{aligned}$$

which is a Herbrand sequent for $\forall x F \vdash$.

When considering again Figure 3.2 we see that Theorem 5 and Corollary 2 allow us to follow one "path" through the objects: For a proof with Π_1 cuts, we can define a Π_1-**EHS** and its corresponding $\mathbf{S\Pi_1}$-**G**. The latter defines a language which is provably a Herbrand term set (Theorem 5) and hence, gives us a Herbrand sequent H (Corollary 2). Since a Herbrand sequent is a tautology, we can find a propositional cut-free proof which, combined with the instantiation rules for the construction of H, is a cut-free proof of the original statement. Hence, we performed cut-elimination via formal grammars. Now, we want to invert this procedure. For this reason, we have to find for every $\mathbf{S\Pi_1}$-**G** a Π_1-**EHS**.

3.3 Schematic Extended Herbrand Sequents for Π_1 Cuts

On the one hand, we noticed in Section 3.2 that $\mathbf{S\Pi_1}$-**G**s can be used to compute a Herbrand sequent since their language covers the Herbrand term set (see Definition 15) and therefore, they contain the main information of a cut-free proof. On the other hand, we already mentioned in Example 9 that the grammars do not give any information about how the cut formula looks like. Without the corresponding Π_1-**EHS**, the $\mathbf{S\Pi_1}$-**G** is just a compressed representation of the instantiations. But if we were able to find always a cut formula for an arbitrary $\mathbf{S\Pi_1}$-**G**, they would yield much more: the $\mathbf{S\Pi_1}$-**G**s would be the exact correspondence of proofs with Π_1 cuts in formal language theory, they would allow us to compute proofs with cuts by simply computing a covering $\mathbf{S\Pi_1}$-**G** of the term representation of the Herbrand sequent. By covering we mean that the language of the grammar is a superset of the term set. Hence, the question arises whether we can construct cut formulas for $\mathbf{S\Pi_1}$-**G**s. Formally, this leads to the schematic extended Herbrand sequent for Π_1 cuts. It consists of all the information of a $\mathbf{S\Pi_1}$-**G**, but put in the format of a Π_1-**EHS**. Hence, we have to represent the missing cut formulas which is done by introducing second-order variables $X_1, \ldots, X_m$. The problem of finding cut formulas can thus be seen as second-order unification problem: there exists formulas $G_1, \ldots, G_m$ such that the replacement of the variables $X_1, \ldots, X_m$ with these formulas yields a Π_1-**EHS**. Everything apart from the cut formula/second-order variable is the same as in the definition of a Π_1-**EHS**.

Definition 26 (Schematic Extended Herbrand Sequent for Π_1 Cuts) *Let S be the provable sequent $\forall \vec{x} F \vdash$ and $[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{l} \vdash$ be a Herbrand sequent for S. Let $\mathcal{G} = \langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ be an $\boldsymbol{S\Pi_1}$-$\boldsymbol{G}$ with the fresh function h_F where $\boldsymbol{a}(h_F) = \boldsymbol{l}(\vec{r}_i)$, with $N =_{def} \{\tau, \alpha_1, \ldots, \alpha_m\}$, and with*

$$\begin{aligned} \mathrm{Pr} \overset{def}{=} & \{\tau \to h_F \vec{r}_i \mid 1 \leq i \leq a\} \cup \\ & \{\alpha_i \to s_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n_m\} \\ & \mathcal{V}(s_{i,j}) \subseteq \{\alpha_{i+1}, \ldots, \alpha_m\} \text{ for all } i, j \end{aligned}$$

such that the language of $\mathcal{G}$ covers the Herbrand term set, i.e.

$$\mathcal{L}(\mathcal{G}) \supseteq \{h_F \vec{r}_i \mid i \in \mathbb{N}_l\}.$$

Then

$$\mathcal{S}(S) \overset{def}{=} [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a}, X_1\alpha_1 \to \bigwedge_{j=1}^{n_1} X_1 s_{1,j}, \ldots, X_m\alpha_m \to \bigwedge_{j=1}^{n_m} X_m s_{m,j} \vdash$$

where $X_1, \ldots, X_m$ are monadic second-order variables is called a schematic extended Herbrand sequent for Π_1 cuts *(shorthand:* Π_1-**SEHS**$)$.

A solution *of a Π_1-$\boldsymbol{SEHS}$ $\mathcal{S}(S)$ is a substitution $\sigma = [X_i\backslash\lambda\alpha_i.G_i]_{i=1}^{m}$ such that $\mathcal{F}(G_i) \subseteq \{\alpha_i, \ldots, \alpha_m\}$ and $\mathcal{S}(S)\sigma$ is a tautology.*

Example 11 *Let $\mathcal{G}$ be defined as in Example 9 and let P, a, f, $f^l a$, and $\forall x F$ be defined as in Example 7. Then*

$$\mathcal{S} \overset{def}{=} F[x\backslash\alpha], F[x\backslash f\alpha], X\alpha \to \left(Xa \wedge Xf^2 a\right) \vdash$$

is a Π_1-$\boldsymbol{SEHS}$ of $\forall x F \vdash$. Note that the sequent is equivalent to S of Example 8 if we replace X with $\lambda x.\,(\neg Px \vee Pf^2 x)$. Since S is a tautology, $\lambda x.\,(\neg Px \vee Pf^2 x)$ is a solution of $\mathcal{S}$. Moreover, the example shows that a solution of Π_1-$\boldsymbol{SEHS}$ gives us a Π_1-$\boldsymbol{EHS}$ at hand which can be used to construct a proof with Π_1 cuts as in Section 3.1 for a single Π_1 cuts or as in [HLRW14, EHL+18] for several Π_1 cuts.

Revisiting the way we have taken, we see that the existence of a $\mathbf{S\Pi_1}$-**G** for the Herbrand term set of a cut-free proof may lead to a proof with Π_1 cuts. Assume we already found an $\mathbf{S\Pi_1}$-**G**, then the main problem of Π_1-cut introduction reduces to find a solution for the corresponding Π_1-**SEHS** which gives us a Π_1-**EHS** at hand. This will be solved in the following section. The question how to find a grammar efficiently is addressed in [EEH17] which will be discussed in Section 3.5.

3.4 The Canonical Solution

In this section, we address the problem of finding a solution for a Π_1-**SEHS**. Fortunately, the solution of a Π_1-**SEHS** is fully determined by the formula F of the end-sequent $\forall x F \vdash$. This means that we do not only find at least one solution, but we can actually give a canonical solution which is based on the formula F. In case of a single Π_1 cut, we are able to give an intuition why this is possible. We know from cut-elimination via $\mathbf{S}\Pi_1$-**G**s that the language of a $\mathbf{S}\Pi_1$-**G** covers a Herbrand term set. The language for a single cut consists of productions $\alpha \to s$ applied to instantiations of the term representation of the context $h_F\vec{r}$. Hence, applying this productions as substitutions $[\alpha \backslash s]$ to the instantiations $F[\vec{x}\backslash\vec{r}]$ of the formula F yields a Herbrand sequent. Now, consider the form of a Π_1-**SEHS** with a single Π_1 cut

$$F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_a], X\alpha \to \bigwedge_{j=1}^{n} Xs_j \vdash$$

and apply an $l\colon\to$-rule backwards. We get the two premises

$$F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_a], \bigwedge_{j=1}^{n} Xs_j \vdash ,$$
$$F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_a] \vdash X\alpha.$$

If we replace X with a conjunction over all instantiations of F as they occur in the $\mathbf{S}\Pi_1$-**G**, $\bigwedge_{j=1}^{n} Xs_j \vdash$ turns into a Herbrand sequent. Thus, the first premise is provable. Moreover, $X\alpha$ becomes the conjunction over all instantiations of F, i.e. $F[\vec{x}\backslash\vec{r}_1] \wedge \ldots \wedge F[\vec{x}\backslash\vec{r}_a]$ making also the second premise provable. As soon as we consider several Π_1 cuts, the intuition becomes more complicated. Still, the main idea is to create on one side a Herbrand sequent and to match on the other side the instantiations of the context. The structure then looks like:

$$F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_a], C_1 \to C_2, \ldots, C_m \to C_{m+1} \vdash$$

where C_1 is the conjunction over all instantiations $F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_a]$ of the context, $C_i \to C_{i+1}$ is the implication corresponding to the i-th cut formula, and $C_{m+1} \vdash$ is a Herbrand sequent. Obviously, the sequent is a tautology, since $C_{m+1} \vdash$ is a Herbrand sequent. In general, the solution can be simplified, but at the moment we are merely interested in the solvability of Π_1-**SEHS**s.

Definition 27 (Canonical Substitution) *Let $\mathcal{S}$ be a Π_1-**SEHS**. Define*

$$C_1 \stackrel{def}{=} \bigwedge_{i=1}^{a} F[\vec{x}\backslash\vec{r}_i] \text{ and}$$
$$C_{i+1} \stackrel{def}{=} \bigwedge_{j=1}^{n_j} C_i[\alpha_i \backslash s_{i,j}] \text{ for } i = 1, \ldots, m.$$

Then

$$\sigma \stackrel{def}{=} [X_i \backslash \lambda\alpha_i.C_i]_{i=1}^m$$

is called the canonical substitution *of* $\mathcal{S}$.

Example 12 *Let* $\mathcal{S}$ *be defined as in Example 11. Then*

$$\begin{aligned}
C_1 &\stackrel{def}{=} F[x\backslash\alpha] \wedge F[x\backslash f\alpha] \\
&= Pa \wedge (P\alpha \to Pf\alpha) \wedge \neg Pf^4a \wedge \\
&\quad Pa \wedge \left(Pf\alpha \to Pf^2\alpha\right) \wedge \neg Pf^4a, \\
C_2 &\stackrel{def}{=} C_1[\alpha\backslash a] \wedge C_1[\alpha\backslash f^2a] = (F[x\backslash\alpha] \wedge F[x\backslash f\alpha])\,[\alpha\backslash a] \wedge (F[x\backslash\alpha] \wedge F[x\backslash f\alpha])\,[\alpha\backslash f^2a] \\
&= Pa \wedge (Pa \to Pfa) \wedge \neg Pf^4a \wedge \\
&\quad Pa \wedge \left(Pfa \to Pf^2a\right) \wedge \neg Pf^4a \wedge \\
&\quad Pa \wedge \left(Pf^2a \to Pf^3a\right) \wedge \neg Pf^4a \wedge \\
&\quad Pa \wedge \left(Pf^3a \to Pf^4a\right) \wedge \neg Pf^4a,
\end{aligned}$$

and $\sigma =_{def} [X\backslash\lambda\alpha.C_1]$ *is a solution of* $\mathcal{S}$ *since*

$$\mathcal{S}\sigma = F[x\backslash\alpha], F[x\backslash f\alpha], C_1 \to C_2 \vdash$$

is a tautology. Note that C_1 *is equal to* $F[x\backslash\alpha]$ *together with* $F[x\backslash f\alpha]$. *Hence,*

$$l{:}\to \frac{\dfrac{\vdots}{C_1 \vdash C_1} \qquad \dfrac{\vdots}{C_1, C_2 \vdash}}{C_1, C_1 \to C_2 \vdash}$$

sketches a valid proof if $C_1, C_2 \vdash$ *or even* $C_2 \vdash$ *is a tautology. Therefore, we compare* $C_2 \vdash$ *with the Herbrand sequent of Example 10. Since they are the same, we can conclude that* $\mathcal{S}\sigma$ *is a tautology. Thus* σ *is a solution.*

Indeed, a canonical substitution of a Π_1-**SEHS** is always a solution. In order to prove this, we consider at first the last element of the chain of implications in

$$F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_a], C_1 \to C_2, \ldots, C_m \to C_{m+1} \vdash,$$

i.e. C_{m+1} (cf. Lemma 1). If $C_{m+1} \vdash$ is a tautology (Herbrand sequent) and C_1 matches $F[\vec{x}\backslash\vec{r}_1], \ldots, F[\vec{x}\backslash\vec{r}_a]$, the canonical substitution is a solution (cf. Lemma 2). The proofs are straight forward and can be found in [HLRW14]. The interested reader might look them up or prove them as an exercise. We will henceforth call a canonical substitution a *canonical solution.*

Lemma 1 (See [HLRW14]) *Let $\mathcal{S}$ and C_i be defined as in Definition 27. Then $C_{m+1} \vdash$ is a tautology.*

Lemma 2 (See [HLRW14]) *Let $\mathcal{S}$ be a Π_1-**SEHS** and σ be its canonical substitution. Then σ is a solution of $\mathcal{S}$.*

Since a canonical substitution is always a solution, cut introduction via tree grammars is possible for Π_1 cuts. For every $\mathbf{S}\Pi_1$-**G**, we find a Π_1-**EHS** and thus, a proof with Π_1 cuts.

Theorem 6 (See [HLRW14]) *$\forall x F \vdash$ has a Π_1-**EHS** $\mathcal{H}$ with $|\mathcal{H}|_{\Pi_1} = l$ iff there is a $\mathbf{S}\Pi_1$-**G** $\mathcal{G}$ with $|\mathcal{G}| = l$ such that*

$$\bigwedge_{\vec{r} \in \mathcal{L}(\mathcal{G})} F[\vec{x} \backslash \vec{r}] \vdash$$

is a tautology.

This completes the theoretical backbone. Obstacles which are left to be tackled are the grammar computation and the *beautification* of the cut formula. Even though, there exists already work on that (cf. [EEH17, EHL$^+$18]), we will only sketch some of the main ideas in the following section.

3.5 Application of Π_1-Cut Introduction

In this section, we discuss two different applications of Π_1-cut introduction and sketch the grammar-computation approach of [EEH17]. We do not describe the algorithm for the grammar computation in full detail, but we explain the main structure and properties of the method. This provides the necessary basis for understanding its behaviour applied in the context of Π_1-cut introduction and in the context of Π_2-cut introduction (cf. Section 5.2). In literature [HLRW14, EHL$^+$18], there is also an alternative approach for computing $\mathbf{S}\Pi_1$-**G**s which we do not present here, since it is not used in the Π_2 case. Nonetheless, we will provide information about its performance in the applications.

Both applications of Π_1-cut introduction are taken from [EHL$^+$18]. On one hand, we look at the more theoretical outcome of cut introduction: it can introduce lemmas with mathematical meaning and makes thereby automatically generated proofs more human readable. On the other hand, we investigate the results of applying Π_1-cut introduction to a large set of cut-free proofs with respect to values such as compressibility, compression ratio, grammar size, etc.

3.5.1 Computation of Schematic Π_1 Grammars

Even though, formal grammars are heavily used for text compression [SS82, NMW97, LM99, KY00], there is little research about finding minimal tree grammars for a fixed type of tree grammar such as $\mathbf{S\Pi_1\text{-}G}$s. Usual methods for compressing sets of terms (most often just a single term) allow changing the type, which is forbidden in our setting. We need a fixed one because of the specific requirements in the context of cut introduction where only $\mathbf{S\Pi_1\text{-}G}$s have a direct correspondence to proofs with Π_1 cuts. For this reason, we consider the algorithm of [EEH17]. Another difference of this approach is that this algorithm might find grammars whose language is a superset of the set of terms we want to compress. Since the term set represents a Herbrand sequent which is a set of quantifier-free formulas, a superset is also a Herbrand sequent. It just contains some additional formulas.

The algorithm itself is based on a polynomial-time reduction to the **MaxSAT** optimization problem (for a list of **MaxSAT** solvers see [ALMP08]). **MaxSAT** is a variant of the Boolean satisfaction problem (**SAT**). The authors describe their method themselves in the following three steps (see the Introduction of [EEH17]):

1. Compute a larger grammar that covers the term set and contains a covering subgrammar of minimal size, in polynomial time.
2. Produce a **MaxSAT** problem that encodes the minimization of this large grammar.
3. Use a **MaxSAT** solver to obtain a solution to the **MaxSAT** problem, and return the minimal **VTRATG** corresponding to this solution.

VTRATG stands for *vectorial totally rigid acyclic tree grammars* which are totally rigid acyclic tree grammars (see Definition 20) where the nonterminals are vectors. They allow vectors in order to introduce Π_1 cuts with blocks of quantifiers, but obviously we can apply the algorithm also in our simpler setting. The minimization is with respect to the number of production rules.

An important assumption, the authors made, is to fix the number of nonterminals. This is due to two properties of the minimization: On the one hand, the reduction is only polynomial with respect to a fixed number of nonterminals. On the other hand, the problem of covering a language with a minimal tree grammar becomes trivial if we allow an arbitrary number of nonterminals.

Theorem 7 (See [EH18]) *Let T be finite set of terms, and $l_0, \ldots, l_m$ be natural numbers such that $|T| \leq \prod_{i=0}^{m} l_i$. Then there exists a **VTRATG** $\mathcal{G}$ of size $\sum_{i=0}^{m} l_i$ such that its language is T.*

As already mentioned in [EEH17]: "In particular, for every set of terms L of size $|L| \leq 2^n$ there exists a covering **VTRATG** of size $2n$." Nonetheless, this result does not directly

apply to $\mathbf{S}\Pi_1$-**G**s. The size of a **VTRATG** is determined by the number of production rules. Since the productions of a **VTRATG** map vectors to vectors, there is no direct correspondence to the quantifier complexity.

3.5.2 Introduction of Meaningful Lemmas

The following example shows that automatic cut introduction can formulate lemmas such as transitivity and anti-symmetry. Due to the advanced algorithm used for this test case, we only present the results. Note that techniques such as beautification are required but not defined in the present work.

Example 13 (See [EHL$^+$18]) *The example is based on Exercise 2 in Birkhoff's classic text book on lattice theory [Bir67]. Assume a lattice defined via meet and join and prove that whenever there is a cycle of four elements where each is smaller or equal to the next one, then all must be equal. One proof is to show first transitivity and then anti-symmetry. Formally, we can prove the sequent*

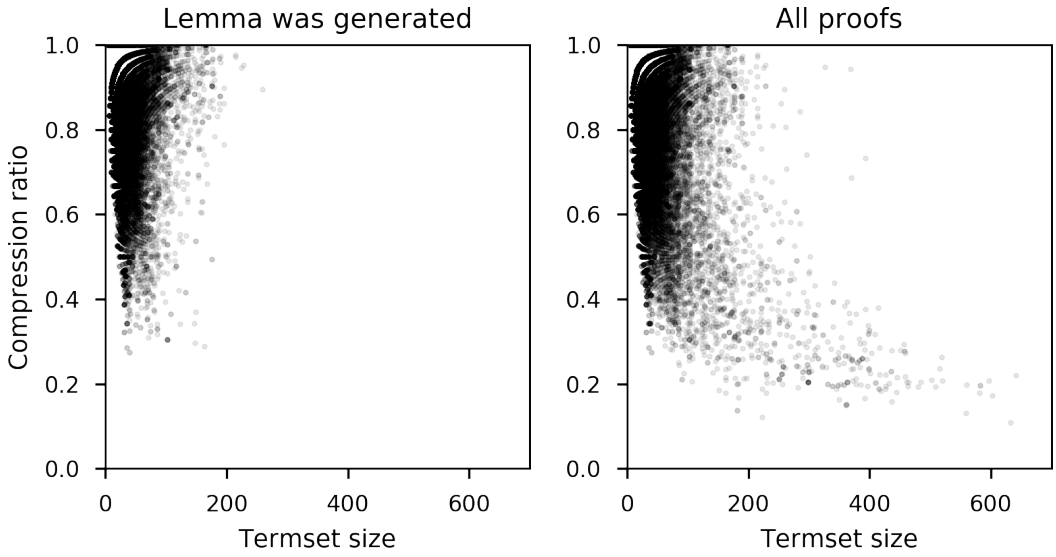
$$
\begin{aligned}
S \stackrel{def}{=} \quad & \forall x.x = x, \\
& \forall x, y, z.\, (x = y \wedge y = z) \to x = z, \\
& \forall x, y.x = y \to y = x, \\
& \forall x_1, x_2, x_3, x_4.\, (x_1 = x_2 \wedge x_3 = x_4) \to f\,(x_1, x_3) = f\,(x_2, x_4)\,, \\
& \forall x, y, z.f\,(f\,(x, y)\,, z) = f\,(x, f\,(y, z))\,, \\
& \forall x, y.f\,(x, y) = f\,(y, x) \\
& \vdash (f\,(a, b) = a \wedge f\,(b, c) = b \wedge f\,(c, d) = c \wedge f\,(d, a) = d) \to a = b \wedge b = c \wedge c = d
\end{aligned}
$$

where f denotes the meet, i.e. the greatest lower bound of two elements. Starting with a manually formalized proof containing the two mentioned lemmas, transitivity and anti-symmetry, we can apply cut-elimination. For the resulting cut-free proof, we can extract a Herbrand sequent and a Herbrand term set. As shown in [EHL$^+$18], we can automatically generate a $\boldsymbol{S}\Pi_1$-$\boldsymbol{G}$ (in the paper denoted as decomposition), compute the canonical solution, and get after some beautification the final solution

$$
\begin{aligned}
& ((f\,(\alpha_2, \alpha_1) = \alpha_2 \wedge \alpha_1 = f\,(\alpha_1, \alpha_2)) \to \alpha_1 = \alpha_2) \wedge \\
& ((f\,(\alpha_2, \alpha_3) = \alpha_2 \wedge f\,(\alpha_1, \alpha_2) = \alpha_1) \to f\,(\alpha_1, \alpha_3) = \alpha_1)\,.
\end{aligned}
$$

The beautification removes superfluous assumptions and direct copies of axioms included in the solution. Note that this increases the size of the decomposition.

Figure 3.3: Compression ratio (size of decomposition divided by size of term set) depending on the term set size. This Figure is taken from [EHL$^+$18].



3.5.3 Experiments with Thousands of Solutions from Theorem Provers

In order to show the practical relevance of Π_1-cut introduction, we consider the **TSTP** library (**T**housands of **S**olutions from **T**heorem **P**rovers, see [Sut09]). In [EHL$^+$18], the authors discuss the results after running Π_1-cut introduction on all importable first-order proofs available on November 2015. Even though the basis of the cut-introduction algorithm is as we presented it in this thesis, they use a much more advanced version applicable to proofs with equality (cf. also [HLW12, HLR$^+$14]). They were able to import 68198 out of 138005 proofs (49.41%) of which 32714 were trivial, i.e. every term started with a different symbol making a compression via Π_1-cut introduction impossible. After running the computation on a Debian Linux system with an Intel i5-4570 CPU and 8 GiB RAM with a maximal runtime of one minute for each instance, they got 19122 (53.90%) decompositions ($\mathbf{S}\Pi_1$-**G**s) and 12035 lemmas (33.92%).

Figure 3.3 illustrates the achieved compression in terms of quantifier complexity. Since the quantifier complexity can already be computed knowing only the $\mathbf{S}\Pi_1$-**G**, we can plot the compressions even for cases in which the construction of the canonical solution fails (due to its large size and the runtime). The figure shows the relation between the compression ratio and the term-set size. Since the size of the term set corresponds to a Herbrand sequent, it represents the quantifier complexity of a cut-free proof. The compression ratio is the size of the decomposition ($\mathbf{S}\Pi_1$-**G**) divided by the size of the term set. Hence, a compression ratio of one corresponds to no compression whereas a

compression ratio of $1/a$ with $1 < a$ denotes that we compressed the quantifier complexity by a factor of a. The figure is split in two parts: in the left part all results are plotted where a lemma was computed, in the right part the results are added where only a decomposition was computed, but no lemma.

Most proofs were computed for term sets of a size between 10 and 50. The compression ratio reaches for a few examples the value 0.3, but according to [EHL$^+$18], the values are most often 0.5. Due to the relative small natural numbers of the term-set sizes, we can observe some concrete lines in the plot. Comparing the left part with the right part, the cut introduction fails especially for large term-set sizes. Here, the canonical solution probably becomes to large, since it grows exponentially. Even tough, the final solution might be much smaller, the algorithm still computes the canonical solution. An alternative could be the computation of a solution via the unification method that will be presented in Section 4.7 (even tough, it was designed for Π_2-cut introduction). This method does not guarantee to find a solution, but it especially searches for small solutions in a direct way (cf. Section 5.3). Since the compression ratio for large term-set sizes improves considerably (see the right plot of Figure 3.3), we should also investigate heuristic methods.

3.6 The Possible Compression of Π_1 Cuts

In the previous section, we provided a practical evaluation of Π_1-cut introduction. Nonetheless, we did neither give theoretical bounds to the maximal compression of a single Π_1 cut nor of several Π_1 cuts. In this section, we will briefly discuss the maximal compression of a single Π_1 cut which also yields a maximal compression of several Π_1 cuts and show an example that achieves the best compression for several Π_1 cuts. For simplicity, we will only consider the quantifier complexity (see Definition 10).

3.6.1 The Maximal Compression of a single Π_1 Cut

In the course of Chapter 3, we showed a direct correspondence between $\mathbf{S}\Pi_1$-**G**s and proofs with Π_1 cuts. More precisely, we saw that even the complexity of a $\mathbf{S}\Pi_1$-**G** gives us the quantifier complexity of a corresponding proof with Π_1 cuts (see Theorem 6). The compression is then given by comparing the size of the Herbrand term set with the size of the $\mathbf{S}\Pi_1$-**G**. Let us consider a cut-free proof φ, its corresponding Herbrand term set H_s, and a proof χ with a single Π_1 cut which corresponds to the $\mathbf{S}\Pi_1$-**G** $\mathcal{G}$ that covers H_s. The size of a $\mathbf{S}\Pi_1$-**G** is

$$\sharp T + m - l$$

(see Definition 25). $\sharp T$ is the instantiation complexity (see Definition 6) of the tuples of terms T, which is the quantifier complexity of instantiating the context in χ. l is the number of productions starting with the starting symbol τ and m is the number of all production rules. Hence, $m - l$ is the number of production rules starting with an eigenvariable as nonterminal. Since χ contains only a single Π_1 cut, there is only a single

eigenvariable α. Thus, a term in the language of $\mathcal{G}$ is of the form

$$r[\alpha\backslash s]$$

where $\tau \to r$ and $\alpha \to s$ are productions of $\mathcal{G}$ (note that the grammar is totally rigid, see Definition 19 and Definition 20). In order to maximize the number of terms we can cover by such a grammar, we assume that for each term t in the language there is a unique pair of production rules

$$(\tau \to r, \alpha \to s)_t$$

such that $r[\alpha\backslash s] = t$. The number of such pairs is $\sharp T \cdot (m - l)$. Since H_s is covered by the language of $\mathcal{G}$, we can assume that $\sharp H_s$ is equal to $\sharp T \cdot (m - l)$ to achieve the best compression. Now, we have to find two natural numbers p and q with $p =_{def} \sharp T$ and $q =_{def} m - l$ such that $p + q$, i.e. $\sharp T - l + m$, is minimal and $p \cdot q = \sharp T \cdot (m - l)$. Let $n =_{def} \sharp T \cdot (m - l)$, then $p+q$ is minimal if $p = q = \sqrt{n}$. This corresponds to the problem of finding a rectangle with minimal perimeter for a fixed area. Therefore, we achieve the best compression if the size of the $\mathbf{S\Pi_1}$-$\mathbf{G}$ is the square root of the size of the Herbrand term set, i.e. the rectangle is a square. This corresponds to a quadratic compression.

Proposition 8 *Let φ be a minimal cut-free proof of a sequent S and $\mathcal{G}$ a $\boldsymbol{S\Pi_1}$-$\boldsymbol{G}$ covering the Herbrand term set of φ containing only two nonterminals. Let χ be a proof of S after applying Π_1-cut introduction with respect to $\mathcal{G}$ and according to Theorem 6. Then*

$$|\chi|_q \leq |\varphi|_q{}^2.$$

When considering $\mathbf{S\Pi_1}$-$\mathbf{G}$s for proofs with several Π_1 cuts, this effect is iterated. Consider a $\mathbf{S\Pi_1}$-$\mathbf{G}$ $\mathcal{G}$ where n_1 is the number of productions with the starting symbol on the left. Let $\alpha_2, \ldots, \alpha_m$ be the other nonterminals such that the productions of the form $\alpha_i \to r$ map on terms r that contain only nonterminals with higher index, i.e. the nonterminals of r are a subset of $\{\alpha_{i+1}, \ldots, \alpha_m\}$. By n_i we denote the number of production rules with the nonterminal α_i on the left. The maximal number of different terms in the language of $\mathcal{G}$ is

$$n_1 \cdot \ldots \cdot n_m.$$

The size of the grammar is given by

$$\sharp T + n_2 + \ldots + n_m.$$

Again the maximal compression is achieved if $\sharp T = n_1 = n_2 = \ldots = n_m$. Moreover, if we were able to introduce n_1 cuts (cf. the following subsection), i.e. $m = n_1 = n_2 = \ldots = n_m$, then the compression would be exponential. Let n be the number of Π_1 cuts and let the number of productions with an arbitrary fixed nonterminal on the left also be n. Then the maximal number of different terms in a corresponding $\mathbf{S\Pi_1}$-$\mathbf{G}$ is

$$n^{n+1}$$

while the size of the grammar is

$$(n + 1) \cdot n.$$

Proposition 9 *Let φ be a minimal cut-free proof of a sequent S and $\mathcal{G}$ a $\boldsymbol{S}\Pi_1$-$\boldsymbol{G}$ covering the Herbrand term set of φ containing $n+1$ nonterminals. Let χ be a proof of S after applying Π_1-cut introduction with respect to $\mathcal{G}$ and according to Theorem 6. Then*

$$|\chi|_q + 1 \leq (|\varphi|_q)^n .$$

3.6.2 Compressing A Schema of Cut-Free Proofs with a Schematic Number of Π_1 Cuts

In the previous section, we discussed the best theoretical compression that can be achieved by introducing Π_1 cuts. Now, we want to show with the help of an example that we are able to obtain an exponential compression. For this reason, we have to consider a schema of sequents for which we introduce a schematic number of Π_1 cuts. Let $f^n a$ be a shorthand for $f \dots f a = f(\dots f(a) \dots)$ such that f occurs n times. Then we can define the schema of sequents S_n as follows:

$$P(a), \forall x.P(x) \to P(fx) \vdash P(f^n a).$$

The left-hand side of S_n gives the theory in which the predicate P is true for a and for $f \dots f a$ with an arbitrary number of f-s, especially for $f^n a$. Thus, every instance of S_n where n is a fixed natural number is a provable sequent. A cut-free proof of an instance of S_n necessarily needs n instantiations of $\forall x.P(x) \to P(fx)$, i.e.

$$P(a) \to P(fa), \dots, P\left(f^{n-1}a\right) \to P(f^n a).$$

A Herbrand sequent is then given by

$$P(a), P(a) \to P(fa), \dots, P\left(f^{n-1}a\right) \to P(f^n a) \vdash P(f^n a).$$

This is the running example of [HLRW14]. Moreover, in Example 7, we saw an instance of a variant of S_n where $n = 4$ and the whole sequent is pressed in a single formula. The sequent is proven with the help of a single Π_1 cut which can be simplified to the formula

$$\forall x.P(x) \to P\left(f^2 x\right).$$

This formula is easily provable in this context, i.e. the sequent

$$P(a), \forall x.P(x) \to P(fx) \vdash P(f^n a), \forall x.P(x) \to P\left(f^2 x\right)$$

is provable. Once proven, the cut formula allows us to make two steps at once. Just assume, we would add $\forall x.P(x) \to P\left(f^2 x\right)$ to the theory:

$$P(a), \forall x.P(x) \to P(fx), \forall x.P(x) \to P\left(f^2 x\right) \vdash P(f^n a).$$

Then we can define a smaller Herbrand sequent (for simplicity let n be an even number):

$$P(a), P(a) \to P\left(f^2 a\right), \dots, P\left(f^{n-2}a\right) \to P(f^n a) \vdash P(f^n a).$$

It only needs half the instantiations in comparison to the first Herbrand sequent. Indeed, the larger n becomes the more compression can be achieved. Let m be a fixed natural number and consider S_n for n being 2^m. Then we are able to introduce m cuts with the cut formulas:

$$\begin{array}{l} \forall x.P(x) \to P\left(f^{2^1}x\right) \\ \vdots \\ \forall x.P(x) \to P\left(f^{2^m}x\right). \end{array}$$

Let $\Gamma =_{def} \{P(a), \forall x.P(x) \to P(fx)\}$ and $\Delta =_{def} \{P\left(f^{2^m}a\right)\}$. In order to define a valid $\mathbf{G3c}^+$-proof, we need to prove the following sequents:

$$\Gamma \vdash \Delta, \forall x.P(x) \to P\left(f^2x\right), \tag{3.1}$$

$$\Gamma, \forall x.P(x) \to P\left(f^2x\right) \vdash \Delta, \forall x.P(x) \to P\left(f^4x\right), \tag{3.2}$$

$$\Gamma, \forall x.P(x) \to P\left(f^4x\right) \vdash \Delta, \forall x.P(x) \to P\left(f^8x\right), \tag{3.3}$$

$$\vdots$$

$$\Gamma, \forall x.P(x) \to P\left(f^{2^{m-1}}x\right) \vdash \Delta, \forall x.P(x) \to P\left(f^{2^m}x\right), \tag{3.4}$$

and

$$\Gamma, \forall x.P(x) \to P\left(f^{2^m}x\right) \vdash \Delta. \tag{3.5}$$

The last sequent needs only a single instantiation to be proven. All others need three instantiations: one application of the $r\!:\!\forall$-rule and two application of the $l\!:\!\forall$-rule to the formula with the highest superscript on the left (in total only two weak quantifier rules). In case of the second sequent this produces the tautology

$$\begin{array}{l} \Gamma, \forall x.P(x) \to P\left(f^2x\right), P(\alpha) \to P\left(f^2\alpha\right), P\left(f^2\alpha\right) \to P\left(f^4\alpha\right) \\ \vdash \Delta, \forall x.P(x) \to P\left(f^4x\right), P(\alpha) \to P\left(f^4\alpha\right). \end{array}$$

All together, we get $2 \cdot m + 1$ weak quantifier instantiations. By combining all sequents via the *Cut*-rule we obtain a proof of S_{2^m} with m Π_1 cuts and a quantifier complexity of $2 \cdot m + 1$. The structure of the proof is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{(3.1) \quad (3.2)}{\Gamma \vdash \Delta, \forall x.P(x) \to P\left(f^4x\right)}\,Cut \quad (3.3)}{\vdots}\,Cut \quad \dfrac{\vdots \quad (3.4)}{\Gamma \vdash \Delta, \forall x.P(x) \to P\left(f^{2^m}x\right)}\,Cut \quad (3.5)}{S_{2^m}}$$

If we want to proof S_{2^m} without cuts, we have to instantiate $\forall x.P(x) \to P(fx)$ 2^m-times to get the Herbrand sequent

$$P(a), P(a) \to P(fa), \ldots, P\left(f^{2^m-1}a\right) \to P\left(f^{2^m}a\right) \vdash P\left(f^{2^m}a\right).$$

Hence, we obtain an exponential decrease from 2^m to $2 \cdot m + 1$ by introducing m Π_1 cuts which proves the bound of Proposition 9 to be sharp. [HLRW14] presents also an improved version of Π_1-cut introduction which is able to compute these cut formulas and therefore, is able to compress the proof exponentially (not only in terms of quantifier complexity; cf. Section 7.2.2. of the mentioned paper).

CHAPTER 4

Π_2-Cut Introduction

The problem of Π_2-cut introduction can be characterized in an analogous way as the problem of Π_1-cut introduction in Chapter 3. The Sections 4.2 to 4.4 are similar to the Sections 3.1 to 3.3 and can be seen as a generalization. However, Section 4.5 shows that Π_2-cut introduction is not always solvable, i.e. in general there is no canonical solution as in the Π_1 case. In the Sections 4.6 and 4.7, we give a full characterization of the conditions that have to be fulfilled in order to find a solution and discuss how we can decide if there are solutions of a certain type, the so called balanced solutions (see Definition 43). In Section 4.8, we show that the presented methods can be extended in order to introduce cut formulas of the form $\forall\vec{x}\exists\vec{y}C$ where C is quantifier free and $\vec{x}$ and $\vec{y}$ are tuples of variables. In Section 4.9, we conclude the chapter by showing the maximal possible proof compression our method can achieve. The main definitions and statements of the chapter are taken from [LL18].

4.1 Motivation

In Section 3.5, we presented two possible applications of Π_1-cut introduction. On the one hand, Π_1-cut introduction gives us a method at hand that is able to introduce lemmas with mathematical meaning. On the other hand, it decreases the proof size making them more human readable and reducing the required space to store the proofs. Both applications can be carried over to the Π_2 case becoming even more powerful. The potential meaning of cut formulas with an alternation in the quantifiers is much higher than the one of purely universal quantifiers. The Π_2 statement for all ($\forall$) values x exists ($\exists$) a value y already defines the notion of a total function. Moreover, the compression that can be achieved relates as follows: while a single Π_2 cut can achieve an exponential compression as we prove in Section 4.9, one needs several Π_1 cuts to achieve the same compression (see Section 3.6).

Apart from this, by giving an decision algorithm for a fragment of the Π_2-cut introduction problem we developed techniques that are promising for automated induction and a more direct introduction of cut formulas. In [LP18], we show how to use the abstraction methods of Definition 40 of Section 4.7 to generalize the concept of literals. Instead of single literals, we think of all literals of a common shape. For instance, $P(ffa), P(fx), P(fb)$ might be seen as a single literal $P(f\alpha)$ where α is a variable representing the terms fa, x, and b. This common literal can be treated as normal literals or be separated again. In the context of clause tableaux methods, this may lead to the automatic introduction of cuts achieving an exponential compression. Thereby, even the efficiency of the search algorithm can be increased.

4.2 Analysis of Π_2 Cuts in Sequent Calculus

In the same way as Herbrand sequents are not sufficient to represent proofs with Π_1 cuts, the Π_1-**EHS** is not sufficient to represent proofs with Π_2 cuts. Hence, we need a new concept of extended Herbrand sequents for handling Π_2 cuts.

We proceed as in Section 3.1 and extract from a proof with a Π_2 cut an extended Herbrand sequent for Π_2 cuts. The construction of an extended Herbrand sequent for Π_2 cuts again follows the idea to store the information of all quantifier instantiations that correspond to a proof with Π_2 cuts in a single sequent. Furthermore, the sequent should be in a form such that we are able to reconstruct a proof with Π_2 cuts. Afterwards, we generalize the concept for proofs with at most a single Π_2 cut of the form $\forall x \exists y C$ where x and y are single variables and C is a quantifier free formula. In Section 4.8, we extend the method to formulas with blocks of quantifiers. Note that there is a difference to Chapter 3 where we introduced several cuts at once.

Example 14 (Cf. Example 3 of [LL18]) *Let P be a binary predicate symbol, a a constant, and f, g, h be unary functions. Let $F =_{def} P(x, fx) \vee P(x, gx)$, $G =_{def} P(a, y_1) \wedge P(hy_1, y_2)$, and $C =_{def} P(z_1, z_2)$. We are going to consider a theory in which F is true for all x. Hence, there is for every term r a partner s such that $P(r, s)$ is true where s is fr or gr. When showing that there are witnesses to satisfy G we can make use of this intuition by introducing the cut formula $\forall z_1 \exists z_2 P(z_1, z_2)$. Figure 4.1 shows a proof of*

$$S \overset{def}{=} \forall x.P(x, fx) \vee P(x, gx) \vdash \exists y_1, y_2.P(a, y_1) \wedge P(hy_1, y_2)$$

with the mentioned Π_2 cut. Without rearranging the rules, we can extract the two midsequents of the subproofs φ_l and φ_r:

$$\forall xF, F[x\backslash\alpha] \vdash P(\alpha, g\alpha), P(\alpha, f\alpha), \exists z_2.C[z_1\backslash\alpha], \exists y_1, y_2 G \text{ and}$$
$$\forall xF, \forall z_1 \exists z_2 C, P(a, \beta_1), P(h\beta_1, \beta_2) \vdash G[(y_1, y_2) \setminus (\beta_1, \beta_2)], \Pi.$$

The first midsequent corresponds to the proof that the cut formula can be proven in the given theory. The second midsequent uses the cut formula to prove the existence of

Figure 4.1: Proof with a single Π_2 cut; Example 14

$$\dfrac{\begin{array}{c}r:\forall\end{array}\dfrac{\varphi_l}{\forall xF \vdash \forall z_1 \exists z_2 C, \exists y_1, y_2 G} \qquad \begin{array}{c}l:\forall\end{array}\dfrac{\varphi_r}{\forall xF, \forall z_1 \exists z_2 C \vdash \exists y_1, y_2 G}}{\begin{array}{c}Cut\end{array}\dfrac{}{\forall xF \vdash \exists y_1, y_2 G}}$$

with

$$\varphi_l \overset{def}{=} \begin{array}{c}l:\vee\\ l:\forall\\ r:\exists\\ r:\exists\end{array}\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\dfrac{\Gamma, P(\alpha, f\alpha) \vdash P(\alpha, f\alpha), \Delta_l \qquad \Gamma, P(\alpha, g\alpha) \vdash P(\alpha, g\alpha), \Delta_r}{\forall xF, F[x\backslash\alpha] \vdash P(\alpha, g\alpha), P(\alpha, f\alpha), \exists z_2.C[z_1\backslash\alpha], \exists y_1, y_2 G}\end{array}}{\forall xF \vdash P(\alpha, g\alpha), P(\alpha, f\alpha), \exists z_2.C[z_1\backslash\alpha], \exists y_1, y_2 G}\end{array}}{\forall xF \vdash P(\alpha, f\alpha), \exists z_2.C[z_1\backslash\alpha], \exists y_1, y_2 G}}{\forall xF \vdash \exists z_2.C[z_1\backslash\alpha], \exists y_1, y_2 G},$$

$$\varphi_r \overset{def}{=} \begin{array}{c}r:\wedge\\ r:\exists\\ r:\exists\\ l:\exists\\ l:\forall\\ l:\exists\end{array}\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\dfrac{\Lambda_l, P(a, \beta_1) \vdash P(a, \beta_1), \Pi \qquad \Lambda_r, P(h\beta_1, \beta_2) \vdash P(h\beta_1, \beta_2), \Pi}{\forall xF, \forall z_1 \exists z_2 C, P(a, \beta_1), P(h\beta_1, \beta_2) \vdash G[(y_1, y_2) \setminus (\beta_1, \beta_2)], \Pi}\end{array}}{\forall xF, \forall z_1 \exists z_2 C, P(a, \beta_1), P(h\beta_1, \beta_2) \vdash \exists y_2.G[y_1\backslash\beta_1], \exists y_1, y_2 G}\end{array}}{\forall xF, \forall z_1 \exists z_2 C, P(a, \beta_1), P(h\beta_1, \beta_2) \vdash \exists y_1, y_2 G}\end{array}}{\forall xF, \forall z_1 \exists z_2 C, P(a, \beta_1), \exists z_2.C[z_1\backslash h\beta_1] \vdash \exists y_1, y_2 G}\end{array}}{\forall xF, \forall z_1 \exists z_2 C, P(a, \beta_1) \vdash \exists y_1, y_2 G}}{\forall xF, \forall z_1 \exists z_2 C, \exists z_2.C[z_1\backslash a] \vdash \exists y_1, y_2 G},$$

$$\begin{aligned}
\Gamma &\overset{def}{=} \{\forall xF\},\\
\Delta_l &\overset{def}{=} \{P(\alpha, g\alpha), \exists z_2.C[z_1\backslash\alpha], \exists y_1, y_2 G\},\\
\Delta_r &\overset{def}{=} \{P(\alpha, f\alpha), \exists z_2.C[z_1\backslash\alpha], \exists y_1, y_2 G\},\\
\Pi &\overset{def}{=} \{\exists y_2.G[y_1\backslash\beta_1], \exists y_1, y_2 G\},\\
\Lambda_l &\overset{def}{=} \{\forall xF, \forall z_1 \exists z_2 C, P(h\beta_1, \beta_2)\}, \text{ and}\\
\Lambda_r &\overset{def}{=} \{\forall xF, \forall z_1 \exists z_2 C, P(a, \beta_1)\}
\end{aligned}$$

witnesses satisfying G. The corresponding Herbrand sequents are

$$P(\alpha, f\alpha) \vee P(\alpha, g\alpha) \vdash P(\alpha, f\alpha), P(\alpha, g\alpha) \text{ and}$$
$$P(a, \beta_1), P(h\beta_1, \beta_2) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2).$$

Here, the usual structure of Π_2 cuts becomes apparent: Since both premises of the corresponding Cut-rule will eventually introduce an eigenvariable, the instantiations of the context can be separated. In the first Herbrand sequent, there is only an instantiation of F and in the second Herbrand sequent, there is only an instantiation of G. We will make use of this property and divide the context always into two parts; one interacting only with the cut formula when occurring in the left branch (in later examples and definitions also denoted with F) and one interacting with the cut formula when occurring in the right branch (in later examples and definitions also denoted with G). As in the Π_1 case (see Example 7), we extend the Herbrand sequents such that they have a shared context and apply an $l\colon\wedge$-inference, an $r\colon\vee$-inference, and an $l\colon\to$-inference:

$$l\colon\to \frac{\chi_l \qquad \chi_r}{T}$$

with

$$\chi_l \overset{def}{=} \quad r\colon\vee \frac{P(\alpha, f\alpha) \vee P(\alpha, g\alpha) \vdash P(\alpha, f\alpha), P(\alpha, g\alpha), P(a, \beta_1) \wedge P(h\beta_1, \beta_2)}{P(\alpha, f\alpha) \vee P(\alpha, g\alpha) \vdash P(\alpha, f\alpha) \vee P(\alpha, g\alpha), P(a, \beta_1) \wedge P(h\beta_1, \beta_2)} \ ,$$

$$\chi_r \overset{def}{=} \quad l\colon\wedge \frac{P(\alpha, f\alpha) \vee P(\alpha, g\alpha), P(a, \beta_1), P(h\beta_1, \beta_2) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2)}{P(\alpha, f\alpha) \vee P(\alpha, g\alpha), P(a, \beta_1) \wedge P(h\beta_1, \beta_2) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2)} \ ,$$

and

$$T \overset{def}{=} \begin{array}{c} P(\alpha, f\alpha) \vee P(\alpha, g\alpha), \\ (P(\alpha, f\alpha) \vee P(\alpha, g\alpha)) \to (P(a, \beta_1) \wedge P(h\beta_1, \beta_2)) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2). \end{array}$$

By construction, T is a tautology and thus, an extended Herbrand sequent for Π_2 cuts.

In Example 14, we built an extended Herbrand sequent for Π_2 cuts for a proof with a single Π_2 cut. In contrast to the Π_1 case where we extended this concept to several cuts, we consider here only a single cut. The structure of the extended Herbrand sequent for Π_2 cuts is the same as in the Π_1 case, i.e. it consists of two parts: The first part is a set of instantiations of the context, but now split into two formulas (once F and once G). The second is an implication representing the information of the cut formula. The major difference to the Π_1 case, appears in the second part where a conjunction is not only implied by a formula, but by a disjunction of formulas. This is due to

multiple instantiations of the cut formula in the left branch. While in the Π_1 case the cut formula in the left branch was only instantiated with a new eigenvariable, we see now several instantiations of the second quantifier (in Example 14, the second quantifier in the left branch was instantiated with $f\alpha$ and $g\alpha$). Moreover, the increased number of eigenvariables with different occurrences require a more complex treatment of the variable conditions.

Definition 28 (Extended Herbrand Sequent for Π_2 Cuts; See Definition 5 of [LL18]) *Let T_1 be the set of tuples of terms $\{\vec{r}_1, \ldots, \vec{r}_l\}$ and T_2 be the set of tuples of terms $\{\vec{s}_1, \ldots, \vec{s}_m\}$ where all tuples of a single set have the same length, let C be a quantifier-free formula, let $\alpha, \beta_1, \ldots, \beta_p$ be variables, and let r_i, s_j for $1 \leq i \leq q, 1 \leq j \leq p$ be terms such that*

$$\begin{aligned} \mathcal{V}(r_i) &\subseteq \{\alpha\} \quad \textit{for all } i \textit{ and} \\ \mathcal{V}(s_j) &\subseteq \{\beta_1, \ldots, \beta_{j-1}\} \quad \textit{for all } j > 1 \textit{ and} \\ \mathcal{V}(s_1) &= \emptyset. \end{aligned}$$

Then the sequent

$$\mathcal{H} \stackrel{def}{=} F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}, \bigvee_{i=1}^{q} C[(x,y) \setminus (\alpha, r_i)] \to \bigwedge_{j=1}^{p} C[(x,y) \setminus (s_j, \beta_j)] \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}$$

is called an extended Herbrand sequent for Π_2 cuts *(shorthand: Π_2-**EHS**) of $\forall\vec{x}F \vdash \exists\vec{y}G$ if $\mathcal{H}$ is a tautology.*

*The complexity of a Π_2-**EHS** is defined as $|\mathcal{H}|_{\Pi_2} =_{def} \sharp T_1 + \sharp T_2 + p + q$.*

The definition of a Π_2-**EHS** is very similar to the definition of a Π_1-**EHS**. Obviously, the cut formula becomes more complicated. Moreover, we consider a version with a single cut formula C. This is due to the much more complicated nesting of the eigenvariables and terms that might contain eigenvariables in a proof with more than a single Π_2 cut. Another interesting change is that we consider a different end sequent. Of course, $\forall\vec{x}F \vdash \exists\vec{y}G$ is more general than $\forall\vec{x}F \vdash$. But the main reason for this extension is that, from now on, we will assume that the formulas F and G are constructed such that

$$F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}, \bigvee_{i=1}^{q} C[(x,y) \setminus (\alpha, r_i)] \to \bigwedge_{j=1}^{p} C[(x,y) \setminus (s_j, \beta_j)] \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}$$

being a tautology implies the provability of the sequents

$$F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l} \vdash \bigvee_{i=1}^{q} C[(x,y) \setminus (\alpha, r_i)]$$

and

$$\bigwedge_{j=1}^{p} C[(x,y) \setminus (s_j, \beta_j)] \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}.$$

This can be assumed because the formulas F and G can always be extended such that this property holds. In this form, the reasoning becomes much simpler (depending on the branch, we can ignore one of the formulas $\forall \vec{x} F$ or $\exists \vec{y} G$ and their instantiations) and all statements and definitions also hold for the original case.

Again, the Π_2-**EHS** allows us to construct a proof with cut, this time a proof with a single Π_2 cut. We consider the Π_2-**EHS** extracted in Example 14.

Example 15 (Cf. Example 3 of [LL18]) *Let P, a, f, g, h, F, G, and T be as in Example 14. Then T is a Π_2-**EHS** with*

$$q = 2, \;\; r_1 = f\alpha, \;\; r_2 = g\alpha,$$
$$p = 2, \;\; s_1 = a, \;\; s_2 = h\beta_1.$$

Now, we want to invert the process described in Example 14 and construct a proof with a single Π_2 cut that corresponds to T. From T (which is tautological), we obtain the tautological sequents

$$P(\alpha, f\alpha) \vee P(\alpha, g\alpha) \vdash P(\alpha, f\alpha) \vee P(\alpha, g\alpha)$$

and

$$P(a, \beta_1) \wedge P(h\beta_1, \beta_2) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2).$$

This implies that we are able to prove

$$\forall x F \vdash \forall z_1 \exists z_2 C$$
$$\forall z_1 \exists z_2 C \vdash \exists y_1, y_2 G$$

with the formula $C = P(z_1, z_2)$ by introducing the quantifiers in a suitable order. Furthermore, the addition of formulas to either side does not destroy the provability (in sequent calculi with structural rules, this can be done by weakening*). Hence, the following sequents are also provable:*

$$\forall x F \vdash \forall z_1 \exists z_2 C, \exists y_1, y_2 G$$
$$\forall x F, \forall z_1 \exists z_2 C \vdash \exists y_1, y_2 G.$$

This can be arranged to a proof with a single Π_2 cut which is depicted in Figure 4.2. In general, the obtained proof is not necessarily equal to the proof of the extraction procedure as in Example 14. Here, the proofs of Figure 4.1 and 4.2 are actually identical.

*Other than in Chapter 3 (more precisely Example 8), we do not consider several cuts at once. Hence, the described procedure corresponds exactly to the general case of introducing a Π_2 cut via a Π_2-**EHS**: We construct propositional proofs for both premises of the $l\colon\!\to$-rule applied to the Π_2-**EHS**, extend them downwards by introducing the necessary quantifiers, and finally apply a Cut-rule.*

Figure 4.2: Reconstructed proof with a single Π_2 cut; Example 15

$$\dfrac{r:\forall\dfrac{\varphi_l}{\forall xF\vdash\forall z_1\exists z_2C,\exists y_1,y_2G}\qquad l:\forall\dfrac{\varphi_r}{\forall xF,\forall z_1\exists z_2C\vdash\exists y_1,y_2G}}{\forall xF\vdash\exists y_1,y_2G}\,Cut$$

where

$$\varphi_l \stackrel{def}{=} \dfrac{\dfrac{\dfrac{\dfrac{\Gamma,P(\alpha,f\alpha)\vdash P(\alpha,f\alpha),\Delta_l\qquad\Gamma,P(\alpha,g\alpha)\vdash P(\alpha,g\alpha),\Delta_r}{\forall xF,F[x\backslash\alpha]\vdash P(\alpha,g\alpha),P(\alpha,f\alpha),\exists z_2C[z_1\backslash\alpha],\exists y_1,y_2G}\,l:\vee}{\forall xF\vdash P(\alpha,g\alpha),P(\alpha,f\alpha),\exists z_2C[z_1\backslash\alpha],\exists y_1,y_2G}\,l:\forall}{\forall xF\vdash P(\alpha,f\alpha),\exists z_2C[z_1\backslash\alpha],\exists y_1,y_2G}\,r:\exists}{\forall xF\vdash\exists z_2C[z_1\backslash\alpha],\exists y_1,y_2G}\,r:\exists$$

and

$$\varphi_r \stackrel{def}{=} \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Lambda_l,P(a,\beta_1)\vdash P(a,\beta_1),\Pi\qquad\Lambda_r,P(h\beta_1,\beta_2)\vdash P(h\beta_1,\beta_2),\Pi}{\forall xF,\forall z_1\exists z_2C,P(a,\beta_1),P(h\beta_1,\beta_2)\vdash G[(y_1,y_2)\setminus(\beta_1,\beta_2)],\Pi}\,r:\wedge}{\forall xF,\forall z_1\exists z_2C,P(a,\beta_1),P(h\beta_1,\beta_2)\vdash\exists y_2G[y_1\backslash\beta_1],\exists y_1,y_2G}\,r:\exists}{\forall xF,\forall z_1\exists z_2C,P(a,\beta_1),P(h\beta_1,\beta_2)\vdash\exists y_1,y_2G}\,r:\exists}{\forall xF,\forall z_1\exists z_2C,P(a,\beta_1),\exists z_2C[z_1\backslash h\beta_1]\vdash\exists y_1,y_2G}\,l:\exists}{\forall xF,\forall z_1\exists z_2C,P(a,\beta_1)\vdash\exists y_1,y_2G}\,l:\forall}{\forall xF,\forall z_1\exists z_2C,\exists z_2C[z_1\backslash a]\vdash\exists y_1,y_2G}\,l:\exists$$

with

$$\begin{aligned}
\Gamma &\stackrel{def}{=} \{\forall xF\},\\
\Delta_l &\stackrel{def}{=} \{P(\alpha,g\alpha),\exists z_2C[z_1\backslash\alpha],\exists y_1,y_2G\},\\
\Delta_r &\stackrel{def}{=} \{P(\alpha,f\alpha),\exists z_2C[z_1\backslash\alpha],\exists y_1,y_2G\},\\
\Pi &\stackrel{def}{=} \{\exists y_2G[y_1\backslash\beta_1],\exists y_1,y_2G\},\\
\Lambda_l &\stackrel{def}{=} \{\forall xF,\forall z_1\exists z_2C,P(h\beta_1,\beta_2)\},\text{ and}\\
\Lambda_r &\stackrel{def}{=} \{\forall xF,\forall z_1\exists z_2C,P(a,\beta_1)\}.
\end{aligned}$$

Figure 4.3: Propositional proof based on the Π_2-**EHS**; Proof of Theorem 10

$$l:\rightarrow \dfrac{\begin{array}{c} r:\vee \dfrac{F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l} \vdash C[(x,y)\setminus(\alpha,r_i)]_{i=1}^{q}, G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}}{\vdots} \\ r:\vee \dfrac{\vdots}{F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l} \vdash \bigvee\limits_{i=1}^{q} C[(x,y)\setminus(\alpha,r_i)], G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}} \end{array} \qquad \chi}{F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}, \bigvee\limits_{i=1}^{q} C[(x,y)\setminus(\alpha,r_i)] \rightarrow \bigwedge\limits_{j=1}^{p} C[(x,y)\setminus(s_j,\beta_j)] \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}}$$

with

$$\chi \overset{def}{=} \begin{array}{c} l:\wedge \dfrac{F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}, C[(x,y)\setminus(s_j,\beta_j)]_{j=1}^{p} \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}}{\vdots} \\ l:\wedge \dfrac{\vdots}{F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}, \bigwedge\limits_{j=1}^{p} C[(x,y)\setminus(s_j,\beta_j)] \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}} \end{array}$$

We obtain a result analogous to that in Π_1-cut introduction. The Π_2-**EHS** corresponds to a proof with a single Π_2 cut. More precisely: We are able to construct a proof φ with a Π_2 cut if and only if there is a Π_2-**EHS** such that its complexity as defined in Definition 28 is the quantifier complexity (see Definition 10) of φ. Thus, Π_2-cut introduction can be formulated in the following way: construct a Π_2-**EHS** for a given cut-free proof.

Theorem 10 (See Theorem 2 of [LL18]) *The sequent $\forall\vec{x}F \vdash \exists\vec{y}G$ has a proof φ with a single Π_2 cut $\forall x\exists y C$ such that $|\varphi|_q = n$ iff it has a Π_2-**EHS** $\mathcal{H}$ of the form*

$$F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}, \bigvee_{i=1}^{q} C[(x,y)\setminus(\alpha,r_i)] \rightarrow \bigwedge_{j=1}^{p} C[(x,y)\setminus(s_j,\beta_j)] \vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}$$

with $|\mathcal{H}|_{\Pi_2} = n$.

Proof:
Concerning the left-to-right direction, the proof follows the steps of Example 14. We pass through the proof φ and read off the instances of quantified formulas (of both the end-formula and the cut). We obtain a Π_2-**EHS** $\mathcal{H}$ with $|\mathcal{H}|_{\Pi_2} \leq |\varphi|_q$ (which can be padded with dummy instances if necessary in order to obtain $|\mathcal{H}|_{\Pi_2} = |\varphi|_q$).

Concerning the right-to-left direction, we conclude from the validity of the proof depicted in Figure 4.3 that the proof of Figure 4.4 is valid. The provability of

$$F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l} \vdash C[(x,y)\setminus(\alpha,r_i)]_{i=1}^{q}, G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m}$$

Figure 4.4: General shape of a constructed proof with Π_2 cut; Proof of Theorem 10

$$\dfrac{r:\forall\dfrac{\psi_l}{\forall\vec{x}F\vdash\forall x\exists yC,\exists\vec{y}G}\qquad l:\forall\dfrac{\psi_r}{\forall\vec{x}F,\forall x\exists yC\vdash\exists\vec{y}G}}{Cut\dfrac{}{\forall\vec{x}F\vdash\exists\vec{y}G}}$$

with

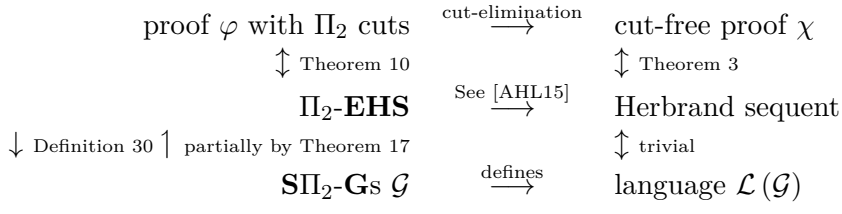
$$\psi_l \overset{def}{=} \cfrac{\cfrac{\cfrac{\vdots}{\Gamma,F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}\vdash C[(x,y)\setminus(\alpha,r_i)]_{i=1}^{q},\exists y.C[x\backslash\alpha],\exists\vec{y}G}\,l:\forall}{\cfrac{\vdots}{\forall\vec{x}F\vdash C[(x,y)\setminus(\alpha,r_i)]_{i=1}^{q},\exists y.C[x\backslash\alpha],\exists\vec{y}G}\,l:\forall}}{\cfrac{\cfrac{\vdots}{\forall\vec{x}F\vdash C[(x,y)\setminus(\alpha,r_1)],\exists y.C[x\backslash\alpha],\exists\vec{y}G}\,r:\exists}{\forall\vec{x}F\vdash\exists y.C[x\backslash\alpha],\exists\vec{y}G}\,r:\exists}$$

and

$$\psi_r \overset{def}{=} \cfrac{\cfrac{\cfrac{\cfrac{\vdots}{\forall\vec{x}F,\forall x\exists yC,C[(x,y)\setminus(s_j,\beta_j)]_{j=1}^{p}\vdash G[\vec{y}\backslash\vec{s}_j]_{j=1}^{m},\Delta}\,r:\exists}{\cfrac{\vdots}{\forall\vec{x}F,\forall x\exists yC,C[(x,y)\setminus(s_j,\beta_j)]_{j=1}^{p}\vdash\exists\vec{y}G\quad(*)}\,r:\exists}}{\cfrac{\cfrac{\vdots}{\forall\vec{x}F,\forall x\exists yC,C[(x,y)\setminus(s_1,\beta_1)]\vdash\exists\vec{y}G\quad(*)}\,l:\forall}{\forall\vec{x}F,\forall x\exists yC,\exists y.C[x\backslash s_1]\vdash\exists\vec{y}G}\,l:\exists}\quad .$$

Γ is the set consisting of $\forall\vec{x}F$ and all of its partly instantiated versions appearing during the derivation of $F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l}$.

Figure 4.5: Proof-theoretic setting of Π_2-cut introduction



is given by the extended Herbrand sequent being a tautology and it implies the provability of

$$\Gamma, F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l} \vdash C[(x,y) \setminus (\alpha, r_i)]_{i=1}^{q}, \exists y.C[x\backslash\alpha], \exists\vec{y}G.$$

Note that we defined F and G such that

$$F[\vec{x}\backslash\vec{r}_i]_{i=1}^{l} \vdash C[(x,y) \setminus (\alpha, r_i)]_{i=1}^{q}$$

is provable. The reasoning for the right branch is analogous. In the proofs, the dots represent multiple applications of $r\colon\vee$, $l\colon\wedge$, $l\colon\forall$, $r\colon\exists$, or $l\colon\exists$. In the particular case between the sequents marked with $(*)$, the dots denote an alternating application of $l\colon\forall$ and $l\colon\exists$ ($p-1$ times).

Given that every term of $\mathcal{H}$ is used exactly once in a quantifier rule, the quantifier complexity is equal to $|\mathcal{H}|_{\Pi_2}$. □

4.3 Schematic Π_2 Grammars

As in the Π_1 case, there seems to be a connection between formal grammars (see Section 2.6) and proofs with Π_2 cuts. In [AHL15], they elaborate an association between proofs with Π_2 cuts only and context-free tree grammars such that the grammars yield Herbrand sequents for the proofs. This leads to the question whether these tree grammars correspond to proofs with Π_2 cuts as **S**Π_1-**G**s correspond to proofs with Π_1 cuts. For this reason, we introduce the concept of *schematic* Π_2 *grammars*. In contrast to the Π_1 case, we consider again only a version for a single cut.

Again, we want to show the validity of all relations as in Figure 3.2, but for the Π_2 case. Unfortunately, there is no canonical solutions for Π_2-cut introduction as we will see in the course of this chapter. For this reason, there are some gaps in the proof-theoretic setting depicted in Figure 4.5.

The right side of Figure 4.5 is equivalent to the right side of Figure 3.2. It shows the relations among cut-free proofs, Herbrand sequents, and the Herbrand term set considered as the language of a grammar. If we only consider the information of the quantifier

inferences in χ, they are interchangeable. On the other side, we see the relations among proofs with Π_2 cuts. While the quantifier inferences of φ can be stored in a Π_2-**EHS** and a proof with Π_2 cuts corresponding to these inferences can be constructed from the Π_2-**EHS**, the situation changes when we also consider schematic Π_2 grammars (here: **S**Π_2-**G**s). We can still extract a schematic Π_2 grammar for an Π_2-**EHS**, but the converse does not hold (see Section 4.5). In Section 4.7, we will show the desired property for a fragment: If there is *balanced solution*, we will find a Π_2-**EHS** (see Theorem 17).

Altogether, we can perform cut-elimination on proofs with Π_2 cuts via schematic Π_2 grammars, but Π_2-cut introduction via schematic Π_2 grammars is only possible for a fragment. The procedure itself is as in the Π_1 case: First we translate the Herbrand sequent of a cut-free proof into a set of terms for which we search a schematic Π_2 grammar. Afterwards, we compute a Π_2-**EHS** that corresponds to the grammar and construct a proof with Π_2 cuts on the basis of the Π_2-**EHS**.

In comparison to the **S**Π_1-**G**, the schematic Π_2 grammar contains additional conditions that are either due to the more complex setting or to the new representation of the end sequent in consideration. As already mentioned in the previous section, we assume that the end sequent of the cut-free proof is split into two parts: $\forall \vec{x} F \vdash \exists \vec{y} G$. For this reason, the starting symbol of the schematic Π_2 grammar maps on two different terms, one representing F and one representing G. This becomes more apparent in Definition 30. Moreover, we distinguish between two types of eigenvariables that occur within a proof with a single Π_2 cut. The eigenvariable α corresponds to the universal quantifier in the cut formula and the eigenvariables $\beta_1, \ldots, \beta_p$ correspond to the existential quantifier. Thus, there are also two types of nonterminals apart from the starting symbol. For simplicity, we denote them by the same lower-case Greek letters. The conditions on the corresponding production rules are due to the eigenvariable conditions.

Definition 29 also introduces the $\exists$-*multiplicity* and the $\forall$-*multiplicity*. Both names are chosen to stress the connection of schematic Π_2 grammars to Π_2-**EHS**s. The $\exists$-multiplicity corresponds to the number of weak quantifier rules applied to the existential quantifier in the potential cut formula. The $\forall$-multiplicity corresponds to the number of weak quantifier rules applied to the universal quantifier in the potential cut formula.

Definition 29 (Schematic Π_2 Grammar; See Definition 9 of [LL18])
Let $t_1, \ldots, t_l$ and $t'_1, \ldots, t'_m$ be terms, let $\alpha, \beta_1, \ldots, \beta_p$ be variables, and let $r_1, \ldots, r_q$ and $s_1, \ldots, s_p$ be terms such that

$$\begin{aligned}
&\mathcal{V}(r_i) \subseteq \{\alpha\} \text{ for all } i,\\
&\mathcal{V}(s_j) \subseteq \{\beta_1, \ldots, \beta_{j-1}\} \text{ for all } j > 1,\\
&\mathcal{V}(s_1) = \emptyset,\\
&\mathcal{V}(t_i) \subseteq \{\alpha\} \text{ for all } i, \text{ and}\\
&\mathcal{V}\left(t'_j\right) \subseteq \{\beta_1, \ldots, \beta_p\} \text{ for all } j.
\end{aligned}$$

Then we call the totally rigid acyclic tree grammar $\mathcal{G} = \langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ *with the set of nonterminals* $N = \{\tau, \alpha, \beta_1, \ldots, \beta_p\}$ *and the productions*

$$\begin{aligned} \mathrm{Pr} \stackrel{def}{=} \quad & \{\tau \to t_i \mid i \in \mathbb{N}_l\} \cup \{\tau \to t'_j \mid j \in \mathbb{N}_m\} \cup \\ & \{\alpha \to s_j \mid j \in \mathbb{N}_p\} \cup \{\beta_j \to r_i s_j \mid j \in \mathbb{N}_p, i \in \mathbb{N}_q\} \end{aligned}$$

(where rs *stands for* $r[\alpha \backslash s]$ *with* r *being a term (possibly) containing the variable* α*)* a schematic Π_2 grammar *(shorthand:* $\mathbf{S\Pi_2\text{-}G}$*). We call* q *the* $\exists$-multiplicity *and* p *the* $\forall$-multiplicity*.*

As in the Π_1 case, we define a corresponding $\mathbf{S\Pi_2\text{-}G}$ for each Π_2-**EHS** with end sequent $\forall \vec{x} F \vdash \exists \vec{y} G$. Since we want to use an ordinary term grammar without tuples, we generate function symbols h_F, h_G where h_F is of the arity of the length of $\vec{x}$ and h_G is of the arity of the length of $\vec{y}$. So every term tuple $\vec{r}_i$ occurring in $F[\vec{x} \backslash \vec{r}_i]_{i=1}^{l}$ is represented by $h_F(\vec{r}_i)$ and every term tuple $\vec{s}_j$ occurring in $G[\vec{y} \backslash \vec{s}_j]_{j=1}^{m}$ is represented by $h_G(\vec{s}_j)$. Moreover, the two different function symbols allow us to distinguish tuples of terms belonging either to F or G even if the two formulas have the same number of free variables.

Definition 30 (Schematic Π_2 Grammars of Π_2-EHSs; See Definition 10 of [LL18]) *Let* $\mathcal{H}$ *be a* Π_2-***EHS*** *as in Definition 28. We define* $\mathcal{G}(\mathcal{H}) = \langle \tau, N, \Sigma, \mathrm{Pr} \rangle$*, the* schematic Π_2 grammar corresponding to $\mathcal{H}$*, where* $N = \{\tau, \alpha, \beta_1, \ldots, \beta_p\}$ *and the production rules* Pr *as well as the variable occurrences are as in Definition 29, except for the start symbol* τ *where we have*

$$\tau \to h_F \vec{r}_1 \mid \ldots \mid h_F \vec{r}_l \mid h_G \vec{s}_1 \mid \ldots \mid h_G \vec{s}_m.$$

We call the production rules $\tau \to h_F \vec{r}_1 \mid \ldots \mid h_F \vec{r}_l$ F-productions *and the production rules* $\tau \to h_G \vec{s}_1 \mid \ldots \mid h_G \vec{s}_m$ G-productions*.*

At this point, it becomes apparent why we have chosen the form $\forall x F \vdash \exists y G$ as end sequent. In an $\mathbf{S\Pi_2\text{-}G}$ we have terms depending on α and terms depending on some β_i with $i \in \mathbb{N}_p$. These terms correspond to the function symbols h_F and h_G, i.e. we implicitly ask for formulas that can be separated within one sequent (by a comma on the right side, a comma on the left side, or the sequent symbol). This separated formulas depend either on α or on some β_i with $i \in \mathbb{N}_p$. Hence, there are no atoms that depend on both, α and β_i for $i \in \mathbb{N}_p$.

Example 16 *The* $\mathbf{S\Pi_2\text{-}G}$ $\mathcal{G}(S) = \langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ *corresponding to* T *of Example 14 is defined as follows:*

$$N \stackrel{def}{=} \quad \{\tau, \alpha, \beta_1, \beta_2\}$$

$$\Pr \stackrel{def}{=} \begin{array}{l} \{\ \tau \to h_F\alpha \mid h_G\beta_1\beta_2, \\ \quad \alpha \to a \mid h\beta_1, \\ \quad \beta_2 \to fh\beta_1 \mid gh\beta_1, \\ \quad \beta_1 \to fa \mid ga \qquad \}. \end{array}$$

Even though, α, β_1, and β_2 denote the eigenvariables in the proof we use the same symbols to represent the nonterminals. If we would insist on technical purity, we would have to introduce a new symbol for each nonterminal. When thinking of the correspondence to proofs with Π_2 cuts, we can intuitively read the production rules in the following way: Once proven for α and β_1, we can exchange α and β_1 with an arbitrary term and we need the terms a and $h\beta_1$ and fa, ga, respectively. Moreover, once proven for β_2, we can exchange β_2 with an arbitrary term. Now, the question arises whether we need the terms $fh\beta_1, gh\beta_1$ or the terms $fhfa, fhga, ghfa, ghga$. For this reason, we compare the single instantiation of the cut where β_2 is introduced the first time in φ (see Figure 4.1), i.e. $P(h\beta_1, \beta_2)$ in φ_r, and compare it with the set of instantiations of the cut formula in the other branch φ_l, i.e. $P(\alpha, f\alpha)$ and $P(\alpha, g\alpha)$. Hence, β_2 has to be once $f\alpha$ and once $g\alpha$ while α has to be mapped on $h\beta_1$. Thus, we need the terms $fh\beta_1, gh\beta_1$. The $\exists$-multiplicity and the $\forall$-multiplicity is 2.

4.4 Schematic Extended Herbrand Sequents for Π_2 Cuts

As in the Π_1 case (see Section 3.3, we have a correspondence between proofs with a single Π_2 cut and the Π_2-**EHS** and we can extract from a Π_2-**EHS** a **S**Π_2-**G** whose language covers the term set of a Herbrand sequent. Moreover, the **S**Π_2-**G** does not contain any information of the cut formula. If we were able to find a cut formula for each such grammar, we could introduce Π_2 cuts by computing **S**Π_2-**G**s. For this reason, we will again abstract the extended Herbrand sequent.

While Herbrand sequents represent cut-free proofs, Π_2-**EHS** represent proofs with Π_2 cuts. In order to introduce (yet unknown) cut-formulas we consider the Herbrand sequent of a cut-free proof and specify the Herbrand term set by a **S**Π_2-**G**. The unknown cut formula is represented by a second-order variable X. Other than in the Π_1 case, we only consider a single second-order variable which depends on two arguments. Moreover, we have two types of eigenvariables, one (α) introduced for a universal quantifier and one ($\beta_1, \ldots, \beta_p$) introduced for an existential quantifier. Another difference to the Π_1-**SEHS** is the disjunction in the premise of the implication which complicates its solvability (see the α-problem and β-problem in Section 4.6).

Definition 31 (Schematic Extended Herbrand Sequent for Π_2 Cuts; See Definition 12 and 13 of [LL18]) *Let S be the provable sequent $\forall\vec{x}F \vdash \exists\vec{y}G$ and $[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{l} \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=l+1}^{m}$ be a Herbrand sequent for S. Let $\mathcal{G} = \langle \tau, N, \Sigma, \Pr \rangle$ be an **S**Π_2-**G** with*

the fresh functions h_F, h_G *where* $\boldsymbol{a}(h_F) = \boldsymbol{l}(\vec{r}_i)$ *and* $\boldsymbol{a}(h_G) = \boldsymbol{l}(\vec{s}_j)$*, with* $N =_{def} \{\tau, \alpha, \beta_1 \ldots, \beta_p\}$*, and with*

$$\begin{aligned} \Pr \overset{def}{=} \quad & \{\tau \to h_F \vec{r}_i \mid i \in \mathbb{N}_a\} \cup \{\tau \to h_G \vec{s}_j \mid j \in \mathbb{N}_b\} \cup \\ & \{\alpha \to s_j \mid j \in \mathbb{N}_p\} \cup \{\beta_j \to r_i s_j \mid j \in \mathbb{N}_p, i \in \mathbb{N}_q\} \end{aligned}$$

such that the language of $\mathcal{G}$ *covers the Herbrand term set, i.e.*

$$\mathcal{L}(\mathcal{G}) \supseteq \{h_F \vec{r}_i \mid i \in \mathbb{N}_l\} \cup \{h_G \vec{s}_j \mid j \in \mathbb{N}_m\}.$$

Then

$$\mathcal{S}(S) \overset{def}{=} [F[\vec{x} \backslash \vec{r}_i]]_{i=1}^{a}, \bigvee_{k=1}^{q} X \alpha r_k \to \bigwedge_{k=1}^{p} X s_k \beta_k \vdash [G[\vec{y} \backslash \vec{s}_j]]_{j=1}^{b}$$

(where X *is a two-placed predicate variable) is called a* schematic extended Herbrand sequent for Π_2 cuts *(shorthand:* Π_2-**SEHS***) corresponding to* $\mathcal{G}$*.*

A solution *of a* Π_2-***SEHS*** $\mathcal{S}(S)$ *is a substitution* $\sigma = [X \backslash \lambda\alpha\beta.C]$ *such that* $\mathcal{F}(C) \subseteq \{\alpha, \beta\}$ *and* $\mathcal{S}(S)\sigma$ *is a tautology.*

Furthermore, we call

$$[F[\vec{x} \backslash \vec{r}_i]]_{i=1}^{a} \vdash [G[\vec{y} \backslash \vec{s}_j]]_{j=1}^{b}$$

the reduced representation *of* $\mathcal{S}(S)$*.*

We will often say a formula C is a solution of a Π_2-**SEHS** meaning that the corresponding substitution $[X \backslash \lambda\alpha\beta.C]$ (or sometimes $[X \backslash \lambda xy.C]$) is a solution.

The Π_2-**SEHS** is an abstraction of a Π_2-**EHS** as well as the Π_2-**EHS** is a Π_2-**SEHS** $\mathcal{S}(S)$ for which a solution has been found, i.e. a substitution $\sigma = [X \backslash \lambda\alpha\beta.G]$ such that $\mathcal{S}(S)\sigma$ is a tautology. Further explanation can be found in Example 17.

Note that we did not require

$$\mathcal{L}(\mathcal{G}) = \{h_F \vec{r}_i \mid i \in \mathbb{N}_l\} \cup \{h_G \vec{s}_j \mid j \in \mathbb{N}_m\};$$

indeed, if we generate a proper superset of the Herbrand term set, we still obtain a Herbrand sequent of S (but not a minimal one). Generating supersets can be beneficial to the construction of cut-formulas. A solution of a Π_2-**SEHS** gives us a cut formula for a proof with a Π_2 cut.

Example 17 (See Example 5 of [LL18]) *In Example 14, we defined the* Π_2-***EHS*** T*:*

$$P(\alpha, f\alpha) \vee P(\alpha, g\alpha),$$
$$(P(\alpha, f\alpha) \vee P(\alpha, g\alpha)) \to (P(a, \beta_1) \wedge P(h\beta_1, \beta_2)) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2)$$

for the end sequent

$$S \stackrel{def}{=} \forall x.P\left(x, fx\right) \vee P\left(x, gx\right) \vdash \exists y_1, y_2.P\left(a, y_1\right) \wedge P\left(hy_1, y_2\right).$$

Later, we generated the corresponding $\boldsymbol{S\Pi_2}$*-**G*** $\mathcal{G} = \langle \tau, N, \Sigma, \Pr \rangle$ *(see Example 16) where*

$$\begin{array}{rl} N \stackrel{def}{=} & \{\tau, \alpha, \beta_1, \beta_2\} \\ \\ \Pr \stackrel{def}{=} & \begin{array}{l} \{\ \tau \to h_F\alpha \mid h_G\beta_1\beta_2, \\ \quad \alpha \to a \mid h\beta_1, \\ \quad \beta_2 \to fh\beta_1 \mid gh\beta_1, \\ \quad \beta_1 \to fa \mid ga \qquad \}. \end{array} \end{array}$$

Then T *witnesses the solvability of the* Π_2*-**SEHS*** $\mathcal{S}\left(S\right)$ *corresponding to* $\mathcal{G}$*, defined as*

$$\begin{array}{c} P\left(\alpha, f\alpha\right) \vee P\left(\alpha, g\alpha\right), \\ \left(X\alpha f\alpha \vee X\alpha g\alpha\right) \to \left(Xa\beta_1 \wedge Xh\beta_1\beta_2\right) \ \vdash P\left(a, \beta_1\right) \wedge P\left(h\beta_1, \beta_2\right), \end{array}$$

for which the solution $\sigma =_{def} [X \backslash \lambda\alpha\beta.P\left(\alpha, \beta\right)]$ *has been found. When applying* σ *to* $\mathcal{S}\left(S\right)$*, we get the tautological sequent* T*. Hence, we can construct a proof with a single* Π_2 *cut.*

4.5 The Solution Problem

While a Π_1-**SEHS** is always solvable (see Section 3.4), Π_2-**SEHS** do not necessarily have solutions. In this section, we will present two counter examples, the first of which seems to be avoidable by redefining $\mathbf{S\Pi_2}$-**G**s. The second example shows that there is no easy solution of this problem.

The examples are based on very simple proofs and differ only in a constant: Assume two binary predicates with the corresponding symbols P and Q such that the first is always true when the second argument is the unary function f applied to the first argument, i.e. $P\left(x, fx\right)$. The second predicate is always true when the second argument is the unary function g applied to the first argument, i.e. $Q\left(x, gx\right)$. The conditions on P and Q only differ in the function applied to x. To express this in sequent calculus, one way is to add the formula $\forall x.P\left(x, fx\right) \wedge Q\left(x, gx\right)$ to the context on the left side of the sequent.

The first proof shows the existence of witnesses such that P holds when the first argument is a and that at the same time Q holds when the first argument is as well a. This can be expressed as $\exists x, y.P\left(a, x\right) \wedge Q\left(a, y\right)$. The statement is obviously provable and the witnesses are fa and ga. A cut-free proof is depicted in Figure 4.6a.

The second proof is the same as the first, except that the constants for P and Q are different. We want to show the existence of witnesses such that P holds when the first

Figure 4.6: Simple cut-free proofs used to show the non-existence of a canonical solution of Π_2-cut introduction problem

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Ax\ \dfrac{}{\Gamma, P(a,fa), Q(a,ga) \vdash P(a,fa), \Delta}}{r:\wedge}\quad Ax\ \dfrac{}{\Gamma, P(a,fa), Q(a,ga) \vdash Q(a,ga), \Delta}}{\Gamma, P(a,fa), Q(a,ga) \vdash P(a,fa) \wedge Q(a,ga), \Delta}}{l:\wedge}}{\Gamma, P(a,fa) \wedge Q(a,ga) \vdash P(a,fa) \wedge Q(a,ga), \Delta}}{r:\exists}}{\Gamma, P(a,fa) \wedge Q(a,ga) \vdash \Delta}}{r:\exists}}{\Gamma, P(a,fa) \wedge Q(a,ga) \vdash \exists x, y.P(a,x) \wedge Q(a,y)}}{l:\forall}}{\forall x.P(x,fx) \wedge Q(x,gx) \vdash \exists x, y.P(a,x) \wedge Q(a,y)}$$

where

$$\Gamma \stackrel{def}{=} \forall x.P(x,fx) \wedge Q(x,gx) \text{ and}$$
$$\Delta \stackrel{def}{=} \{\exists x, y.P(a,x) \wedge Q(a,y), \exists y.P(a,fa) \wedge Q(a,y)\}$$

(a) Same constants

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Ax\ \dfrac{}{\Gamma_l, P(a,fa) \vdash P(a,fa), \Delta}}{r:\wedge}\quad Ax\ \dfrac{}{\Gamma_r, Q(b,gb) \vdash Q(b,gb), \Delta}}{\Gamma, P(a,fa), Q(a,ga), P(b,fb), Q(b,gb) \vdash P(a,fa) \wedge Q(b,gb), \Delta}}{l:\wedge}}{\Gamma, P(a,fa) \wedge Q(a,ga), P(b,fb), Q(b,gb) \vdash P(a,fa) \wedge Q(b,gb), \Delta}}{l:\wedge}}{\Gamma, P(a,fa) \wedge Q(a,ga), P(b,fb) \wedge Q(b,gb) \vdash P(a,fa) \wedge Q(b,gb), \Delta}}{r:\exists}}{\Gamma, P(a,fa) \wedge Q(a,ga), P(b,fb) \wedge Q(b,gb) \vdash \exists y.P(a,fa) \wedge Q(b,y), \Delta'}}{r:\exists}}{\Gamma, P(a,fa) \wedge Q(a,ga), P(b,fb) \wedge Q(b,gb) \vdash \exists x, y.P(a,x) \wedge Q(b,y)}}{l:\forall}}{\Gamma, P(a,fa) \wedge Q(a,ga) \vdash \exists x, y.P(a,x) \wedge Q(b,y)}}{l:\forall}}{\forall x.P(x,fx) \wedge Q(x,gx) \vdash \exists x, y.P(a,x) \wedge Q(b,y)}$$

where

$$\Gamma \stackrel{def}{=} \forall x.P(x,fx) \wedge Q(x,gx),$$
$$\Gamma_l \stackrel{def}{=} \{\forall x.P(x,fx) \wedge Q(x,gx), Q(a,ga), P(b,fb), Q(b,gb)\},$$
$$\Gamma_r \stackrel{def}{=} \{\forall x.P(x,fx) \wedge Q(x,gx), P(a,fa), Q(a,ga), P(b,fb)\},$$
$$\Delta \stackrel{def}{=} \{\exists x, y.P(a,x) \wedge Q(b,y), \exists y.P(a,fa) \wedge Q(b,y)\}, \text{ and}$$
$$\Delta' \stackrel{def}{=} \exists x, y.P(a,x) \wedge Q(b,y)$$

(b) Different constants

argument is a and that at the same time Q holds when the first argument is b instead of a. This can be expressed as $\exists x, y.P(a,x) \wedge Q(b,y)$. A cut-free proof is depicted in Figure 4.6b.

In general, one could say that in these examples there is no need to introduce cuts. But if already there cut introduction fails, their simplicity rather suggests that the problem occurs more often, also in more complex examples. Nonetheless, the reason why cut introduction fails lays in the chosen $\mathbf{S\Pi_2}$-$\mathbf{G}$. Abstracting the left context by the term $h_F\alpha$ where h_F is the term representation of $P(x,fx) \wedge Q(x,gx)$ and α might be replaced with a or b is still fine. Indeed, the abstraction of the conclusion is particularly chosen to fail. Here, we introduce in one formula two nonterminals (eigenvariables): $h_G\beta_1\beta_2$ where h_G is the term representation of either $P(a,x) \wedge Q(a,y)$ or $P(a,x) \wedge Q(b,y)$. Since the conclusion is a conjunction of two atoms with different predicate symbol and since we have to prove both parts by the potential cut formula, the cut formula would then have to introduce both eigenvariables at once which is impossible.

Lemma 3 (See Lemma 2 of [LL18]) *Let* $F =_{def} P(x,fx) \wedge Q(x,gx)$ *and* $G =_{def} P(a,x) \wedge Q(a,y)$ *and consider the sequent* $S =_{def} \forall F \vdash \exists G$, *the* Π_2-***SEHS*** $\mathcal{S}(S)$

$$\begin{gathered} P(\alpha, f\alpha) \wedge Q(\alpha, g\alpha), \\ (X\alpha f\alpha \vee X\alpha g\alpha) \rightarrow (Xa\beta_1 \wedge Xa\beta_2) \ \vdash P(a,\beta_1) \wedge Q(a,\beta_2), \end{gathered}$$

with the corresponding $\boldsymbol{S\Pi_2}$-$\boldsymbol{G}$ $\mathcal{G} = \langle \tau, N, \Sigma, \Pr \rangle$ *where* $N =_{def} \{\tau, \alpha, \beta_1, \beta_2\}$ *and*

$$\Pr \overset{def}{=} \begin{array}{l} \{\ \tau \rightarrow h_F\alpha, \tau \rightarrow h_G\beta_1\beta_2, \\ \quad \alpha \rightarrow a, \alpha \rightarrow a, \\ \quad \beta_2 \rightarrow fa, \beta_2 \rightarrow ga, \\ \quad \beta_1 \rightarrow fa, \beta_1 \rightarrow ga \qquad \}. \end{array}$$

Then $\mathcal{S}(S)$ *does not have a solution.*

Proof:
We prove the lemma by contradiction. Let us assume a valid cut-formula C that corresponds to the grammar $\mathcal{G}$. A maximal **G3c**-derivation φ of the reduced representation produces the leaves

$$\begin{array}{l} \{\ P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash P(a,\beta_1); \\ \quad P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash Q(a,\beta_2)\ \}. \end{array}$$

As C is valid, the following sequents have to be tautologies

$$\mathcal{A} \overset{def}{=} \begin{array}{l} \{\ P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash P(a, \beta_1), C(\alpha, f\alpha), C(\alpha, g\alpha); \\ \quad P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash Q(a, \beta_2), C(\alpha, f\alpha), C(\alpha, g\alpha); \\ \quad C(a, \beta_1), C(a, \beta_2), P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash P(a, \beta_1); \\ \quad C(a, \beta_1), C(a, \beta_2), P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash Q(a, \beta_2) \quad \} \end{array}$$

as are the following sequents

$$\mathcal{B} \overset{def}{=} \begin{array}{l} \{\ P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash C(\alpha, f\alpha), C(\alpha, g\alpha); \\ \quad C(a, \beta_1) \vdash P(a, \beta_1); \\ \quad C(a, \beta_2) \vdash Q(a, \beta_2) \qquad\qquad\qquad\qquad \}. \end{array}$$

The possibility to drop $P(a, \beta_1)$ and $Q(a, \beta_2)$ in the first two lines of $\mathcal{A}$ and $P(\alpha, f\alpha)$, $Q(\alpha, g\alpha)$ in the last two lines of $\mathcal{A}$ in order to obtain the sequents in $\mathcal{B}$ is obvious (Neither the formulas $C(\alpha, f\alpha), C(\alpha, g\alpha)$ can contain an atom depending on β_1 or β_2 nor $C(a, \beta_1)$ and $C(a, \beta_2)$ can contain an atom depending on α). In order to prove that we can also ignore $C(a, \beta_2)$ in the third line, we assume that $T =_{def} C(a, \beta_1) \vdash P(a, \beta_1)$ is not provable. Hence, there is a non-tautological branch $\Gamma_1 \vdash \Delta_1, P(a, \beta_1)$ in every maximal **G3c**-derivation φ of T. Given that $C(a, \beta_2)$ has the same logical structure as $C(a, \beta_1)$, we can apply the same **G3c**-rules of φ to $C(a, \beta_2)$ and get the sequent $\Gamma_2 \vdash \Delta_2$. The atoms of the sets Γ_1 and Δ_1 are the same as the atoms in Γ_2 and Δ_2 except for those which depend on the second argument of C, i.e. they contain β_1 or β_2. Thus, the sequent $\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2$ is not a tautology and also the atom $P(a, \beta_1)$ is not an element of $\Gamma_1 \cup \Gamma_2$. Then also $S' =_{def} \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, P(a, \beta_1)$ is not a tautology. But S' is a leaf of every proof tree of $C(a, \beta_1), C(a, \beta_2) \vdash P(a, \beta_1)$. This is a contradiction and therefore, T has to be a tautology. Analogously, we can prove that $C(a, \beta_2) \vdash Q(a, \beta_2)$ has to be a tautology if $C(a, \beta_1), C(a, \beta_2) \vdash Q(a, \beta_2)$ is a tautology.

If the sequents in $\mathcal{B}$ are provable, then we can replace in their proofs α with a, β_1 with ga, and β_2 with fa to get the provable sequents

$$\begin{array}{l} \{\ P(a, fa), Q(a, ga) \vdash C(a, fa), C(a, ga); \\ \quad C(a, ga) \vdash P(a, ga); \\ \quad C(a, fa) \vdash Q(a, fa) \qquad\qquad \}. \end{array}$$

Now we can apply two times the *Cut*-rule

$$Cut\ \frac{\chi \qquad C(a, ga) \vdash P(a, ga)}{P(a, fa), Q(a, ga) \vdash P(a, ga), Q(a, fa)}$$

with

$$\chi \stackrel{def}{=} \quad Cut\, \frac{P(a,fa), Q(a,ga) \vdash C(a,fa), C(a,ga) \qquad C(a,fa) \vdash Q(a,fa)}{P(a,fa), Q(a,ga) \vdash Q(a,fa), C(a,ga)}$$

and derive the sequent $P(a,fa), Q(a,ga) \vdash Q(a,fa), C(a,ga)$. But this sequent is not valid and by contradiction, there is no cut formula. □

In general, this example suffices to show that there is not always a solution for a Π_2-**SEHS**. But at this point, one can argue that we have to refine the definition of **S**Π_2-**G**. If production rules have to be unique, then the given example would be inappropriate (the production rules with β_1 and β_2 on the left map on the same terms and are, therefore, not unique). The Π_2-**SEHS** of Lemma 4 only contains unique production rules.

Lemma 4 (See Lemma 1 of [LL18]) *Let* $F =_{def} P(x,fx) \wedge Q(x,gx)$ *and* $G =_{def} P(a,x) \wedge Q(b,y)$. *Assume the sequent* $S =_{def} \forall F \vdash \exists G$, *the* Π_2-***SEHS*** $\mathcal{S}(S)$

$$P(\alpha, f\alpha) \wedge Q(\alpha, g\alpha),$$
$$(X\alpha f\alpha \vee X\alpha g\alpha) \rightarrow (Xa\beta_1 \wedge Xb\beta_2) \vdash P(a,\beta_1) \wedge Q(b,\beta_2),$$

with the corresponding ***S***Π_2-***G*** $\mathcal{G} = \langle \tau, N, \Sigma, \Pr \rangle$ *where* $N =_{def} \{\tau, \alpha, \beta_1, \beta_2\}$ *and*

$$\Pr \stackrel{def}{=} \begin{array}{l} \{\ \tau \rightarrow h_F\alpha, \tau \rightarrow h_G\beta_1\beta_2, \\ \quad \alpha \rightarrow a, \alpha \rightarrow b, \\ \quad \beta_2 \rightarrow fb, \beta_2 \rightarrow gb, \\ \quad \beta_1 \rightarrow fa, \beta_1 \rightarrow ga \qquad \}. \end{array}$$

Then $\mathcal{S}(S)$ *does not have a solution.*

Proof:
In order to prove the lemma we provide a model equating a and b because such models, together with Lemma 3 entail the non-existence of a cut formula. Let a be the natural number 0, b be the natural number 2, $\lambda x.f$ be the successor function $\lambda x.sx$, and $\lambda x.g$ be $\lambda x.ssx$. As a model we consider the natural numbers modulo 2. Hence, the interpretation of a and of b are both the common representative of the equivalence class of even numbers. Furthermore, we can interpret P as the relation that is true iff its arguments are not equal and Q is true iff its arguments are equal. In this model, a is equal to b and hence, there cannot be a cut-formula. □

The lemmas show that Π_2-cut introduction as formulated in this thesis is not always solvable. In the following sections, we concentrate on the solvability of a relevant fragment and a complexity analysis. An interesting topic that is not addressed is whether there

are completely solvable reformulations of Π_2-cut introduction. One candidate strongly suggested by Remark 3 is to add equality in order to allow case distinctions. In this scenario, we would be able to find a cut formula. When comparing the size of the cut formula in the remark with the actual end sequent, we see that this would increase the complexity of potential cut formulas significantly. This is due to the additional expressibility. Since it is already hard to find Π_2-cut formulas as outlined in this thesis, it seems more reasonable to first consider a scenario without equality.

Remark 3 (See Remark 2 of [LL18]) *If we take a sequent calculus with equality and add the formula $\neg a = b$ to the left of the end-sequent, i.e. an additional assumption, then*

$$\forall x \exists y. C(x,y) \stackrel{def}{=} \forall x \exists y. (x = a \to P(x,y)) \wedge (\neg x = a \to Q(x,y))$$

is a valid cut formula that corresponds to the given $\boldsymbol{S}\Pi_2$-$\boldsymbol{G}$. Furthermore, the proof depicted in Figure 4.7 is a valid proof with Π_2 cut corresponding to the $\boldsymbol{S}\Pi_2$-$\boldsymbol{G}$ of Lemma 4. Note that the sequents in Line (4.1) *and in Line* (4.2) *are only provable in a sequent calculus with equality where the rules*

$$\textit{Ref}{:=}\frac{\Gamma, r = r \vdash \Delta}{\Gamma \vdash \Delta} \qquad \textit{Rep}{:=}\frac{\Gamma, r = s, R[x\backslash r], R[x\backslash s] \vdash \Delta}{\Gamma, r = s, R[x\backslash r] \vdash \Delta}$$

with atomic R are added (see [TS96, Section 4.7]).

Both examples show that, in general, we cannot expect to find a solution for a Π_2-**SEHS**. Moreover, it is difficult to give an easy restriction to the grammar such that the solvability is guaranteed.

4.6 A Characterization of Solvability

As outlined in the previous section, we cannot hope for a canonical solution, since there are unsolvable Π_2-**SEHS**s. In order to find out whether a Π_2-**SEHS** is solvable, we have to characterize some conditions for the introduction of Π_2 cuts. In general, the cut formula can be any logical combination of literals in the given signature such that the result is a Π_2 formula. For this reason, we define a restriction of the search space such that the solvability for this restricted search space is decidable. Assume a finite number of literals and that all formulas are in prenexed **DNF**. Then there is only a finite number of combinations up to redundancies that are Π_2 formulas.

In order to define such a restriction, we introduce the concept of a so called *starting set*, i.e. a finite set of sets of literals. It may contain a set of clauses that is interpreted as a formula in **DNF** a solution for the Π_2-**SEHS**. Later, we will define starting sets that always contain a solution as a subset for certain classes of solutions.

Figure 4.7: Proof of the counterexample in a system with equality; Remark 3

$$\textit{Cut}\ \dfrac{r{:}\forall\ \dfrac{\varphi_l}{\Gamma_l \vdash \forall x \exists y.C\,(x,y)\,, D} \qquad l{:}\forall\ \dfrac{\varphi_r}{\Gamma_l, \forall x \exists y.C\,(x,y) \vdash D}}{\neg a = b, \forall x.P\,(x, fx) \wedge Q\,(x, gx) \vdash \exists x, y.P\,(a,x) \wedge Q\,(b,y)}$$

with

$$\varphi_l \overset{def}{=} \ l{:}\forall\ \dfrac{\dfrac{\dfrac{\vdots}{\Gamma_l, P\,(\alpha, f\alpha) \wedge Q\,(\alpha, g\alpha) \vdash C\,(\alpha, g\alpha)\,, C\,(\alpha, f\alpha)\,, \Delta_l}}{r{:}\exists\ \dfrac{\Gamma_l \vdash C\,(\alpha, g\alpha)\,, C\,(\alpha, f\alpha)\,, \Delta_l}{\Gamma_l \vdash C\,(\alpha, f\alpha)\,, \exists y.C\,(\alpha, y)\,, \exists x, y.P\,(a,x) \wedge Q\,(b,y)}}}{r{:}\exists\ \dfrac{}{\Gamma_l \vdash \exists y.C\,(\alpha, y)\,, \exists x, y.P\,(a,x) \wedge Q\,(b,y)}}\ ,$$

$$\varphi_r \overset{def}{=} \begin{array}{r} r{:}\exists\ \dfrac{\dfrac{\dfrac{\vdots}{\Gamma_r, \neg a = b, C\,(a, \beta_1)\,, C\,(b, \beta_2) \vdash P\,(a, \beta_1) \wedge Q\,(b, \beta_2)\,, \Delta_r}}{\Gamma_r, \neg a = b, C\,(a, \beta_1)\,, C\,(b, \beta_2) \vdash \Delta_r}}{\Gamma_r, \neg a = b, C\,(a, \beta_1)\,, C\,(b, \beta_2) \vdash \exists x, y.P\,(a,x) \wedge Q\,(b,y)} \\ r{:}\exists\ \dfrac{}{\Gamma_r, \neg a = b, C\,(a, \beta_1)\,, \exists y.C\,(b,y) \vdash \exists x, y.P\,(a,x) \wedge Q\,(b,y)} \\ l{:}\exists\ \dfrac{}{\Gamma_r, \neg a = b, C\,(a, \beta_1) \vdash \exists x, y.P\,(a,x) \wedge Q\,(b,y)} \\ l{:}\forall\ \dfrac{}{\Gamma_r, \neg a = b, \exists y.C\,(a,y) \vdash \exists x, y.P\,(a,x) \wedge Q\,(b,y)} \end{array}\ ,$$

$D \overset{def}{=} \exists x, y.P\,(a,x) \wedge Q\,(b,y)\,,$ $\Delta_l \overset{def}{=} \{\exists y.C\,(\alpha, y)\,, \exists x, y.P\,(a,x) \wedge Q\,(b,y)\},$
$\Gamma_l \overset{def}{=} \{\neg a = b, \forall x.P\,(x, fx) \wedge Q\,(x, gx)\},$ $\Gamma_r \overset{def}{=} \{\forall x.P\,(x, fx) \wedge Q\,(x, gx)\,, \forall x \exists y.C\,(x,y)\},$
$\Delta_r \overset{def}{=} \{\exists y.P\,(a, \beta_1) \wedge Q\,(b,y)\,, \exists x, y.P\,(a,x) \wedge Q\,(b,y)\},$

the axiomatic leaves

$$\begin{array}{l} \{\ \Gamma_l, \alpha = a, P\,(\alpha, f\alpha)\,, Q\,(\alpha, g\alpha) \vdash P\,(\alpha, f\alpha)\,, C\,(\alpha, g\alpha)\,, \Delta_l; \\ \quad \Gamma_l, \alpha = a, P\,(\alpha, f\alpha)\,, Q\,(\alpha, g\alpha) \vdash \alpha = a, Q\,(\alpha, f\alpha)\,, P\,(\alpha, g\alpha)\,, \Delta_l; \\ \quad \Gamma_l, P\,(\alpha, f\alpha)\,, Q\,(\alpha, g\alpha) \vdash \alpha = a, Q\,(\alpha, f\alpha)\,, Q\,(\alpha, g\alpha)\,, \Delta_l; \\ \quad \Gamma_r, \neg a = b, \neg a = a \to Q\,(a, \beta_1)\,, C\,(b, \beta_2)\,, P\,(a, \beta_1) \vdash P\,(a, \beta_1)\,, \Delta_r; \\ \quad \Gamma_r, C\,(a, \beta_1)\,, b = a \to P\,(b, \beta_2)\,, Q\,(b, \beta_2)\ \vdash a = b, Q\,(b, \beta_2)\,, \Delta_r \qquad \}, \end{array}$$

and the leaves

$$\begin{array}{ll} \{\ \Gamma_r, \neg a = b, \neg a = a \to Q\,(a, \beta_1)\,, C\,(b, \beta_2) \vdash a = a, P\,(a, \beta_1)\,, \Delta_r; & (4.1) \\ \quad \Gamma_r, C\,(a, \beta_1)\,, b = a \to P\,(b, \beta_2)\,, b = a\ \vdash a = b, Q\,(b, \beta_2)\,, \Delta_r \qquad \} & (4.2) \end{array}$$

Definition 32 (Starting Set; See Definition 16 of [LL18]) *Let $\mathcal{O}$ be a set of variables. We call a finite set of finite sets of literals $\mathcal{A}^{\mathcal{O}}$ such that $\mathcal{V}\left(\mathcal{A}^{\mathcal{O}}\right) \subseteq \{x, y\} \cup \mathcal{O}$ for designated variables x, y a starting set. The variables $\beta_1, \ldots, \beta_p$, and α may not occur in $\mathcal{O}$. If $\mathcal{O} = \emptyset$, we abbreviate $\mathcal{A}^{\emptyset}$ by $\mathcal{A}$.*

Below, we present a method to check whether a starting set contains a solution for a given Π_2-**SEHS**. By "contain", we mean that there is a combination of literals within the starting set that actually gives us a solution.

In general, we assume that the set of variables in the reduced representation (see Definition 31) contains only the eigenvariables $\alpha, \beta_1, \ldots, \beta_p$. This is not a restriction because all other variables can be treated as constants. Hence, we can treat the variables in $\mathcal{O}$ as constants such that $\mathcal{O}$ can be considered empty. Thus, we will always consider $\mathcal{O}$ to be empty.

In order to decide, whether a formula is a solution for a Π_2-**SEHS**, we can substitute the formula for the second order variable and check whether the result is a tautology. This can be done by constructing a proof. Since the sequent is propositional, this is decidable, but still costly. If the formula was not a solution and we want to check another formula, we would have to construct another proof. In order to avoid the permanent construction of new proofs, we use the invertibility of **G3c** and apply as many rules as possible to the reduced representation of the Π_2-**SEHS**, i.e. the part that is not dependent on the cut formula. As a result, we get a set of sequents, some of them being already a tautology, some of them not. Since all these sequents occur as subsequents in a proof where we add the solution, i.e. the cut formula, we only have to consider the non-tautological sequents. The others are already tautologies and will stay tautological. Below, we define a normal form for the representation of the non-tautological leaves of the reduced representation. This can then be used to decide whether a formula is a solution.

Apart from just storing the non-tautological leaves, we also distribute the literals occurring in the sequents into three sets which will for simplicity be written as sequents. This proves to be useful, since there are never literals containing both types of eigenvariables, i.e. α and some β_i's. Hence, for certain parts of the problem we can ignore the set of literals containing α and for other parts we can ignore the set of literals containing some β_i's (see the α-problem and β-problem in the course of this section). The literals containing neither of both have always to be considered.

Proposition 11 (Non-Tautological Leaves; See Proposition 3 of [LL18]) *Let R be the reduced representation of a Π_2-**SEHS** as in Definition 31 and φ be a maximal **G3c**-derivation (see Definition 1 and Definition 2) of R. Let NTA(φ) be the* set of non-tautological leaves *of φ, i.e. the leaves that do not belong to an Ax- or an $l\colon\bot$-rule, and $S \in$ NTA(φ). Then S is of the form*

$$A(S) \circ B(S) \circ N(S)$$

where

- *$A(S)$ (A stands for "alpha") is the sequent of all atoms in S containing α,*
- *$B(S)$ (B stands for "beta") is the sequent of all atoms in S containing a non-empty subset of the variables $\{\beta_1, \ldots, \beta_p\}$, and*
- *$N(S)$ (N stands for "neutral") is the sequent of all atoms in S neither containing α nor β_i-s.*

Proof:
Let P be an atom occurring in S. We know that P is a subformula of the reduced representation R. The reduced representation $R =_{def} [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a} \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}$ can be divided into two parts: $[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a} \vdash$ and $\vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}$. In the first part, neither of the variables $\beta_1, \ldots, \beta_p$ appear; in the second part, the variable α does not appear. P is either a subformula occurring in the first or second part, i.e. it cannot contain both, variables of the set $\{\beta_1, \ldots, \beta_p\}$ and the variable α. $\square$

The proposition gives us a representation of the leaves, but in this form, we are not able to distinguish between atoms occurring on the left-hand side of a sequent and atoms occurring on the right-hand side of the sequent.

Definition 33 (Literal Normal Form; See Definition 17 of [LL18]) *Let $S =_{def} P_1, \ldots, P_l \vdash Q_1, \ldots, Q_m$ be a sequent containing only atoms. Then we define the* literal normal form *$D(S)$ of the sequent S as $\neg Q_1, \ldots, \neg Q_m, P_1, \ldots, P_l \vdash$.*

Now each literal carries the information on which side of the sequent it occurs. If it is an atom, it occurs on the left-hand side. If it is a negated atom, it occurs on the right-hand side. Hence, we can define a normal form of the sequents.

Apart from the normal form, we will define the set of all literals that can be mapped on a literal of a non-tautological leaf containing α. This will become important in Definition 36 and can be ignored in the meantime.

Definition 34 (Non-Tautological Leaves in Literal Normal Form; See Definition 18 of [LL18]) *Let $NTA(\varphi)$ be the set of non-tautological leaves of a maximal **G3c**-derivation φ of a reduced representation R. We define the* set of non-tautological leaves in literal normal form

$$DNTA(\varphi) \stackrel{def}{=} \{D(S) \mid S \in NTA(\varphi)\}.$$

Let $S \in DNTA(\varphi)$. Then S is also of the form

$$A(S) \circ B(S) \circ N(S)$$

where

- *$A(S)$ is the sequent of all literals in S containing α,*
- *$B(S)$ is the sequent of all literals in S containing a non-empty subset of the variables $\{\beta_1, \ldots, \beta_p\}$, and*
- *$N(S)$ is the sequent of all literals in S neither containing α nor β_i-s.*

Let $\mathcal{T}$ be the set of all literals. For all literals $L \in A(S)$ let

$$\xi(L) \stackrel{def}{=} \{M \mid M \in \mathcal{T} \wedge \mathcal{V}(M) \subseteq \{x, y\} \wedge \exists i \in \mathbb{N}_q M[(x, y) \setminus (\alpha, r_i)] = L\}$$

where r_i for $i \in \mathbb{N}_q$ is as in Definition 31 then

$$A^{-1}(S) \stackrel{def}{=} \bigcup_{L \in A(S)} \xi(L)$$

denotes the set of all literals that can be mapped to an element of $A(S)$.

By means of this normal form, we are able to reformulate the necessary conditions for Π_2-cut introduction. Let us reconsider the Π_2-**SEHS**

$$\mathcal{S} \stackrel{def}{=} [F[\vec{x} \backslash \vec{r}_i]]_{i=1}^{a}, \bigvee_{k=1}^{q} X \alpha r_k \to \bigwedge_{k=1}^{p} X s_k \beta_k \vdash [G[\vec{y} \backslash \vec{s}_j]]_{j=1}^{b}.$$

Instead of finding a substitution for X such that $\mathcal{S}$ becomes a tautology, we have to find a substitution $\sigma =_{def} [X \backslash \hat{C}]$ such that, for all leaves $S \in \mathit{DNTA}(\varphi)$ of the reduced representation R corresponding to $\mathcal{S}$, the sequent

$$S \circ \left(\bigvee_{k=1}^{q} \hat{C} \alpha r_k \to \bigwedge_{k=1}^{p} \hat{C} s_k \beta_k \right)$$

is a tautology. Hence, we can divide it into two problems:

- the *β-problem* of S,
$$S \circ (X s_1 \beta_1, \ldots, X s_p \beta_p \vdash)$$
and
- the *α-problem* of S,
$$S \circ (\vdash X \alpha r_1, \ldots, X \alpha r_q),$$

and say that C is a solution of the β-problem and the α-problem if there is a substitution σ for X such that σ is of the form $\lambda xy.C$ where $\beta_1, \ldots, \beta_p$, or α may not occur in C and the sequents of the β-problem and α-problem for all $S \in \mathit{DNTA}(\varphi)$ become tautologies. A shared solution for the β-problem and the α-problem is also a solution for the Π_2-**SEHS**.

Now we want to find formulas in **DNF** that are solutions. We assume an arbitrary starting set $\mathcal{A}^{\mathcal{O}}$ that is a collection of literals not containing $\beta_1, \ldots, \beta_p$, or α (see Definition 32). Again, we can set $\mathcal{O}$ to $\emptyset$. The characterization we give in this section finds (for a given starting set) all possible solutions of the β-problem and the α-problem and therefore, of the Π_2-**SEHS**. For a finite starting set we can implement a terminating algorithm in order to find all solutions that can be built by the literals in the starting set based on this characterization. Hence, after defining the characterization, we have to construct starting sets containing solutions. In Section 4.7, we define a method constructing finite starting sets, such that it always finds a solution if there is a balanced solution (see Definition 43). However, the concept of balanced solution is not needed in the characterization below.

A solution of the Π_2-**SEHS** has to solve the β-problem as well as the α-problem. Therefore, we formulate the restrictions obtained by them and gradually eliminate all subsets of $\mathcal{A}$ that are not solutions. First we consider the β-problem. In Definition 35, we eliminate all subsets of $\mathcal{A}$ that do not turn the β-problem into a tautology. Consider the sequent of the β-problem: If we substitute a possible solution in **DNF** for X, then the sequent branches into all possible sequents with one clause for each $Xs_1\beta_1, \ldots, Xs_p\beta_p$ on the left-hand side of the sequent. In Definition 35, the choice of these p arbitrary clauses is represented by the p-tuple $(C_1, \ldots, C_p)$ where C_i is instantiated with s_i and β_i for $i \in \mathbb{N}_p$. For each choice, we guarantee the provability by demanding an axiomatic constant (T_1), an axiomatic literal (T_2), or an interactive literal (T_3). These literals cover every possible case in which there is a literal and its dual on the left-hand side of the sequent. Finally, we can shift the negated literal to the right and receive a tautological axiom.

Definition 35 (Set of Possible Sets of Clauses; See Definition 19 of [LL18]) *Let R be a reduced representation of a Π_2-**SEHS** $\mathcal{S}$ and φ be a maximal **G3c**-derivation of R. Let $S \in DNTA(\varphi)$, p be the $\forall$-multiplicity (see Definition 29) of $\mathcal{S}$, and $\mathcal{C}$ be a set of clauses. Let $\vec{\mathcal{C}}_p$ be the set of all p-tuples $(C_1, \ldots, C_p)$ where $C_i \in \mathcal{C}$ for $i \in \mathbb{N}_p$. If $\vec{C} \in \vec{\mathcal{C}}_p$, $\vec{C} = (C_1, \ldots, C_p)$, and $i \in \mathbb{N}_p$ we write $\vec{C}(i)$ for C_i. Furthermore, let $\mathcal{A}$ be a starting set, $N(S)$, $B(S)$ as in Definition 34, and s_i for $i \in \mathbb{N}_p$ as in Definition 31.*

We define the three conditions – (T_1) axiomatic constant, (T_2) axiomatic literal, (T_3) interactive literal –

$$T_1\left(\vec{C}, S\right) \overset{def}{=} \exists i \in \mathbb{N}_p \exists L \in \vec{C}(i) : L[x \backslash s_i] \in \overline{N(S)},$$

$$T_2\left(\vec{C}, S\right) \overset{def}{=} \exists i \in \mathbb{N}_p \exists L \in \vec{C}(i) : L[(x,y) \setminus (s_i, \beta_i)] \in \overline{B(S)},$$

$$T_3\left(\vec{C}\right) \overset{def}{=} \exists i, j \in \mathbb{N}_p \exists L \in \vec{C}(i)\, \exists M \in \vec{C}(j) : L[(x,y) \setminus (s_i, \beta_i)] = \overline{M}[(x,y) \setminus (s_j, \beta_j)],$$

and

$$T\left(\vec{C}, S\right) \overset{def}{=} T_1\left(\vec{C}, S\right) \vee T_2\left(\vec{C}, S\right) \vee T_3\left(\vec{C}\right).$$

Then

$$Cl(\mathcal{A}) \overset{def}{=} \{\mathcal{C} \subseteq \mathcal{A} \mid \forall \vec{C} \in \vec{\mathcal{C}}_p \forall S \in DNTA(\varphi) : T\left(\vec{C}, S\right)\}$$

Figure 4.8: Cut-free proof φ; Example 18

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{F, P(a, fa), Q(a, ga) \vdash P(a, fa), Q(a, fa), G}\,Ax}{F, P(a, fa), Q(a, ga) \vdash P(a, fa) \vee Q(a, fa), G}\,r:\vee}{F, P(a, fa), Q(a, ga) \vdash P(a, fa) \vee Q(a, fa), G}\,l:\wedge}{F, P(a, fa) \wedge Q(a, ga) \vdash P(a, fa) \vee Q(a, fa), G}\,r:\exists}{F, P(a, fa) \wedge Q(a, ga) \vdash \exists x.P(a, x) \vee Q(a, x)}\,r:\exists \qquad \varphi}{F, (P(a, fa) \wedge Q(a, ga)) \vee (P(a, ga) \wedge Q(a, fa)) \vdash \exists x.P(a, x) \vee Q(a, x)}\,l:\vee}{\forall x.(P(x, fx) \wedge Q(x, gx)) \vee (P(x, gx) \wedge Q(x, fx)) \vdash \exists x.P(a, x) \vee Q(a, x)}\,l:\forall$$

with

$$\varphi \overset{def}{=} \dfrac{\dfrac{\dfrac{\dfrac{}{F, P(a, ga), Q(a, fa) \vdash P(a, ga), Q(a, ga), G}\,Ax}{F, P(a, ga), Q(a, fa) \vdash P(a, ga) \vee Q(a, ga), G}\,r:\vee}{F, P(a, ga), Q(a, fa) \vdash P(a, ga) \vee Q(a, ga), G}\,l:\wedge}{F, P(a, ga) \wedge Q(a, fa) \vdash \exists x.P(a, x) \vee Q(a, x)}\,r:\exists$$

and

$$F \overset{def}{=} \forall x.(P(x, fx) \wedge Q(x, gx)) \vee (P(x, gx) \wedge Q(x, fx))$$
$$G \overset{def}{=} \exists x.P(a, x) \vee Q(a, x)$$

---

is the set of possible sets of clauses.

A possible set of clauses, i.e. an element of the set of possible sets of clauses need not be a solution for the corresponding Π_2-**SEHS** since the α-problem might not be satisfied. Consider the following example in which we construct a non-empty set of possible sets of clauses that does not contain a solution.

Example 18 (See Example 6 of [LL18]) *Consider the proof φ of Figure 4.8. It proves the sequent*

$$\forall x.(P(x, fx) \wedge Q(x, gx)) \vee (P(x, gx) \wedge Q(x, fx)) \vdash \exists x.P(a, x) \vee Q(a, x)$$

with the two binary predicate symbols P and Q. Obviously, the left-hand side of the sequent guarantees that there are witnesses fa, ga making $P(a, fa) \vee Q(a, fa) \vee P(a, ga) \vee Q(a, ga)$ true. Furthermore, let $\mathcal{G} = \langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ be a $\boldsymbol{S}\Pi_2$-$\boldsymbol{G}$ with $N =_{def} \{\tau, \alpha, \beta\}$ and

$$\mathrm{Pr} \overset{def}{=} \{\tau \to h_F\alpha \mid h_G\beta, \alpha \to a, \beta \to fa \mid ga\}$$

where $\tau \to h_F\alpha$ is the only F-production and $\tau \to h_G\beta$ is the only G-production according to Definition 30. Then we can define the Π_2-***SEHS***

$$(P(\alpha, f\alpha) \wedge Q(\alpha, g\alpha)) \vee (P(\alpha, g\alpha) \wedge Q(\alpha, f\alpha)),$$
$$(X\alpha f\alpha \vee X\alpha g\alpha) \to Xa\beta \vdash P(a,\beta) \vee Q(a,\beta)$$

with the reduced representation

$$(P(\alpha, f\alpha) \wedge Q(\alpha, g\alpha)) \vee (P(\alpha, g\alpha) \wedge Q(\alpha, f\alpha)) \vdash P(a,\beta) \vee Q(a,\beta).$$

A maximal ***G3c****-derivation χ gives us the set of non-tautological leaves*

$$\begin{aligned} &\mathit{DNTA}(\chi) = \{S_1; S_2\} \\ &S_1 \stackrel{def}{=} P(\alpha, f\alpha), Q(\alpha, g\alpha), \neg P(a,\beta), \neg Q(a,\beta) \vdash \\ &S_2 \stackrel{def}{=} P(\alpha, g\alpha), Q(\alpha, f\alpha), \neg P(a,\beta), \neg Q(a,\beta) \vdash . \end{aligned}$$

The sequents $N(S_1)$ and $N(S_2)$ are empty and thus, also their duals $\overline{N(S_1)}$ and $\overline{N(S_2)}$ are empty. The sequents $\overline{B(S_1)}$ and $\overline{B(S_2)}$ are equal and compute to $P(a,\beta), Q(a,\beta) \vdash$. In order to solve the β-problem, we can concentrate on finding axiomatic literals. Since the cut formula in the β-problem is instantiated by $[(x,y) \setminus (a,\beta)]$, the literals $P(x,y), Q(x,y)$ are good candidates. Let us consider the starting set $\mathcal{A} =_{def} \{\{P(x,y), Q(x,y)\}\}$ and compute $Cl(\mathcal{A})$. The only subsets of $\mathcal{A}$ are the empty set and $\mathcal{A}$ itself. The empty set does not fulfill any of the conditions of a possible set of clauses. The only clause in $\mathcal{A}$ is $\{P(x,y), Q(x,y)\}$ which contains for each $S \in \mathit{DNTA}(\chi)$ an axiomatic literal, i.e. $P(a,\beta)$ and $Q(a,\beta)$. Thus, $Cl(\mathcal{A}) = \{\mathcal{A}\}$. But the Π_2-***SEHS*** *where X is replaced with $\lambda xy.P(x,y) \wedge Q(x,y)$ is not a tautology. A maximal* ***G3c****-derivation of*

$$\begin{aligned} &(P(\alpha, f\alpha) \wedge Q(\alpha, g\alpha)) \vee (P(\alpha, g\alpha) \wedge Q(\alpha, f\alpha)), \\ &(P(\alpha, f\alpha) \wedge Q(\alpha, f\alpha)) \vee (P(\alpha, g\alpha) \wedge Q(\alpha, g\alpha)) \\ &\qquad \to P(a,\beta) \wedge Q(a,\beta) \vdash P(a,\beta) \vee Q(a,\beta) \end{aligned}$$

gives us the non-tautological leaves

$$\begin{aligned} &\{P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash P(a,\beta), Q(a,\beta), Q(\alpha, f\alpha), P(\alpha, g\alpha); \\ &\ P(\alpha, g\alpha), Q(\alpha, f\alpha) \vdash P(a,\beta), Q(a,\beta), P(\alpha, f\alpha), Q(\alpha, g\alpha) \ \}. \end{aligned}$$

This is due to the existence of a leaf S in $\mathit{DNTA}(\chi)$ that fulfills the following property: we find for each term $f\alpha$ and $g\alpha$ an atom $P(\alpha, f\alpha)$ or $Q(\alpha, g\alpha)$ that does not appear in the leaf S.

In Definition 36, we generalize the property in the end of Example 18 and define a set $\mathit{ACl}(S)$ for each leaf S that contains only *allowed clauses*. Clauses as $\{P(x,y), Q(x,y)\}$ in the previous example are excluded.

In order to understand the necessity of this property for all clauses, we have to examine the behaviour of a set of clauses on the right of a sequent, i.e. the α-problem. Consider a single clause $\{L\} \cup C$ and a single instantiation (the $\exists$-multiplicity q is 1) such that $L[(x,y) \setminus (\alpha, r_1)] \notin A(S)$ for the non-tautological leaf S. Then neither $S \circ (\vdash L[(x,y) \setminus (\alpha, r_1)])$ nor $S \circ (\vdash L[(x,y) \setminus (\alpha, r_1)] \wedge C[(x,y) \setminus (\alpha, r_1)])$ is provable. If we extend the number of instantiations q without gaining an instantiation $1 \leq j \leq q$ such that for all literals M in $\{L\} \cup C$, then the substituted variant $M[(x,y) \setminus (\alpha, r_j)]$ is not an element of $A(S)$ the sequent T

$$S \circ (\vdash L[(x,y) \setminus (\alpha, r_1)] \wedge C[(x,y) \setminus (\alpha, r_1)]) \circ \ldots \circ (\vdash L[(x,y) \setminus (\alpha, r_q)] \wedge C[(x,y) \setminus (\alpha, r_q)])$$

stays non-tautological. For each instantiation of $L \wedge C$, we find a literal that does not occur in $A(S)$. Thus, there is at least one non-tautological leaf in a maximal **G3c**-derivation of T. If we consider the case that there is more than a single clause and one clause does not fulfill the described property, we can eliminate this clause. Note that if we consider the clauses made of formulas that are solutions of the β-problem, those clauses are solutions of the β-problem themselves, i.e. we are allowed to eliminate all but one clause without making the solution invalid.

Definition 36 constructs the set of all clauses with the described property for a given leaf S.

Definition 36 (Allowed Clauses and Refined Allowed Clauses; See Definition 20 of [LL18]) *Let R be a reduced representation of a Π_2-**SEHS** $\mathcal{S}$, φ be a maximal **G3c**-derivation of R, and $S \in DNTA(\varphi)$. Let $A^{-1}(S)$ be defined as in Definition 34. Let*

$$M(i) \subseteq A^{-1}(S) \text{ such that } |M(i)| = i \text{ then}$$
$$ACl(S) \stackrel{def}{=} \bigcup_{i \leq |A^{-1}(S)|} \{M(i) \mid \exists j \in \mathbb{N}_q \forall L \in M(i) : L[(x,y) \setminus (\alpha, r_j)] \in A(S)\}$$

is the set of allowed clauses.

Let $\mathcal{A}$ be a starting set as defined in Definition 32. We denote the set of refined allowed clauses $RCl(S)$ as

$$RCl(S) \stackrel{def}{=} ACl(S) \cap \mathcal{A}.$$

A useful tool for the application of the set of allowed clauses in practice can be obtained from the following proposition. It shows that the allowed clauses are closed under the subset relation. For practice, this can be used in the following way: If we checked a clause C and figured out that it is not an allowed clause, we do not have to check any other clause that contains C as a subset.

Proposition 12 (See Proposition 4 of [LL18]) *Let R be a reduced representation of a Π_2-**SEHS** $\mathcal{S}$, φ be a maximal **G3c**-derivation of R, $S \in DNTA(\varphi)$ and $ACl(S)$ the set of allowed clauses. If C is an element of $ACl(S)$ and D is a subset of C, then D is an element of $ACl(S)$.*

Proof:
The claim trivially holds by definition of $ACl(S)$. □

Now we can formulate the conditions that guarantee the provability of the sequent of the α-problem. Again, we need for each non-tautological leaf an axiomatic constant, an axiomatic literal, or an interactive literal. The differences to Definition 35 are due to the different behaviour of formulas in disjunctive normal form on different sides of a sequent in a proof in sequent calculus. In particular, the q-tuples and the Cartesian product of the subspaces $\vec{\mathcal{L}}(C)$ are only necessary to be able to consider all collections of literals (of the potential cut formula) with a corresponding substitution that can occur together in a leaf. This is indeed very technical, but can be read as follows:

1. We consider for all potential solutions occurring in $Cl(\mathcal{A})$
2. all leaves after applying a maximal number of **G3c** rules to
3. all α-problems where the potential solution in consideration is substituted for X
4. and check whether it contains an axiomatic constant, an axiomatic literal, or an interactive literal.

Note that there is a α-problem for every leaf of the maximal **G3c**-derivation of the reduced representation. Moreover, we define two sets of solution candidates; one by using the allowed clauses and the other by using the refined allowed clauses (see Definition 36).

Definition 37 (Set of Solution Candidates; See Definition 21 of [LL18]) *Let R be a reduced representation of a Π_2-**SEHS** $\mathcal{S}$ and φ be a maximal **G3c**-derivation of R. Let $S \in DNTA(\varphi)$, q be the $\exists$-multiplicity, and $\mathcal{C}$ be a set of clauses. Let $\vec{\mathcal{L}}(C)$ be the set of all q-tuples $(L_1, \ldots, L_q)$ where $L_i \in C$ for $i \in \mathbb{N}_q$ and $C \in \mathcal{C}$. If $\vec{L} \in \vec{\mathcal{L}}(C)$, $\vec{L} = (L_1, \ldots, L_q)$, and $i \in \mathbb{N}_q$ we write $\vec{L}(i)$ for $\vec{L}_i$. Let $\vec{\mathcal{C}} =_{def} \prod_{C \in \mathcal{C}} \vec{\mathcal{L}}(C)$ be the Cartesian product of the subspaces $\vec{\mathcal{L}}(C)$ where $C \in \mathcal{C}$. If $\vec{C} \in \vec{\mathcal{C}}$ and $\vec{L} \in \vec{\mathcal{L}}(C)$ is the element of $\vec{C}$ that corresponds to the subspace $\vec{\mathcal{L}}(C)$ we write $L(C, i)$ for $\vec{L}(i)$. Furthermore, let $\mathcal{A}$ be a starting set, $\mathcal{D}$ be either the set of allowed clauses of the set of refined allowed clauses, $N(S)$, $B(S)$ as in Definition 34, and r_i for $i \in \mathbb{N}_q$ as in Definition 31.*

We define the three conditions – (T_1') axiomatic constant, (T_2') axiomatic literal, (T_3')

interactive literal –

$$
\begin{aligned}
T_1'\left(\mathcal{C},\vec{C},S\right) &\stackrel{def}{=} \exists C\in\mathcal{C}\exists i\in\mathbb{N}_q\colon L\left(C,i\right)[y\backslash s_i]\in N(S)\,,\\
T_2'\left(\mathcal{C},\vec{C},S,\mathcal{D}\right) &\stackrel{def}{=} \exists C\in\mathcal{C}\exists I\in\mathcal{D}\,\forall i\in\mathbb{N}_q\colon L\left(C,i\right)\in I,\\
T_3'\left(\mathcal{C},\vec{C}\right) &\stackrel{def}{=} \exists C,D\in\mathcal{C}\exists i,j\in\mathbb{N}_q\colon L\left(C,i\right)[(x,y)\backslash(\alpha,r_i)]=\overline{L\left(D,j\right)}[(x,y)\backslash(\alpha,r_j)],\\
&\quad\ \textit{and}\\
T'\left(\mathcal{C},\vec{C},S,\mathcal{D}\right) &\stackrel{def}{=} T_1'\left(\mathcal{C},\vec{C},S\right)\vee T_2'\left(\mathcal{C},\vec{C},S,\mathcal{D}\right)\vee T_3'\left(\mathcal{C},\vec{C}\right).
\end{aligned}
$$

Then, for $\mathcal{D}=ACl(S)$*, the set*

$$Sol(\mathcal{A})\stackrel{def}{=}\{\mathcal{C}\in Cl(\mathcal{A})\mid\forall\vec{C}\in\vec{\mathcal{C}}\forall S\in DNTA\left(\varphi\right):T'\left(\mathcal{C},\vec{C},S,\mathcal{D}\right)\}$$

is called the set of solution candidates*, and, for* $\mathcal{D}=RCl(S)$*, the* set of refined solution candidates *(for a given starting set and a given* Π_2*-**SEHS** in **DNF**).*

The difference between the refined solution candidates and the solution candidates is based on the definition of the refined allowed clauses and the allowed clauses. While the allowed clauses are independent of the starting set, there are only finitely many refined allowed clauses, since we intersect the set of allowed clauses with the finite starting set. If this set were not finite, the condition $\exists I\in\mathcal{D}$ (there exists an allowed clause) in the axiomatic literal T_2' would not be computable. Fortunately, the following theorem shows that it does not matter which definition we use. All solution candidates are also refined solution candidates.

Theorem 13 (See Theorem 5 of [LL18]) *The set of refined solution candidates coincides with the set of solution candidates.*

Proof:
If $\mathcal{C}$ is a refined solution candidate, then $\mathcal{C}$ is a solution candidate by definition.

Assume $\mathcal{C}$ is a solution candidate. The only difference to a refined solution candidate is the axiomatic literal. $\mathcal{C}$ being a solution candidate, there is a clause C in $\mathcal{C}$ and an allowed clause I such that for all $i\in\mathbb{N}_q$ the literal $L\left(C,i\right)$ is an element of I. Furthermore, $\mathcal{C}$ is an element of $Cl(\mathcal{A})$, i.e. $\mathcal{C}\subseteq\mathcal{A}$. Altogether, $L\left(C,i\right)$ is a literal occurring in $\mathcal{A}$ and there is a subset J of I such that $I\supseteq J=\bigcup_{i\in\mathbb{N}_q}L\left(C,i\right)$. By Proposition 12, J is an element of $ACl(S)$ and therefore, J is a refined allowed clause. Since it is always possible to construct a refined allowed clause for a given axiomatic literal, $\mathcal{C}$ is also an element of the set of refined solution candidates. □

Example 19 (See Example 7 of [LL18]) *If we consider Example 18 again and compute $Sol(\mathcal{A})$, we will get the empty set. $RCl(\mathcal{A})$ consists of all clauses C that are an element of the starting set $\mathcal{A}$ such that there is an index $i \in \mathbb{N}_q$ for all literals L in the clause C with $L[(x,y) \setminus (\alpha, r_i)] \in A(S)$. For the two non-tautological leaves S_1, S_2, we get the sequents*

$$A(S_1) \stackrel{def}{=} P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash,$$
$$A(S_2) \stackrel{def}{=} P(\alpha, g\alpha), Q(\alpha, f\alpha) \vdash .$$

The only element of the starting set is $\{P(x,y), Q(x,y)\}$, and $q = 2$. For $i = 1$, the substituted literals are $\{P(\alpha, f\alpha), Q(\alpha, f\alpha)\}$ and for $i = 2$ the substituted literals are $\{P(\alpha, g\alpha), Q(\alpha, g\alpha)\}$. In both cases and independent from the chosen leaf ($j \in \mathbb{N}_2$), one of the substituted literals is not an element of $A(S_j)$. For instance: Since $Q(\alpha, f\alpha)$ of the substituted literals $\{P(\alpha, f\alpha), Q(\alpha, f\alpha)\}$ does not appear in

$$A(P(\alpha, f\alpha), Q(\alpha, g\alpha), \neg P(a, \beta), \neg Q(a, \beta) \vdash) = (P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash)$$

and $P(\alpha, g\alpha)$ of the substituted literals $\{P(\alpha, g\alpha), Q(\alpha, g\alpha)\}$ does not appear in

$$P(\alpha, f\alpha), Q(\alpha, g\alpha) \vdash$$

we conclude

$$RCl(P(\alpha, f\alpha), Q(\alpha, g\alpha), \neg P(a, \beta), \neg Q(a, \beta) \vdash) = \emptyset.$$

Hence, $Sol(\mathcal{A})$ is empty.

Remark 4 (See Remark 3 of [LL18]) *The condition of Definition 36 of allowed clauses is necessary.*

Proof:
Assume a solution σ of a Π_2-**SEHS** $\mathcal{S}$ such that the substituted formula C is in **DNF** and no clause fulfills the condition of Definition 36, i.e. there is a leaf S for all clauses C and all $i \in \mathbb{N}_q$ such that we find literals $L_{i,C}$ where $L_{i,C}[(x,y) \setminus (\alpha, r_i)] \notin A(S)$. Let $\mathcal{L}$ be the set of all $L_{i,C}[(x,y) \setminus (\alpha, r_i)]$ with $C \in S$ and $i \in \mathbb{N}_q$. Then $A(S) \circ (\vdash \mathcal{L})$ is a non-tautological sequent whose initial sequent appears in every proof of $\mathcal{S}\sigma$. Therefore, σ cannot be a solution. □

In a first step, we can show that each solution candidate is actually a solution. Hence, the conditions of Definition 35 and Definition 37 are sufficient and guarantee that every solution candidate solves both the α-problem and the β-problem.

Theorem 14 (Soundness; See Theorem 6 of [LL18]) *Let*

$$\mathcal{S} \stackrel{def}{=} [F[\vec{x} \backslash \vec{r_i}]]_{i=1}^{a}, \bigvee_{k=1}^{q} X\alpha r_k \to \bigwedge_{k=1}^{p} Xs_k\beta_k \vdash [G[\vec{y} \backslash \vec{s_j}]]_{j=1}^{b}$$

*be a Π_2-**SEHS**, $Sol(\mathcal{A}) \neq \emptyset$ be defined as in Definition 37 for a given starting set $\mathcal{A}$, and $\mathcal{C} \in Sol(\mathcal{A})$. Let $C =_{def} DNF(\mathcal{C})$ be the formula in **DNF** corresponding to $\mathcal{C}$ and $\sigma =_{def} [X \backslash \lambda xy.C]$. Then $\mathcal{S}\sigma$ is a tautology, i.e. solution candidates and refined solution candidates are solutions.*

Proof:
If we want to prove that

$$\mathcal{S}\sigma = [F[\vec{x}\backslash\vec{r_i}]]_{i=1}^{a}, \bigvee_{k=1}^{q} \lambda xy.C\alpha r_k \rightarrow \bigwedge_{k=1}^{p} \lambda xy.Cs_k\beta_k \vdash [G[\vec{y}\backslash\vec{s_j}]]_{j=1}^{b}$$

is a tautology, we have to prove the sequents

$$[F[\vec{x}\backslash\vec{r_i}]]_{i=1}^{a} \vdash \lambda xy.C\alpha r_1, \ldots, \lambda xy.C\alpha r_q, [G[\vec{y}\backslash\vec{s_j}]]_{j=1}^{b} \text{ and} \tag{4.3}$$

$$[F[\vec{x}\backslash\vec{r_i}]]_{i=1}^{a}, \lambda xy.Cs_1\beta_1, \ldots, \lambda xy.Cs_p\beta_p \vdash [G[\vec{y}\backslash\vec{s_j}]]_{j=1}^{b} \tag{4.4}$$

representing the substituted α-problem and the substituted β-problem. First, we show that the sequent (4.4) is a tautology. Assume it is not provable. The formulas $\lambda xy.Cs_k\beta_k$ for $k \in \mathbb{N}_p$ are formulas in **DNF** which can be interpreted as sets of sets of literals. In **G3c** a sequent with a disjunction on the left

$$\Gamma, \bigvee_{i \in I} D_i \vdash \Delta$$

is true if the sequents

$$\Gamma, D_i \vdash \Delta$$

are true for all $i \in I$. Hence, if the sequent (4.4) is not provable, then there are clauses $D_1, \ldots, D_p$ in $\mathcal{C}$ such that, for $C_i =_{def} DNF(\{D_i\})$,

$$[F[\vec{x}\backslash\vec{r_i}]]_{i=1}^{a}, \lambda xy.C_1s_1\beta_1, \ldots, \lambda xy.C_ps_p\beta_p \vdash [G[\vec{y}\backslash\vec{s_j}]]_{j=1}^{b} \tag{4.5}$$

is not provable.

Now we apply the rules of a maximal **G3c**-derivation φ of

$$R = [F[\vec{x}\backslash\vec{r_i}]]_{i=1}^{a} \vdash [G[\vec{y}\backslash\vec{s_j}]]_{j=1}^{b}$$

and let the instantiations $\lambda xy.C_1s_1\beta_1, \ldots, \lambda xy.C_ps_p\beta_p$ untouched. The non-tautological leaves of R can be represented by $DNTA(\varphi)$ where $A(S) \circ B(S) \circ N(S)$ for $S \in DNTA(\varphi)$ is defined as in Definition 34. Hence, we can add the literals of the clauses $\lambda xy.C_1s_1\beta_1$, $\ldots$, $\lambda xy.C_ps_p\beta_p$ to $B(S)$ and $N(S)$ to get a representation of the non-tautological leaves of a maximal **G3c**-derivation of the sequent (4.5). The part of literals that has been added to $B(S)$ will be denoted by B and the part that has been added to $N(S)$ will be denoted by N. If the sequent (4.5) is not provable, there has to be a non-tautological leaf S', such that

$$\forall L, M \in S' \circ B \circ N.\, L \neq \overline{M}.$$

But this implies that there is no axiomatic constant (T_1), axiomatic literal (T_2), or interactive literal (T_3). Thus it contradicts Definition 35 and the sequent (4.4) is provable.

Now we have to prove that the sequent (4.3) is a tautology. We will again assume that it is not a tautology and derive a contradiction. Let us assume there are n clauses $D_1, \ldots, D_n$ in $\mathcal{C}$. Thus, the sequent

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a} \vdash \lambda xy.C_1\alpha r_1, \ldots, \lambda xy.C_n\alpha r_1, \ldots, \lambda xy.C_1\alpha r_q, \ldots, \lambda xy.C_n\alpha r_q, [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}$$

where $C_i =_{def} DNF(\{D_i\})$ is also not a tautology. Now we apply the rules of a maximal **G3c**-derivation φ of R again and let the clauses be untouched. Given that the sequent above is not a tautology, there is also a leaf S' in the derivation that is not a tautology. We find in each $\lambda xy.C_i\alpha r_j$ for $i \in \mathbb{N}_n$ and $j \in \mathbb{N}_q$ a literal L_k with $k =_{def} (i-1)\cdot q + j$ such that

$$S' \stackrel{def}{=} S \circ (\vdash L_1, \ldots, L_{n\cdot q})$$

is not a tautology. But this implies that there is neither an axiomatic constant (T_1'), nor an axiomatic literal (T_2'), nor an interactive literal (T_3') and this contradicts Definition 37. Hence, the sequent (4.3) is a tautology. □

Furthermore, we can show that the Definitions 35 and 37 do not eliminate solutions, i.e. if there is a subset in the starting set $\mathcal{A}$ that is a solution, then this set will also be an element of $Sol(\mathcal{A})$. Important is the dependency on the starting set $\mathcal{A}$. Hence, the theorem is called "Partial Completeness" to indicate that only the existence of a sufficient starting set could guarantee full completeness. In Section 4.7, we give a construction of a starting set (see Definition 42) that allows us to formulate a completeness theorem for a relevant fragment (see Theorem 17).

Theorem 15 (Partial Completeness; See Theorem 7 of [LL18]) *Let*

$$\mathcal{S} \stackrel{def}{=} [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a}, \bigvee_{k=1}^{q} X\alpha r_k \to \bigwedge_{k=1}^{p} Xs_k\beta_k \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}$$

be a Π_2-***SEHS***, *$\mathcal{A}$ be a starting set, and $\mathcal{C} \subseteq \mathcal{A}$. Let $C =_{def} DNF(\mathcal{C})$ and $\sigma =_{def} [X\backslash\lambda xy.C]$. If $\mathcal{S}\sigma$ is a tautology, then $\mathcal{C} \in Sol(\mathcal{A})$ where $Sol(\mathcal{A})$ is as in Definition 37.*

Proof:
Let us assume that there is a solution $\mathcal{C}$ for the Π_2-**SEHS** that is a subset of the starting set $\mathcal{A}$ but $\mathcal{C}$ is not an element of $Cl(\mathcal{A})$ of Definition 35. Let φ be a maximal **G3c**-derivation of

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a} \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}.$$

If $\mathcal{C}$ is not an element of $Cl(\mathcal{A})$ but $\mathcal{C} \subseteq \mathcal{A}$ then

$$\exists \vec{C} \in \vec{\mathcal{C}_p} \exists S \in \mathit{DNTA}(\varphi) : \neg T\left(\vec{C}, S\right)$$

with

$$\begin{aligned}
\neg T\left(\vec{C}, S\right) &\stackrel{def}{=} \neg T_1\left(\vec{C}, S\right) \wedge \neg T_2\left(\vec{C}, S\right) \wedge \neg T_3\left(\vec{C}\right), \\
\neg T_1\left(\vec{C}, S\right) &\stackrel{def}{=} \forall i \in \mathbb{N}_p \forall L \in \vec{C}(i) : L[x \backslash s_i] \notin \overline{N(S)}, \\
\neg T_2\left(\vec{C}, S\right) &\stackrel{def}{=} \forall i \in \mathbb{N}_p \forall L \in \vec{C}(i) : L[(x,y) \setminus (s_i, \beta_i)] \notin \overline{B(S)}, \text{ and} \\
\neg T_3\left(\vec{C}\right) &\stackrel{def}{=} \forall i,j \in \mathbb{N}_p \forall L \in \vec{C}(i) \forall M \in \vec{C}(j) : L[(x,y) \setminus (s_i, \beta_i)] \neq \overline{M}[(x,y) \setminus (s_j, \beta_j)]
\end{aligned}$$

where $\vec{\mathcal{C}_p}$ is defined as in Definition 35. Let S be an element of $\mathit{DNTA}(\varphi)$ of the form $A(S) \circ B(S) \circ N(S)$. There is a p-tuple of clauses $(D_1, \ldots, D_p)$ with $D_i \in \mathcal{C}$ for $i \in \mathbb{N}_p$ fulfilling the following property: let $C_i =_{def} \mathit{DNF}(\{D_i\})$ for $i \in \mathbb{N}_p$ then

$$S \circ (\lambda xy.Cs_1\beta_1, \ldots, \lambda xy.Cs_p\beta_p \vdash)$$

is not a tautology. But then also

$$[F[\vec{x} \backslash \vec{r_i}]]_{i=1}^{a}, \bigwedge_{k=1}^{p} \lambda xy.Cs_k\beta_k \vdash [G[\vec{y} \backslash \vec{s_j}]]_{j=1}^{b}$$

is not a tautology, i.e. $\mathcal{C}$ is not a solution and, by contradiction, $\mathcal{C} \in Cl(\mathcal{A})$.

Now we assume $\mathcal{C} \notin Sol(\mathcal{A})$. As $\mathcal{C} \in Cl(\mathcal{A})$, we find an element $\vec{C} \in \vec{\mathcal{C}}$ and a leaf $S \in \mathit{DNTA}(\varphi)$ such that

$$\begin{aligned}
&\forall C \in \mathcal{C} \forall i \in \mathbb{N}_q \colon L(C,i)[y \backslash s_i] \notin N(S), \\
\wedge\ &\forall C \in \mathcal{C} \forall I \in \mathit{RCl}(S) \forall i \in \mathbb{N}_q \colon L(C,i) \notin I, \\
\wedge\ &\forall C, D \in \mathcal{C} \forall i,j \in \mathbb{N}_q \colon L(C,i)[(x,y) \setminus (\alpha, r_i)] \neq \overline{L(D,j)}[(x,y) \setminus (\alpha, r_j)]
\end{aligned}$$

where $\vec{\mathcal{C}}$ is defined as in Definition 37 and $\mathit{RCl}(S)$ is defined as in Definition 36. Let $\mathcal{C} = \{C_1, \ldots, C_n\}$, then, for all of them, we find q literals

$$L(C_1,1), \ldots, L(C_1,q), \ldots, L(C_n,1), \ldots, L(C_n,q)$$

such that the sequent

$$S \circ \left(\begin{array}{l} \vdash \lambda xy.\mathit{DNF}(\{L(C_1,1)\})\, \alpha r_1, \ldots, \lambda xy.\mathit{DNF}(\{L(C_1,q)\})\, \alpha r_q, \\ \quad \ldots, \\ \quad \lambda xy.\mathit{DNF}(\{L(C_n,1)\})\, \alpha r_1, \ldots, \lambda xy.\mathit{DNF}(\{L(C_n,q)\})\, \alpha r_q \end{array} \right)$$

does not contain an axiomatic constant (T_1'), an axiomatic literal (T_2'), or an interactive literal (T_3'). Furthermore, S is not a tautology and the literals

$$\lambda xy.DNF(\{L(C_1,1)\})\,\alpha r_1,\ldots,\lambda xy.DNF(\{L(C_1,q)\})\,\alpha r_q,$$
$$\ldots,$$
$$\lambda xy.DNF(\{L(C_n,1)\})\,\alpha r_1,\ldots,\lambda xy.DNF(\{L(C_n,q)\})\,\alpha r_q$$

do not contain the eigenvariables $\beta_1,\ldots,\beta_p$. Hence, none of the literals occurs in $A(S)$, $B(S)$, or $N(S)$ and the sequent is not a tautology. This contradicts the assumption that $\mathcal{C}$ is a solution and is not an element of $Sol(\mathcal{A})$. Thus, $\mathcal{C} \in Sol(\mathcal{A})$. □

As already mentioned, we need a starting set for every possible reduced representation in order to prove full completeness. Section 4.7 shows that we can define starting sets, provided that a balanced solution of the Π_2-**SEHS** exists. The characterization is complete inasmuch as it will always computes a solution if a solution can be constructed by the clauses of the starting set. So the problem reduces to find appropriate starting sets.

Finally we show that, whenever $Sol(\mathcal{A}) \neq \emptyset$ for a given starting set $\mathcal{A}$, the problem of Π_2-cut introduction is solvable. This completes, omitting the dependency on the starting set, the main goal of Π_2-cut introduction: Starting from a cut-free proof, we construct a proof with a Π_2 cut using a Π_2-**EHS** and a **S**Π_2-**G** according to Figure 4.5. More precisely:

1. We start with a cut-free proof and a **S**Π_2-**G**.
2. We formulate the Π_2-**SEHS**.
3. (We find a starting set.)
4. We specify all subproblems by Definition 35 and Definition 37 and get the set of solution candidates.
5. We construct a Π_2-**EHS** for an arbitrary solution candidate.
6. We construct a proof with Π_2 cut.

Theorem 16 (See Theorem 8 of [LL18]) *Let*

$$\mathcal{S} \stackrel{def}{=} [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a},\ \bigvee_{k=1}^{q} X\alpha r_k \rightarrow \bigwedge_{k=1}^{p} X s_k \beta_k \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}$$

be a Π_2-***SEHS*** *corresponding to a Herbrand sequent of a cut-free proof of the sequent* $S =_{def} \forall\vec{x}F \vdash \exists\vec{y}G$ *and a* ***S***Π_2-***G*** $\mathcal{G}$ *covering the Herbrand term set of* S*. Let* $Sol(\mathcal{A}) \neq \emptyset$ *be defined as in Definition 37 for a given starting set* $\mathcal{A}$*, and* $\mathcal{C} \in Sol(\mathcal{A})$*. Let* $C =_{def} DNF(\mathcal{C})$ *be the formula corresponding to* $\mathcal{C}$ *and* $\mathcal{V}(C) \subseteq \{x,y\}$*. Then there exists a proof of* S *with one cut and the cut formula* $\forall x\exists yC$*.*

Proof:
If there is an element $\mathcal{C}$ in $Sol(\mathcal{A})$ for a given starting set $\mathcal{A}$ and a given Π_2-**SEHS**, we are able to construct a proof with a Π_2 cut. Consider the Π_2-**SEHS**

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^a, \bigvee_{k=1}^q X\alpha r_k \to \bigwedge_{k=1}^p X s_k\beta_k \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^b$$

and the clause set $\mathcal{C} \in Sol(\mathcal{A})$ for the starting set $\mathcal{A}$. Let $C =_{def} DNF(\mathcal{C})$. Then there are maximal **G3c**-derivations χ and ψ with tautological leaves of the sequents

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^a \vdash \bigvee_{k=1}^q \lambda xy\,(C)\,\alpha r_k, [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^b$$

and

$$[F[\vec{x}\backslash\vec{r}_i]]_{i=1}^a, \bigwedge_{k=1}^p \lambda xy\,(C)\,s_k\beta_k \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^b,$$

respectively. The proof below is valid and contains a single Π_2 cut:

$$\dfrac{\dfrac{\dfrac{\chi}{\Gamma, [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^a \vdash \bigvee_{k=1}^q \lambda xy\,(C)\,\alpha r_k, [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^b, \Delta'}}{\vdots}}{\forall\vec{x}F \vdash \forall x\exists y C, \exists\vec{y}G} \quad Cut \quad \dfrac{(\varphi)}{\forall\vec{x}F, \forall x\exists y C \vdash \exists\vec{y}G}}{\forall\vec{x}F \vdash \exists\vec{y}G}$$

with

$$\varphi \overset{def}{=} \quad \dfrac{\dfrac{\psi}{\Gamma', [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^a, \bigwedge_{k=1}^p \lambda xy\,(C)\,s_k\beta_k \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^b, \Delta}}{\vdots} \quad .$$

This is guaranteed by the Theorems 14 and 10 and, hence, solves the main problem of this chapter. □

4.7 The Unification Method

In the previous section, we developed a method to check whether a given starting set contains a solution for a Π_2-**SEHS**. However, we did not explain how such starting sets can be constructed. In this section, we present a method that produces a starting set for a given reduced representation of a Π_2-**SEHS**. This starting set will contain a solution if there is a so-called *balanced solution*.

In order to understand the construction of the starting set, we take a look at the leaves $\mathit{DNTA}(\varphi)$ of a maximal **G3c**-derivation φ of a given reduced representation R. If we do not consider interactive literals as in Definition 37 (T_3') and as in Definition 35 (T_3), then a solution of the corresponding Π_2-**SEHS** contains for each leaf S in $\mathit{DNTA}(\varphi)$ at least one literal L with $\mathcal{V}(L) \subseteq \{x, y\}$, such that L becomes an element of $A(S)$, $B(S)$, or $N(S)$ under the correct substitution for x and y. Hence, the first approach consists in collecting all literals that can be instantiated to at least one element of $A(S)$, $B(S)$, or $N(S)$. Then we consider all possible sets containing a subset of these literals (see the naive starting set in Definition 44).

Definition 38 (See Definition 22 of [LL18]) *A literal L with $\mathcal{V}(L) \subseteq \{x, y\}$* interacts *with a literal in $A(S)$, $B(S)$, or $N(S)$ if there are substitutions $[(x, y) \setminus (t_1, t_2)]$ corresponding to the Π_2-**SEHS** such that $L[(x, y) \setminus (t_1, t_2)]$ is an element of $A(S)$, $B(S)$, or $N(S)$. We say $[(x, y) \setminus (t_1, t_2)]$ corresponds to the Π_2-**SEHS** (see Definition 31) if $t_1 = \alpha \wedge t_2 = r_i$ for some $i \in \mathbb{N}_q$ or $t_1 = s_j \wedge t_2 = \beta_j$ for some $j \in \mathbb{N}_p$ where $\alpha, \beta_1, \ldots, \beta_p, r_1, \ldots, r_q, s_1, \ldots, s_p$ are as in Definition 31.*

Let us assume that a literal L of the solution interacts twice with a literal in $A(S)$, $B(S)$, or $N(S)$: $L[(x, y) \setminus (\alpha, r_i)] = L_\alpha$ and $L[(x, y) \setminus (s_j, \beta_j)] = L_\beta$. We call L_α and L_β *interacting literals*. In a reduced representation R and therefore, also in all elements of $\mathit{DNTA}(\varphi)$ with a maximal **G3c**-derivation of R the literals occur in the form of L_α or L_β. We will present a method that searches for interacting literals and constructs afterwards the common shape of L_α and L_β, i.e. L with $\mathcal{V}(L) \subseteq \{x, y\}$. The basic idea of the unification method to be defined below is to find all interacting literals and use them for the construction of a starting set. The benefit of this approach is that on one side the number of interacting literals that have a common shape is relatively small and on the other side the literals seem to be most promising, since they can form tautological leaves for the α-problem and the β-problem. Theorem 17 will show that the naive approach and the approach via interacting literals find solutions – in case balanced solutions exist. Thus, so far, the unification method is the most efficient method for this fragment.

In the first step, we collect for each $S \in \mathit{DNTA}(\varphi)$ all pairs of literals which are potential candidates for interacting literals.

Definition 39 (Unification Candidates; See Definition 23 of [LL18]) *Let $\mathcal{S}$ be a Π_2-**SEHS** with the corresponding reduced representation R of $\mathcal{S}$. Let $S, S' \in \mathit{DNTA}(\varphi)$ for a maximal **G3c**-derivation φ of R. Then*

$$\mathit{UC}(S, S') \stackrel{def}{=} \{(L, M) \mid L \in A(S) \cup N(S) \wedge M \in B(S') \cup N(S')\}$$

is the set of unification candidates *for the leaves S and S'.*

Example 20 *Let $\mathcal{S}$ be as in Example 17:*

$$P(\alpha, f\alpha) \vee P(\alpha, g\alpha), (X\alpha f\alpha \vee X\alpha g\alpha) \to (Xa\beta_1 \wedge Xh\beta_1\beta_2) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2).$$

*Let φ be the maximal **G3c**-derivation*

$$\dfrac{\dfrac{S_1 \qquad S_2}{P(\alpha, f\alpha) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2)}\, r:\wedge \qquad \dfrac{S_3 \qquad S_4}{P(\alpha, g\alpha) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2)}\, r:\wedge}{P(\alpha, f\alpha) \vee P(\alpha, g\alpha) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2)}\, l:\vee$$

of the reduced representation $R =_{def} P(\alpha, f\alpha) \vee P(\alpha, g\alpha) \vdash P(a, \beta_1) \wedge P(h\beta_1, \beta_2)$ with

$$\begin{aligned}
S_1 &\overset{def}{=} P(\alpha, f\alpha) \vdash P(a, \beta_1), & \overline{S_1} &= \neg P(a, \beta_1), P(\alpha, f\alpha) \vdash,\\
S_2 &\overset{def}{=} P(\alpha, f\alpha) \vdash P(h\beta_1, \beta_2), & \overline{S_2} &= \neg P(h\beta_1, \beta_2), P(\alpha, f\alpha) \vdash,\\
S_3 &\overset{def}{=} P(\alpha, g\alpha) \vdash P(a, \beta_1), & \overline{S_3} &= \neg P(a, \beta_1), P(\alpha, g\alpha) \vdash, \text{ and}\\
S_4 &\overset{def}{=} P(\alpha, g\alpha) \vdash P(h\beta_1, \beta_2), & \overline{S_4} &= \neg P(h\beta_1, \beta_2), P(\alpha, g\alpha) \vdash.
\end{aligned}$$

Then $NTA(\varphi) = \{S_1, S_2, S_3, S_4\}$ and $DNTA(\varphi) = \{\overline{S_1}, \overline{S_2}, \overline{S_3}, \overline{S_4}\}$. The set of unification candidates for $\overline{S_1}$ and $\overline{S_2}$ is $\{(P(\alpha, f\alpha), \neg P(h\beta_1, \beta_2))\}$ and the union over all sets of unification candidates $I =_{def} \bigcup_{S,S' \in DNTA(\varphi)} UC(S, S')$ is

$$\begin{aligned}
\{\ &(P(\alpha, f\alpha), \neg P(a, \beta_1)), (P(\alpha, f\alpha), \neg P(h\beta_1, \beta_2)),\\
&(P(\alpha, g\alpha), \neg P(a, \beta_1)), (P(\alpha, g\alpha), \neg P(h\beta_1, \beta_2))\ \}.
\end{aligned}$$

Since all interacting literals also occur as a unification candidate, we have to define a method constructing the common shape. In general, this is an anti-unification problem. In this particular case, we have to take care of the grammar $\mathcal{G}$ in consideration. While a general anti-unification problem would replace arbitrary terms by an arbitrary number of variables, we have to ensure that the common shape, i.e. the resulting literal which might occur in a cut formula, can be mapped on the interacting literals via substitutions that correspond to the grammar. Moreover, the common shape is not allowed to contain any of the eigenvariables/nonterminals. For this reason, we introduce a specific unrestricted tree grammar (see Definition 16). It is based on the given $\mathbf{S}\Pi_2$-$\mathbf{G}$, but with a slightly changed signature and new production rules. We add the two variables x and y to the signature, i.e. as terminals, in order to have fresh and common variables for the cut formula. Note that we only consider Π_2-cut formulas with two quantified variables. The variable x will correspond to the universally quantified variable and the variable y will correspond to the existentially quantified.

Concerning the new production rules, the most important ones are collected in the sets S_2 and S_3. When comparing with the original $\mathbf{S}\Pi_2$-$\mathbf{G}$, we recognize that terms can be mapped to the universally quantified variable x if the terms correspond to substitutions

of x in the proof of the potential cut formula. Analogously, terms can be mapped to y if they correspond to substitutions of y in the proof of the potential cut formula. Since several of those terms are terminals (and not nonterminals) in the considered grammar, we have to use an unrestricted tree grammar.

Definition 40 (See Definition 24 of [LL18]) *Let $\mathcal{G} =_{def} \langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ be a $\boldsymbol{S}\Pi_2$-$\boldsymbol{G}$ with the nonterminals $\tau, \beta_1, \ldots, \beta_p$, and α. We define the unrestricted tree grammar $\mathcal{G}^* =_{def} \langle \tau, N, \Sigma^*, \mathrm{Pr}^* \rangle$ by*

$$\Sigma^* \stackrel{def}{=} \Sigma \cup \{x, y\} \text{ and } \mathrm{Pr}^* \stackrel{def}{=} S_1 \cup S_2 \cup S_3 \text{ with}$$
$$S_1 \stackrel{def}{=} \{\rho \mid \rho \text{ is an } F\text{-production or a } G\text{-production}\}$$
$$\text{(see Definition 30),}$$
$$S_2 \stackrel{def}{=} \{\alpha \to x, r_1 \to x, \ldots, r_q \to x\}, \text{ and}$$
$$S_3 \stackrel{def}{=} \{s_1\alpha \to y, \ldots, s_p\alpha \to y, \beta_1 \to y, \ldots, \beta_q \to y\}.$$

Example 21 *The $\boldsymbol{S}\Pi_2$-$\boldsymbol{G}$ $\mathcal{G}$ of Example has the production rules*

$$\tau \to h_F\alpha \mid h_G\beta_1\beta_2,$$
$$\alpha \to a \mid h\beta_1,$$
$$\beta_2 \to fh\beta_1 \mid gh\beta_1, \text{ and}$$
$$\beta_1 \to fa \mid ga.$$

Then the corresponding grammar $\mathcal{G}^*$ has the production rules

$$\tau \to h_F\alpha \mid h_G\beta_1\beta_2,$$
$$\alpha \to x, a \to x, h\beta_1 \to x,$$
$$\beta_2 \to y, \beta_1 \to y, f\alpha \to y \text{ and}$$
$$g\alpha \to y.$$

Note that in a proof of a potential cut formula with the two quantified variables x and y will eventually be instantiated such that the pair (x, y) will be replaced with:

$$(\alpha, f\alpha), (\alpha, g\alpha), (a, \beta_1), (h\beta_1, \beta_2).$$

The substitutions of $\mathcal{G}^*$ concerning α, β_1, and β_2 result from turning these substitutions around. Instead of replacing x with α, we substitute x for α, i.e. $\alpha \to x$.

Concerning the definition of the unification method, we need a notion of a derivation applied to a literal. A derivation d consists of a finite number of positions $p_1, \ldots, p_n$ and production rules $r_1 \to s_1, \ldots, r_n \to s_n$. If we apply d to a literal L, i.e. $L|d$, then we

replace sequentially the terms r_i with s_i at the positions p_i for $i = 1$ until $i = n$. Let L_j be the literal after the j-th replacement. If p_{j+1} does not occur in L_j, then $L_{j+1} = L_j$.

Now, that we have settled the framework for the construction of common shapes of interacting literals, from now on called $\mathcal{G}^*$-*unified literal*, we can define the method that computes $\mathcal{G}^*$-unified literals. For this reason, we consider all unification pairs and try to unify them according to the fresh grammar $\mathcal{G}^*$. In case this unification process succeeds, we add the $\mathcal{G}^*$-unified literal to set which in the end will define a new starting set for the Π_2-cut introduction algorithm. Since this unification corresponds to the grammar $\mathcal{G}^*$, we call it $\mathcal{G}^*$-*unification*.

Definition 41 ($\mathcal{G}^*$-Unifiability; See Definition 25 of [LL18]) *Assume a Π_2-**SEHS** with the corresponding reduced representation R and **S**Π_2-**G** $\mathcal{G}$. Let $S, S' \in DNTA(\varphi)$ for a maximal **G3c**-derivation φ of R, $(L, M) \in UC(S, S')$, and $\mathcal{G}^* = \langle \tau, N, \Sigma^*, \Pr^* \rangle$ as in Definition 40.*

We say (L, M) is $\mathcal{G}^*$-unifiable *if there are derivations d and e in $\mathcal{G}^*$ such that $L|d = \overline{M}|e$ and $\mathcal{V}(L|d) \subseteq \{x, y\}$. Furthermore, we call $L|d$ the* $\mathcal{G}^*$-unified literal *of (L, M).*

We call R $\mathcal{G}^*$-unifiable if we find for every $S \in DNTA(\varphi)$ an $S' \in DNTA(\varphi)$ such that there is a $\mathcal{G}^*$-unifiable unification candidate in $UC(S, S')$.

Formally, we define the maximal set of $\mathcal{G}^*$-unified literals *as*

$$MGUL(S, S') \stackrel{def}{=} \{L \mid L \text{ is a } \mathcal{G}^*\text{-unified literal of } (L_1, L_2) \in UC(S, S')\}.$$

Example 22 *Consider the unification pair $(P(\alpha, f\alpha), \neg P(h\beta_1, \beta_2))$ of Example 20 and the grammar $\mathcal{G}^*$ of Example 21. The pair is $\mathcal{G}^*$-unifiable and the $\mathcal{G}^*$-unified literal is $P(x, y)$. The necessary substitutions are*

$$\alpha \to x, f\alpha \to y, h\beta_1 \to x, \beta_2 \to y.$$

In the construction of a starting set for a unifiable reduced representation R, we use all possible clauses that consist of $\mathcal{G}^*$-unified literals.

Definition 42 (Starting Set for $\mathcal{G}^*$-unifiable Sequents; See Definition 26 of [LL18]) *Let $R =_{def} [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^a \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^b$ be a $\mathcal{G}^*$-unifiable reduced representation of a Π_2-**SEHS** with the corresponding **S**Π_2-**G** $\mathcal{G}$. Let φ be a fixed maximal **G3c**-derivation. For each pair of leaves $S, S' \in DNTA(\varphi)$, we defined in Definition 41 the maximal set of $\mathcal{G}^*$-unifiable literals $MGUL(S, S')$. Then the* starting set for the $\mathcal{G}^*$-unifiable sequent R *(shorthand:* starting set for $\mathcal{G}^*$-unifiable sequents*) is defined as*

$$\mathcal{U}(R) \stackrel{def}{=} \{C \mid C \subseteq \bigcup_{S,S' \in DNTA(\varphi)} MGUL(S, S')\}.$$

Example 23 *Let R and I be as in Example 20 and $\mathcal{G}^*$ be as in Example 21. The starting set $\mathcal{U}(R)$ for the $\mathcal{G}^*$-unifiable sequent R is $\mathcal{A} =_{def} \{\{P(x,y)\}\}$. The set of solution candidates $Sol(\mathcal{A})$ is $\{\mathcal{A}\}$ and hence, the only solution in the starting set is $\forall x \exists y P(x,y)$.*

The benefit of the starting set for $\mathcal{G}^*$-unifiable sequents is the low costs of its computation and the relative small size in comparison with the naive starting set. Indeed, the set can be computed in polynomial time. Moreover, the computation of the set of solution candidates, when considering all possible clauses consisting of a given set of literals as in the naive starting set and the starting set for $\mathcal{G}^*$-unifiable sequents, is at least exponential in the number of literals. This is due to the computation of the powerset of the set of literals.

Lemma 5 (See Lemma 3 of [LL18]) *Let R be the reduced representation of a given Π_2-**SEHS** with the corresponding **S**Π_2-**G** $\mathcal{G}$, l be the number of atoms occurring in R, and m be the length of an encoding of R. Let p and q be the $\forall$-multiplicity and the $\exists$-multiplicity, respectively. Then the starting set for $\mathcal{G}^*$-unifiable sequents $\mathcal{U}(R)$ can be constructed in polynomial time $\mathcal{O}\big(l^2 \cdot m^3 \cdot (p+q)\big)$.*

Proof:
Note that the set of pairs we can build by picking two atoms of R is a superset of all unification candidates and that checking whether an element of this superset is actually a unification candidate can be done in less than m^2 operations. The size of this superset is l^2. For each pair, we have to compare at most m symbols in order to unify them. The unification itself compares two symbols with each other or checks whether the symbols can be replaced simultaneously with x (there are $2 \cdot (p+1)$ cases) or y (there are $2 \cdot (q+p)$ cases). Altogether, there exists a constant a such that $a \cdot \big(l^2 \cdot m^3 \cdot (p+q)\big)$ is an upper bound to the number of operations to construct the starting set for $\mathcal{G}^*$-unifiable sequents $\mathcal{U}(R)$. □

As mentioned before, the starting set for $\mathcal{G}^*$-unifiable sequents suffices to find *balanced solutions*. Balanced solutions ensure that every solution of an α-problem or a β-problem contains at least one (T_1'),(T_2'),(T_1), or (T_2) as in Definition 35 and Definition 37. That is, we avoid the case where the cut formula interacts with itself.

Definition 43 (Balanced Solution; See Definition 27 of [LL18]) *Let*

$$\mathcal{S} \stackrel{def}{=} [F[\vec{x}\backslash\vec{r_i}]]_{i=1}^{a}, \bigvee_{k=1}^{q} X\alpha r_k \to \bigwedge_{k=1}^{p} X s_k \beta_k \vdash [G[\vec{y}\backslash\vec{s_j}]]_{j=1}^{b}$$

be a Π_2-***SEHS***, $\mathcal{C}$ *be a finite set of sets of literals not containing the variables* $\alpha, \beta_1, \ldots, \beta_p$, *and* $C =_{def} DNF(\mathcal{C})$ *such that*

$$\mathcal{H} \stackrel{def}{=} [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a}, \bigvee_{k=1}^{q} (\lambda xy.C)\, \alpha r_k \to \bigwedge_{k=1}^{p} (\lambda xy.C)\, s_k \beta_k \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}$$

is a tautology. Let φ *be a maximal* ***G3c****-derivation of* $\mathcal{H}$. *We say* $[X\backslash\lambda xy.C]$ *is a* balanced solution *if, in all axioms of* $\mathcal{H}$, *at least one of the active formulas is not an ancestor of* C *in* φ. *With a slight abuse of language, we also say* $\mathcal{C}$ *or* C *is a balanced solution.*

Example 24 *We can define a very simple example of a* Π_2-***SEHS*** *where the solution is not balanced. For this reason, we consider the obviously provable sequent*

$$S \stackrel{def}{=} \forall x P(x, fa) \vdash \exists y P(a, y).$$

By instantiating x *with* a *and* y *with* fa *we obtain a* ***G3c****-axiom. Nonetheless, we can define the* Π_2-***SEHS***

$$\mathcal{S} \stackrel{def}{=} P(\alpha, fa), X\alpha f\alpha \to Xa\beta \vdash P(a, \beta).$$

A solution of the problem is

$$\sigma =_{def} [X\backslash\lambda xy.P(x, fa) \wedge (\neg P(a, fx) \vee P(a, y))].$$

with the corresponding Π_2-***EHS***

$$\begin{aligned}\mathcal{H} \stackrel{def}{=} \quad & P(\alpha, fa), P(\alpha, fa) \wedge (\neg P(a, f\alpha) \vee P(a, f\alpha)) \to \\ & P(a, fa) \wedge (\neg P(a, fa) \vee P(a, \beta)) \vdash P(a, \beta).\end{aligned}$$

Consider the maximal ***G3c****-derivation depicted in Figure 4.9. Then the solution* σ *is not a balanced solution since the active formulas of*

$$\Gamma_l, P(\alpha, fa), P(a, f\alpha) \vdash P(a, f\alpha), \Delta_l$$

and

$$\Gamma_r, P(a, fa) \vdash P(a, fa), P(a, \beta), \Delta_r$$

are all ancestors of the instantiations of the cut formula, i.e. of

$$P(\alpha, fa) \wedge (\neg P(a, f\alpha) \vee P(a, f\alpha))$$

or

$$P(a, fa) \wedge (\neg P(a, fa) \vee P(a, \beta)).$$

Figure 4.9: Proof with a non-balanced Π_2-cut formula; Example 24

$$l:\rightarrow \frac{\varphi_l \qquad \varphi_r}{\mathcal{H}}$$

of $\mathcal{H}$ where

$$\varphi_l \overset{def}{=} \cfrac{Ax\ \overline{S_l} \qquad r:\neg\ \cfrac{Ax\ \overline{\Gamma_l, P(\alpha, fa), P(a, f\alpha) \vdash P(a, f\alpha), \Delta_l}}{r:\vee\ \cfrac{\Gamma_l, P(\alpha, fa) \vdash \neg P(a, f\alpha), P(a, f\alpha), \Delta_l}{\Gamma_l, P(\alpha, fa) \vdash \neg P(a, f\alpha) \vee P(a, f\alpha), \Delta_l}}}{r:\wedge\ \dfrac{}{\Gamma_l, P(\alpha, fa) \vdash P(\alpha, fa) \wedge (\neg P(a, f\alpha) \vee P(a, f\alpha)), \Delta_l}}$$

$$\varphi_r \overset{def}{=} \cfrac{\cfrac{l:\neg\ \cfrac{Ax\ \overline{\Gamma_r, P(a, fa) \vdash P(a, fa), P(a, \beta), \Delta_r}}{\Gamma_r, P(a, fa), \neg P(a, fa) \vdash P(a, \beta), \Delta_r} \qquad Ax\ \overline{S_r}}{l:\vee\ \dfrac{}{\Gamma_r, P(a, fa), \neg P(a, fa) \vee P(a, \beta) \vdash P(a, \beta), \Delta_r}}}{l:\wedge\ \dfrac{}{\Gamma_r, P(a, fa) \wedge (\neg P(a, fa) \vee P(a, \beta)) \vdash P(a, \beta), \Delta_r}}$$

$$\begin{aligned}
\Gamma_l &\overset{def}{=} \{\forall x P(x, fa)\},\\
\Gamma_r &\overset{def}{=} \{\forall x P(x, fa), \forall x \exists y.P(x, fa) \wedge (\neg P(a, fx) \vee P(a, y))\},\\
\Delta_l &\overset{def}{=} \{\exists y.P(\alpha, fa) \wedge (\neg P(a, f\alpha) \vee P(a, y)), \exists y P(a, y)\},\\
\Delta_r &\overset{def}{=} \{\exists y P(a, y)\},\\
S_l &\overset{def}{=} \Gamma_l, P(\alpha, fa) \vdash P(\alpha, fa), \Delta_l, \text{ and}\\
S_r &\overset{def}{=} \Gamma_r, P(a, fa), P(a, \beta) \vdash P(a, \beta), \Delta_r.
\end{aligned}$$

Indeed, whenever a Π_2-**SEHS** has a balanced solution, we find a solution when taking the starting set for $\mathcal{G}^*$-unifiable sequents as starting set. That does not mean that every literal of a balanced solution is the $\mathcal{G}^*$-unified literal of two interacting literals, but we will find a solution consisting only of such literals.

Theorem 17 (See Theorem 9 of [LL18]) *Let*

$$\mathcal{S} \overset{def}{=} [F[\vec{x}\backslash\vec{r}_i]]_{i=1}^{a}, \bigvee_{k=1}^{q} X\alpha r_k \rightarrow \bigwedge_{k=1}^{p} Xs_k\beta_k \vdash [G[\vec{y}\backslash\vec{s}_j]]_{j=1}^{b}$$

*be a Π_2-**SEHS** with the reduced representation R and **S**Π_2-**G** $\mathcal{G}$. Assume that $\mathcal{S}$ has a balanced solution. Then the set of solution candidates $Sol(\mathcal{U}(R))$ (defined as in Definition 37) is not empty where $\mathcal{U}(R)$ is the starting set for the $\mathcal{G}^*$-unifiable sequent R as in Definition 42.*

In order to prove the theorem, we present the same result for the naive starting set instead of the starting set for $\mathcal{G}^*$-unifiable sequents $\mathcal{U}(R)$ and conclude that $Sol(\mathcal{U}(R))$ is also not empty. Unlike the starting set for $\mathcal{G}^*$-unifiable sequents, the naive starting set collects "naively" all literals that might be relevant for the construction of a balanced solution. That is why the naive starting set is not useful for practical purposes.

Definition 44 (Naive Starting Set; See Definition 28 of [LL18]) *Let R be a reduced representation of a Π_2-**SEHS** and φ be a maximal **G3c**-derivation of R. We define for each leaf $S \in DNTA(\varphi)$ of the form $A(S) \circ B(S) \circ N(S)$ the sets*

$$NA(S) \stackrel{def}{=} \{L \mid \exists i \in \mathbb{N}_q\, (\lambda xy.L)\, \alpha r_i \in A(S) \cup N(S) \wedge \mathcal{V}(L) \subseteq \{x, y\}\} \text{ and}$$
$$NB(S) \stackrel{def}{=} \{L \mid \exists i \in \mathbb{N}_p\, (\lambda xy.L)\, s_i \beta_i \in \overline{B(S)} \cup \overline{N(S)} \wedge \mathcal{V}(L) \subseteq \{x, y\}\}.$$

Then

$$\mathcal{N}(R) \stackrel{def}{=} \{C \mid C \subseteq \bigcup_{S \in DNTA(\varphi)} NA(S) \cup NB(S)\}$$

is called the naive starting set.

Example 25 *In comparison to the starting set for $\mathcal{G}^*$-unifiable sequents, the naive starting set is much bigger. For this reason, we consider again the reduced representation R and the set of non-tautological axioms $DNTA(R)$ of Example 20. Then the naive starting set $\mathcal{N}(R)$ is*

$$\{P(x, fx), P(x, gx), P(x, y), P(a, y), P(hy, y)\}.$$

By construction of the naive starting set, the existence of a solution candidate, i.e. a solution by Theorem 14, in the presence of a balanced solution is a direct consequence of Theorem 15. We only need to show that every literal of the balanced solution occurs in the naive starting set.

Corollary 3 (See Corollary 1 [LL18]) *Let*

$$\mathcal{S} \stackrel{def}{=} [F[\vec{x} \backslash \vec{r}_i]]_{i=1}^{a}, \bigvee_{k=1}^{q} X \alpha r_k \to \bigwedge_{k=1}^{p} X s_k \beta_k \vdash [G[\vec{y} \backslash \vec{s}_j]]_{j=1}^{b}$$

*be a Π_2-**SEHS** with the reduced representation R. Assume that $\mathcal{S}$ has a balanced solution. Then the set of solution candidates $Sol(\mathcal{N}(R))$ (defined as in Definition 37) is not empty where $\mathcal{N}(R)$ is the naive starting set as in Definition 44.*

Proof:
Let $[X \backslash \lambda xy.C]$ be a balanced solution where C is in **DNF**. Definition 43 of a balanced solution implies that every literal L of C is either an element of $N(S) \cup \overline{N(S)}$ for a leaf

$S \in \mathit{DNTA}(\varphi)$ of the maximal **G3c**-derivation φ of the Π_2-**SEHS** or it is an element of the sets $\mathit{NA}(S)$ and $\mathit{NB}(S)$. For a literal L of $N(S) \cup \overline{N(S)}$, we can define $\lambda xy.L$ even though L is variable free. Hence, L is an element of $\mathit{NA}(S)$ or $\mathit{NB}(S)$. By Theorem 15, $\mathcal{C} \in \mathit{Sol}(\mathcal{N}(R))$ where $\mathcal{C}$ is the set of clauses corresponding to C. □

Given a solution which is a subset of the naive starting set, we define a new solution that is a subset of the starting set for $\mathcal{G}^*$-unifiable sequents.

Lemma 6 (See Lemma 4 of [LL18]) *Assume that $\mathit{Sol}(\mathcal{N}(R))$ contains a balanced solution for a given Π_2-**SEHS**, for a maximal **G3c**-derivation φ of its reduced representation R, and for the naive starting set $\mathcal{N}(R)$. Let $\mathcal{G}$ be the corresponding **S**Π_2-**G**. Then $\mathit{Sol}(\mathcal{U}(R)) \neq \emptyset$ for the starting set $\mathcal{U}(R)$ for $\mathcal{G}^*$-unifiable sequents.*

Proof:
Let $\mathcal{C} \in \mathit{Sol}(\mathcal{N}(R))$ be a balanced solution. We choose an arbitrary literal L of $\mathcal{C}$ that is not an element of any set of literals in $\mathcal{U}(R)$. If there are none, all literals of $\mathcal{C}$ occur in $\mathcal{U}(R)$. Since we consider in $\mathcal{U}(R)$ all possible sets with a finite number of literals, $\mathcal{C}$ is an element of $\mathcal{U}(R)$, $\mathit{Sol}(\mathcal{U}(R)) \neq \emptyset$, and we are done. Otherwise, we distinguish between two cases

$$L \in \bigcup_{S \in \mathit{DNTA}(\varphi)} \mathit{NA}(S) \text{ and } \{L\} \notin \mathcal{U}(R) \tag{4.6}$$

$$L \in \bigcup_{S \in \mathit{DNTA}(\varphi)} \mathit{NB}(S) \text{ and } \{L\} \notin \mathcal{U}(R)\,. \tag{4.7}$$

First, we consider (4.6). In this case, there is a leaf $S \in \mathit{DNTA}(\varphi)$ and there is a $j \in \mathbb{N}_q$ such that

$$(\lambda xy.L)\,\alpha r_j \in A(S) \cup N(S)\,.$$

By $\{L\} \notin \mathcal{U}(R)$, there is no leaf S' such that $M \in B(S') \cup N(S')$ where $((\lambda xy.L)\,\alpha r_j, M)$ is $\mathcal{G}^*$-unifiable with the $\mathcal{G}^*$-unified literal L. If $C \in \mathcal{C}$ and $C = \{L\}$ is a unit clause then the sequent

$$((\lambda xy.L)\,s_1\beta_1, \ldots, (\lambda xy, L)\,s_p\beta_p \vdash) \circ S$$

is not a tautology and $\mathcal{C}$ is not a solution. Thus if $C \in \mathcal{C}$ and C contains L, it cannot be a unit clause. So we define the new clause $D = C \backslash \{L\}$ and we know that D is not empty. A maximal **G3c**-derivation of the sequent $T \circ (\vdash \mathcal{D})$[1], where $\mathcal{D} = (\mathcal{C} \backslash \{C\}) \cup \{D\}$ and T is an arbitrary element of $\mathit{DNTA}(\varphi)$, contains only axioms also appearing in $T \circ (\vdash \mathcal{C})$. Hence, the new sequent is also a tautology and $\mathcal{D}$ is a solution of the α-problem.

Now, we consider the sequent $(\mathcal{D} \vdash) \circ T$ for an arbitrary $T \in \mathit{DNTA}(\varphi)$. If it were not a tautology, there would be a leaf $S' \in \mathit{DNTA}(\varphi)$ and an $i \in \mathbb{N}_p$ such that

$$\overline{(\lambda xy.L)\,s_i\beta_i} \in B(S') \cup N(S')$$

[1]For a set of clauses $\mathcal{C}$ and a sequent S, we abbreviate $S \circ (\vdash \mathit{DNF}(\mathcal{C}))$ with $S \circ (\mathcal{C})$

(Note that the given solution is a balanced solution. Otherwise, we would have to consider the case that $(\lambda xy.L)\, s_i\beta_i$ appears in $\mathcal{D}$, too.). But then there exists the $\mathcal{G}^*$-unifiable pair

$$\left((\lambda xy.L)\, \alpha r_j, \overline{(\lambda xy.L)\, s_i\beta_i}\right)$$

and L is an element of $\mathcal{U}(R)$ contradicting our assumption; we conclude that $(\mathcal{D} \vdash) \circ T$ is a tautology, i.e. $\mathcal{D}$ is also a solution of the β-problem.

By using this procedure, we can erase all literals in $\mathcal{C}$ that are elements of

$$\bigcup_{S \in \mathit{DNTA}(\varphi)} \mathit{NA}\,(S)$$

but do not appear in a clause of $\mathcal{U}(R)$.

Now let us consider (4.7). In this case there is a leaf $S \in \mathit{DNTA}\,(\varphi)$ and there is a $j \in \mathbb{N}_p$ such that $(\lambda xy.L)\, s_j\beta_j \in \overline{B(S)} \cup \overline{N(S)}$. Given that $\{L\} \notin \mathcal{U}(R)$, there is no leaf S' such that $M \in A(S') \cup N(S')$ where $\left(M, \overline{(\lambda xy.L)\, s_j\beta_j}\right)$ is $\mathcal{G}^*$-unifiable with the $\mathcal{G}^*$-unified literal L. Let C be a clause containing L. Assume C is the only clause in $\mathcal{C}$ then $\mathcal{C}$ is not a solution because $S \circ (\vdash \mathcal{C})$ contains the branch

$$S \circ (\vdash (\lambda xy.L)\, \alpha r_1, \ldots, (\lambda xy.L)\, \alpha r_q)$$

which is not a tautology. Therefore, $\mathcal{C}$ does not only contain the clause C and we can define $\mathcal{D} = \mathcal{C} \backslash \{C\}$. Since $\mathcal{C}$ contains more than one clause, $\mathcal{D}$ is not empty. A maximal **G3c**-derivation of the sequent $T \circ (\mathcal{D} \vdash)$, where T is an arbitrary element of $\mathit{DNTA}\,(\varphi)$, only contains axioms also appearing in $S \circ (\mathcal{C} \vdash)$. Hence, the new sequent is also a tautology and $\mathcal{D}$ is a solution of the β-problem.

Now we consider the sequent $T \circ (\vdash \mathcal{D})$ for an arbitrary $T \in \mathit{DNTA}\,(\varphi)$. If it were not a tautology, there would be a leaf $S' \in \mathit{DNTA}\,(\varphi)$ and an $i \in \mathbb{N}_q$ such that $(\lambda xy.L)\, \alpha r_i \in A(S') \cup N(S')$. But then there exists the $\mathcal{G}^*$-unifiable pair

$$\left((\lambda xy.L)\, \alpha r_i, \overline{(\lambda xy.L)\, s_j\beta_j}\right).$$

So we obtain $L \in \mathcal{U}(R)$ contradicting our assumption; again we conclude that $T \circ (\vdash \mathcal{D})$ is a tautology, i.e. $\mathcal{D}$ is a solution of the α-problem.

With this procedure, we can erase all literals of $\mathcal{C}$ that are elements of

$$\bigcup_{S \in \mathit{DNTA}(\varphi)} \mathit{NB}\,(S)$$

but do not appear in a clause of $\mathcal{U}(R)$.

By an exhaustive application of these two methods, we get a solution that is a subset of $\mathcal{U}(R)$. $\square$

Since the naive starting set is sufficient to find balanced solutions and since we know that the existence of a solution for the naive starting set guarantees a solution for the starting set for $\mathcal{G}^*$-unifiable sequents, we can finally prove Theorem 17.

Proof of Theorem 17
The proof can be obtained by combining Corollary 3 and Lemma 6. □

Together with the results of Section 4.6, i.e. Theorem 16, we solved the problem of finding Π_2 cuts for $\mathbf{S}\Pi_2$-**G** for the class of balanced solutions.

4.8 Generalizing the Cut Formula

In the previous sections, we considered (for the sake of simplicity) only cut formulas of the form $\forall x \exists y C(x, y)$ for single variables x, y. This section's purpose is to generalize the approach to the construction of cut formulas of the form $\forall \vec{x} \exists \vec{y} C(\vec{x}, \vec{y})$ for variable tuples $\vec{x}, \vec{y}$. Most definitions and proofs remain almost unchanged by replacing terms by tuples of terms. We indicate the changes in the most important definitions and theorems and reformulate the crucial definitions of the previous sections.

Let $\vec{x}$ be a tuple of variables and $\vec{r}$ be a tuple of terms such that $\mathbf{a}(\vec{x}) = \mathbf{a}(\vec{r}) = l$. We write $[\vec{x} \backslash \vec{r}]$ for the substitution $[\vec{x}|_1 \backslash \vec{r}|_1] \dots [\vec{x}|_l \backslash \vec{r}|_l]$. Let $\vec{s}$ be a tuple of terms (possibly) containing variables of $\vec{x}$ and $\mathbf{a}(\vec{s}) = m$. Then we write $\vec{s}\,\vec{r}$ for $(\vec{s}|_1[\vec{x} \backslash \vec{r}], \dots, \vec{s}|_m[\vec{x} \backslash \vec{r}])$.

To extend the notion of grammars, we have to allow production rules to handle tuples. A production rule of the form $\vec{\alpha} \to \vec{r}$ applied to a term s is the replacement of the nonterminals $\alpha_1, \dots, \alpha_{\mathbf{a}(\vec{\alpha})}$ according to $[\vec{\alpha} \backslash \vec{r}]$, i.e. we substitute r_i for α_i at a designated position.

Definition 45 ($\mathbf{S}\Pi_2$-G with Tuples of Variables; See Definition 29 of [LL18])
Let $\mathcal{G}$ be a totally rigid acyclic tree grammar of the form $\langle \tau, N, \Sigma, \Pr \rangle$ with $N = \{\tau, \vec{\alpha}, \vec{\beta}_1, \dots, \vec{\beta}_p\}$ and $\boldsymbol{a}\left(\vec{\beta}_i\right) = \boldsymbol{a}\left(\vec{\beta}_j\right)$ for $i, j \in \mathbb{N}_p$. We call $\mathcal{G}$ a schematic Π_2 grammar with tuples of variables *(shorthand: $\mathbf{S}\Pi_2$-**GT**) if the production rules are of the following form:*

$$\begin{aligned}
&\tau \to h_F \vec{t}_1 \mid \dots \mid h_F \vec{t}_l \mid h_G \vec{t}_{l+1} \mid \dots \mid \vec{t}_m, \\
&\vec{\alpha} \to \vec{s}_1 \mid \dots \mid \vec{s}_p, \text{ and} \\
&\vec{\beta}_i \to \vec{r}_1 \vec{s}_i \mid \dots \mid \vec{r}_q \vec{s}_i \text{ for } i \in \mathbb{N}_p
\end{aligned}$$

where

$$\begin{aligned}
&\mathcal{V}\left(\vec{t}_i\right) \subseteq \mathcal{V}\left(\vec{\alpha}\right) \text{ for } i \in \mathbb{N}_l,\\
&\mathcal{V}\left(\vec{t}_i\right) \subseteq \mathcal{V}\left(\vec{\beta}_1\right) \cup \ldots \cup \mathcal{V}\left(\vec{\beta}_p\right) \text{ for } i \in \left(\mathbb{N}_m \backslash \mathbb{N}_l\right),\\
&\mathcal{V}\left(\vec{s}_i\right) \subseteq \mathcal{V}\left(\vec{\beta}_1\right) \cup \ldots \cup \mathcal{V}\left(\vec{\beta}_{i-1}\right) \text{ for } i \in \left(\mathbb{N}_p \backslash \mathbb{N}_1\right), \text{ and}\\
&\mathcal{V}\left(\vec{s}_1\right) = \emptyset.
\end{aligned}$$

We call p the $\forall$-multiplicity*, q the* $\exists$-multiplicity *and denote $\boldsymbol{a}\left(\vec{\alpha}\right)$ by $q_\forall$ and $\boldsymbol{a}\left(\vec{\beta}_1\right)$ by $q_\exists$.*

Example 26 (See Example 8 of [LL18]) *Let a, b, c be constants, f, g, h unary functions, h_F a function with arity six, h_G a function with arity four, and $\vec{\alpha} = (\alpha_1, \alpha_2), \vec{\beta} = (\beta_1, \beta_2), \vec{\gamma} = (\gamma_1, \gamma_2)$. We define the $\boldsymbol{S}\Pi_2$-$\boldsymbol{GT}$ $\mathcal{G} = \langle \tau, N, \Sigma, \Pr \rangle$ with τ being the designated starting symbol, $N =_{def} \{\tau, \vec{\alpha}, \vec{\beta}, \vec{\gamma}\}$, $\boldsymbol{a}\left(\vec{\alpha}\right) = \boldsymbol{a}\left(\vec{\beta}\right) = \boldsymbol{a}\left(\vec{\gamma}\right) = 2$, and*

$$\Pr \overset{def}{=} \begin{array}{ll}
\{\ \tau \rightarrow & h_F(\alpha_1, \alpha_1, \alpha_2, \alpha_2, \alpha_2, \alpha_2) \mid h_G(\beta_1, \beta_2, \gamma_1, \gamma_2)\\
& \vec{\alpha} \rightarrow (a, b) \mid (c, c)\\
& \vec{\gamma} \rightarrow (fc, gc) \mid (fc, hc)\\
& \vec{\beta} \rightarrow (fa, fb) \mid (fa, hb) \qquad\qquad \}.
\end{array}$$

The language consists of the words

$$\begin{aligned}
&h_F(a, a, b, b, b, b), h_F(c, c, c, c, c, c),\\
&h_G(fa, fb, fc, gc), h_G(fa, fb, fc, hc),\\
&h_G(fa, hb, fc, gc), h_G(fa, hb, fc, hc).
\end{aligned}$$

*A corresponding Π_2-**EHS** with tuples of variables (an* extended Herbrand sequent for Π_2 cuts with tuples of variables *or shorthand: Π_2-**EHST**) can be extracted from the proof depicted in Figure 4.10. The proven sequent is of a similar structure as the proven sequent of Example 14, but with two predicate symbols. The formula Γ gives us witnesses for the second argument of P and Q when the first argument is known. This is generalized in the cut formula that formalizes the existence of witnesses. Hence, we can prove the special case in which we assume the first argument of either P or Q to be certain constants, i.e. Δ is provable.*

Since the Π_2-**EHS** with tuples of variables is exactly as in Definition 28, but with tuples of variables, we omit its definition here and proceed with the schematic extended Herbrand sequent. Note that the Π_2-**EHS** with tuples of variables is a solved schematic extended Herbrand sequent for Π_2 cuts with tuples of variables which can be used to give a precise definition.

Figure 4.10: Proof with a Π_2 cut with tuples of quantifiers; Example 26

$$Cut\ \frac{\dfrac{\vdots}{\Gamma \vdash \forall x_1, x_2 \exists y_1, y_2 C, \Delta} \qquad \dfrac{\vdots}{\Gamma, \forall x_1, x_2 \exists y_1, y_2 C \vdash \Delta}}{\Gamma \vdash \Delta}$$

where

$$\begin{aligned}
\Gamma &\stackrel{def}{=} \forall x, y. P(x, fx) \vee Q(y, gy) \vee Q(y, hy), \\
\Delta &\stackrel{def}{=} \exists z_1, \ldots, z_4. (P(a, z_1) \vee Q(b, z_2)) \wedge (P(c, z_3) \vee Q(c, z_4)), \\
C &\stackrel{def}{=} P(x_1, y_1) \vee Q(x_2, y_2),
\end{aligned}$$

and

$$\begin{aligned}
&P(\alpha_1, f\alpha_1) \vee Q(\alpha_2, g\alpha_2), P(\alpha_1, f\alpha_1) \vee Q(\alpha_2, h\alpha_2), \\
&P(a, \beta_1) \vee Q(b, \beta_2), P(c, \gamma_1) \vee Q(c, \gamma_2)
\end{aligned}$$

are all quantifier-free instantiations of the cut formula in the proof,

$$\begin{aligned}
&P(\alpha_1, f\alpha_1) \vee Q(\alpha_2, g\alpha_2) \vee Q(\alpha_2, h\alpha_2), \\
&(P(a, \beta_1) \vee Q(b, \beta_2)) \wedge (P(c, \gamma_1) \vee Q(c, \gamma_2))
\end{aligned}$$

are all quantifier-free instantiations of the context $\Gamma \vdash \Delta$ in the proof.

Definition 46 (Π_2-SEHS with Tuples of Variables; See Definition 30 of [LL18]) *Let S be the provable sequent $\forall \vec{z}_1 F \vdash \exists \vec{z}_2 G$ and H_s be a Herbrand term set of S. Let $\mathcal{G} = \langle \tau, N, \Sigma, \mathrm{Pr} \rangle$ be a $\boldsymbol{S}\Pi_2$-$\boldsymbol{GT}$ as in Definition 45 with the nonterminals $N =_{def} \{\tau, \vec{\alpha}, \vec{\beta}_1, \ldots, \vec{\beta}_p\}$ and the production rules*

$$\begin{aligned}
\mathrm{Pr} \stackrel{def}{=} \{\ & \tau \to h_F \vec{t}_1 \mid \ldots \mid h_F \vec{t}_l \mid h_G \vec{t}_{l+1} \mid \ldots \mid \vec{t}_m, \\
& \vec{\alpha} \to \vec{s}_1 \mid \ldots \mid \vec{s}_p, \\
& \vec{\beta}_i \to \vec{r}_1 \vec{s}_i \mid \ldots \mid \vec{r}_q \vec{s}_i \text{ for } i \in \mathbb{N}_p \qquad \}
\end{aligned}$$

where

$$\begin{aligned}
&\mathcal{V}\left(\vec{t}_i\right) \subseteq \mathcal{V}\left(\vec{\alpha}\right) \text{ for } i \in \mathbb{N}_l,\\
&\mathcal{V}\left(\vec{t}_i\right) \subseteq \mathcal{V}\left(\vec{\beta}_1\right) \cup \ldots \cup \mathcal{V}\left(\vec{\beta}_p\right) \text{ for } i \in \left(\mathbb{N}_m \backslash \mathbb{N}_l\right),\\
&\mathcal{V}\left(\vec{r}_i\right) \subseteq \mathcal{V}\left(\vec{\alpha}\right),\\
&\mathcal{V}\left(\vec{s}_i\right) \subseteq \mathcal{V}\left(\vec{\beta}_1\right) \cup \ldots \cup \mathcal{V}\left(\vec{\beta}_{i-1}\right) \text{ for } i \in \left(\mathbb{N}_p \backslash \mathbb{N}_1\right), \text{ and}\\
&\mathcal{V}\left(\vec{s}_1\right) = \emptyset.
\end{aligned}$$

Let $\mathcal{L}\left(\mathcal{G}\right)$ be the language of $\mathcal{G}$ generated only by rigid derivations, $H_s \subseteq \mathcal{L}\left(\mathcal{G}\right)$, and X be a $\left(q_\forall + q_\exists\right)$-place predicate variable. Then we call the sequent

$$\mathcal{S}\left(S\right) \stackrel{def}{=} \left[F[\vec{z}_1 \backslash \vec{t}_i]\right]_{i=1}^{l}, \bigvee_{k=1}^{q} X\vec{\alpha}\vec{r}_k \rightarrow \bigwedge_{k=1}^{p} X\vec{s}_k\vec{\beta}_k \vdash \left[G[\vec{z}_2 \backslash \vec{t}_j]\right]_{j=l+1}^{m},$$

a schematic extended Herbrand sequent for Π_2 cuts with tuples of variables *(shorthand:* Π_2-**SEHST***) corresponding to $\mathcal{G}$ and S. Furthermore, we call*

$$\left[F[\vec{z}_1 \backslash \vec{t}_i]\right]_{i=1}^{l} \vdash \left[G[\vec{z}_2 \backslash \vec{t}_j]\right]_{j=l+1}^{m}$$

the reduced representation *of $\mathcal{S}\left(S\right)$.*

Example 27 (See Example 9 of [LL18]) *Let $\mathcal{G}$ be as in Example 26. Then we can define the Π_2-**SEHST***

$$\begin{aligned}
&P\left(\alpha_1, f\alpha_2\right) \vee Q\left(\alpha_2, g\alpha_2\right) \vee Q\left(\alpha_2, h\alpha_2\right),\\
&\left(X\alpha_1\alpha_2 f\alpha_1 g\alpha_2 \vee X\alpha_1\alpha_2 f\alpha_1 h\alpha_2\right) \rightarrow \left(Xab\beta_1 \wedge Xcc\gamma_1\gamma_2\right)\\
&\vdash \left(P\left(a, \beta_1\right) \vee Q\left(b, \beta_2\right)\right) \wedge \left(P\left(c, \gamma_1\right) \vee Q\left(c, \gamma_2\right)\right).
\end{aligned}$$

The corresponding end sequent is $\Gamma \vdash \Delta$ for Γ and Δ as in Example 26.

Definition 47 (See Definition 31 of [LL18]) *Let S be a provable sequent, $\mathcal{G}$ a $\boldsymbol{S}\Pi_2$-**GT** with the nonterminals $\{\tau, \vec{\alpha}, \vec{\beta}_1, \ldots, \vec{\beta}_p\}$ as in Definition 45, and $\mathcal{S}$ the corresponding Π_2-**SEHST**. Let the sequent $\mathcal{S}[X \backslash \lambda\vec{x}\vec{y}.C]$ be a tautology where C may not contain any variable in $\vec{\alpha}$ or $\vec{\beta}_i$ with $i \in \mathbb{N}_p$. Then we call $[X \backslash \lambda\vec{x}\vec{y}.C]$ a* solution *of the Π_2-**SEHST** $\mathcal{S}$. With a slight abuse of language, we call C or $\lambda\vec{x}\vec{y}.C$ a solution.*

Example 28 (See Example 10 of [LL18]) *A solution of the Π_2-**SEHST** of Example 26 is*

$$[X \backslash \lambda x_1 x_2 y_1 y_2 . P\left(x_1, y_2\right) \vee Q\left(x_2, y_2\right)].$$

Obviously, Definition 47 generalizes the concepts of the previous sections. For $q_\forall = 1$ and $q_\exists = 1$, the generalization tallies with the method described in the Sections 4.2 – 4.4. Furthermore, the rest of the procedure has only to be adjusted to operate with tuples of variables. The *starting set with tuples of variables* of quantifiers now contains the designated variables $x_1, \ldots, x_{q_\forall}$ and $y_1, \ldots, y_{q_\exists}$ and may not contain variables of $\vec{\alpha}$ or $\vec{\beta}_j$ with $i \in \mathbb{N}_p$ (compare to Definition 32). The *set of possible sets of clauses with tuples of variables*, the *set of refined allowed clauses with tuples of variables*, and the *set of solution candidates with tuples of variables* can be defined accordingly.

The main theorem for the characterization generalizes to the case of blocks of quantifiers.

Theorem 18 (See Theorem 10 of [LL18]) *Let*

$$\mathcal{S} \stackrel{def}{=} [F[\vec{z}_1 \backslash \vec{t}_i]]_{i=1}^{l}, \bigvee_{k=1}^{q} X\vec{\alpha}\vec{r}_k \rightarrow \bigwedge_{k=1}^{p} X\vec{s}_k\vec{\beta}_k \vdash [G[\vec{z}_2 \backslash \vec{t}_j]]_{j=l+1}^{m}$$

*be a Π_2-**SEHST** corresponding to a Herbrand sequent S of a cut-free proof of $T =_{def} \forall \vec{z}_1 F \vdash \exists \vec{z}_2 G$ and a grammar $\mathcal{G}$ covering the Herbrand term set of S. Let $Sol(\mathcal{A}) \neq \emptyset$ be the set of solution candidates with tuples of variables for a given starting set with tuples of variables $\mathcal{A}$, and $\mathcal{C} \in Sol(\mathcal{A})$. Let $\mathcal{V}(\mathcal{C}) \subseteq \mathcal{V}(\vec{x}) \cup \mathcal{V}(\vec{y})$. Then there exists a proof of T with one cut and the cut formula $\forall \vec{x} \exists \vec{y} DNF(\mathcal{C})$.*

Moreover, we can easily adjust the unification method of Section 4.7 by replacing the production rules of Definition 40 with

$$\begin{aligned}
&\Pr^* \stackrel{def}{=} S_1 \cup S_2 \cup S_3 \text{ with} \\
S_1 &\stackrel{def}{=} \{\rho \mid \rho \text{ is a } F\text{-production or a } G\text{-production}\} \\
&\quad \text{(see Definition 30)}, \\
S_2 &\stackrel{def}{=} \{\vec{\alpha} \rightarrow \vec{x}, \vec{r}_1 \rightarrow \vec{x}, \ldots, \vec{r}_q \rightarrow \vec{x}\}, \text{ and} \\
S_3 &\stackrel{def}{=} \{\vec{s}_1\vec{\alpha} \rightarrow \vec{y}, \ldots, \vec{s}_p\vec{\alpha} \rightarrow \vec{y}, \vec{\beta}_1 \rightarrow \vec{y}, \ldots, \vec{\beta}_q \rightarrow \vec{y}\}.
\end{aligned}$$

Then we can prove the non-emptiness of the set of solution candidates with tuples of variables by assuming the existence of a balanced solution as in the simplified case.

Theorem 19 (See Theorem 11 of [LL18]) *Let $S =_{def} \forall \vec{z}_1 F \vdash \exists \vec{z}_2 G$, $\mathcal{G}$ be a $\boldsymbol{S}\Pi_2$-**GT**, and*

$$\mathcal{S} \stackrel{def}{=} [F[\vec{z}_1 \backslash \vec{t}_i]]_{i=1}^{l}, \bigvee_{k=1}^{q} X\vec{\alpha}\vec{r}_k \rightarrow \bigwedge_{k=1}^{p} X\vec{s}_k\vec{\beta}_k \vdash [G[\vec{z}_2 \backslash \vec{t}_j]]_{j=l+1}^{m}$$

*be a Π_2-**SEHST** for S and $\mathcal{G}$ with the reduced representation R. Assume that $\mathcal{S}$ has a balanced solution σ. Then the set of solution candidates with tuples of variables $Sol(\mathcal{U}(R))$ is not empty where $\mathcal{U}(R)$ is the starting set for the $\mathcal{G}^*$-unifiable sequent R with tuples (where we consider production rules for tuples).*

4.9 Proof Compression

In Section 4.7 we have defined a method to find balanced solutions for Π_2-**SEHS**. Here we demonstrate their potential of proof compression via Π_2 cuts. To this aim, we consider the following sequence of sequents. Let $n \geq 2$ be a natural number and

$$\begin{aligned}
A_n &\stackrel{def}{=} \forall x \left(P\left(x, f_1 x\right) \vee \ldots \vee P\left(x, f_n x\right)\right), \\
B &\stackrel{def}{=} \forall x, y \left(P\left(x, y\right) \rightarrow P\left(x, fy\right)\right), \\
Z_n &\stackrel{def}{=} P\left(x_1, fx_2\right) \wedge P\left(fx_2, fx_3\right) \wedge \ldots \wedge P\left(fx_{n-1}, fx_n\right), \\
C_n &\stackrel{def}{=} \forall x_1, \ldots, x_n \left(Z_n \rightarrow P\left(x_1, gx_n\right)\right), \text{ and} \\
D &\stackrel{def}{=} \exists x, y P\left(x, gy\right).
\end{aligned}$$

Then the sequents

$$S_n \stackrel{def}{=} A_n, B, C_n \vdash D \tag{4.8}$$

are provable. Note that the examples of the Sections 4.2-4.6 are often variants of some instance of S_n, mostly of S_2. The intuition behind the sequents stays the same: A_n guarantees witnesses for the second argument of P which can be used to find instantiations of Z_n. This becomes more apparent when we consider a proof with cut. For every variable x, we find a witness y such that $P\left(x, fy\right)$ is true. Since the sequents

$$\begin{array}{r}
A_n, B \vdash \forall x \exists y P\left(x, fy\right) \\
\forall x \exists y P\left(x, fy\right), C_n \vdash D
\end{array}$$

are provable, there is a proof with a single Π_2 cut

$$\dfrac{\dfrac{\vdots}{A_n, B, C_n \vdash \forall x \exists y P\left(x, fy\right), D} \quad \dfrac{\vdots}{A_n, B, C_n, \forall x \exists y P\left(x, fy\right) \vdash D}}{A_n, B, C_n \vdash D}$$

As will be shown below, a minimal cut-free proof of S_n will at least exponentially increase with respect to n whereas a proof with the displayed Π_2 cut will only linearly or polynomially increase. Furthermore, a sequence of $\mathbf{S}\Pi_2$-**G**s whose language covers the Herbrand term set of the cut-free proof is constructed. The corresponding sequence of Π_2-**SEHS**s is solvable, where the solution is for each instance $[X \backslash \lambda xy.P\left(x, fy\right)]$. We show that the method defined in Section 4.7 is able to find this solution since $P\left(x, fy\right)$ is a balanced solution.

Before we start to analyse the complexity of our example, we adjust the form of the end-sequents S_n. In the presented method we require a sequent of the form $\forall \vec{x} F \vdash \exists \vec{y} G$.

But as already mentioned we can transform each sequent into this format. Let A'_n, B', C'_n, and D' be the quantifier free part of A_n, B, C_n, and D (we rename the variables)

$$\begin{aligned}
A'_n &\stackrel{def}{=} P(x_1, f_1 x_1) \vee \ldots \vee P(x_1, f_n x_1), \\
B' &\stackrel{def}{=} P(x_2, x_3) \to P(x_2, f x_3), \\
C'_n &\stackrel{def}{=} (P(y_1, f y_2) \wedge P(f y_2, f y_3) \wedge \ldots \wedge P(f y_{n-1}, f y_n)) \to P(y_1, g y_n), \text{ and} \\
D' &\stackrel{def}{=} P(y_{n+1}, g y_{n+2}).
\end{aligned}$$

Furthermore, let $\vec{x} =_{def} (x_1, x_2, x_3)$ be the tuple of the three variables occurring in A'_n and B' and $\vec{y}_n =_{def} (y_1, \ldots, y_{n+2})$ be the tuples of the $n+2$ variables occurring in C'_n and D'. Let $\overline{C'_n}$ be the negation of C'_n, i.e.

$$\overline{C'_n} \stackrel{def}{=} (P(y_1, f y_2) \wedge P(f y_2, f y_3) \wedge \ldots \wedge P(f y_{n-1}, f y_n)) \wedge \neg P(y_1, g y_n).$$

Then we can define equivalent sequents

$$S'_n \stackrel{def}{=} \forall \vec{x}. A'_n \wedge B' \vdash \exists \vec{y}_n. \overline{C'_n} \vee D'. \tag{4.9}$$

From now on S'_n will always refer to the rewritten sequence of sequents that is in the correct form for the method presented in Section 4.7. S_n will refer to the original version.

4.9.1 Minimal Cut-Free Proofs

In this section we consider cut-free proofs of S_n for a fixed natural number n. For convenience we will compute lower bounds on the complexity of minimal proofs of S'_n in terms of complexity measurements defined below. In particular, we will show that minimal proofs of S_n always have a smaller complexity than minimal proofs of S'_n no matter which complexity measurement we choose. Moreover, by constructing minimal bounds for the complexity of minimal proofs of S_n we also construct minimal bounds for the complexity of minimal proofs of S'_n.

Lemma 7 (See Lemma 5 of [LL18]) *Let φ_n be a minimal proof of the sequent S_n (see Equation (4.8)) in terms of quantifier (see Definition 10), inference (see Definition 11), or symbol (see Definition 12) complexity and χ_n be a minimal proof of the sequent S'_n (see Equation (4.9)) in prenex normal form in terms of quantifier, inference, or symbol complexity, respectively then*

$$|\varphi_n|_\zeta \leq |\chi_n|_\zeta$$

where $\zeta \in \{i, q, s\}$.

Proof:
Each minimal proof of S_n can be transformed into a minimal proof of S'_n since all

instantiations of variables in φ_n are also required in χ_n. Therefore, the only difference is that in χ_n are additional inferences that correspond to the connectives in

$$\begin{array}{l} A'_n \wedge B', \\ \overline{C'_n} \vee D', \text{ and} \\ \overline{C'_n} \end{array}$$

which do not exist in φ_n. $\square$

In order to compute the complexities of minimal proofs of S_n we have to show some properties of minimal proofs. In a first step we show that in a minimal proof (with respect to an arbitrary complexity measurement) all atoms that appear in an instantiation of A_n, B, C_n, or D are active in an axiom. In a second step, we will show that every of the mentioned formulas is instantiated at least once. This allows us to define a procedure that constructs a minimal set of terms occurring in a minimal proof. The procedure then assumes an instantiation of A_n that implies an instantiation of B and of a part of C_n which then implies some instantiations of A_n and so on.

Lemma 8 (See Lemma 6 of [LL18]) *Let φ_n be a minimal proof in terms of quantifier, inference, or symbol complexity of the sequent S_n and*

$$\begin{array}{l} P\left(a, f_1 a\right) \vee \ldots \vee P\left(a, f_n a\right), \\ P\left(b_1, b_2\right) \rightarrow P\left(b_1, f b_2\right), \\ \left(P\left(c_1, f c_2\right) \wedge P\left(f c_2, f c_3\right) \wedge \ldots \wedge P\left(f c_{n-1}, f c_n\right)\right) \rightarrow P\left(c_1, g c_n\right), \\ P\left(d_1, g d_2\right) \end{array}$$

be instantiations of A_n, B, C_n, and D for some proof-specific terms a, b_1, b_2, c_1, …, c_n, d_1, and d_2. Then there are axioms for each atom

$$\begin{array}{l} P\left(a, f_1 a\right), \ldots, P\left(a, f_n a\right), P\left(b_1, b_2\right), P\left(b_1, f b_2\right), \\ P\left(c_1, f c_2\right), P\left(f c_2, f c_3\right), \ldots, P\left(f c_{n-1}, f c_n\right), P\left(c_1, g c_n\right), \text{ and} \\ P\left(d_1, g d_2\right) \end{array}$$

in which the respective atom is active.

Proof:
The proof works for all four formulas in a similar way. We will only consider the formula

$$A_n^a \stackrel{def}{=} P\left(a, f_1 a\right) \vee \ldots \vee P\left(a, f_n a\right).$$

Assume there is an $i \in \mathbb{N}_n$ such that $P\left(a, f_i a\right)$ is not active in any axiom. Then we can order φ_n such that the $l\colon\vee$-rules that apply to A_n^a are the rules in the new minimal proof

χ that appear at the top of the corresponding proof tree. Let $A_n^a, \Gamma \vdash \Delta$ be a sequent in χ in which A_n^a appears. The provability implies that also $P(a, f_i a), \Gamma \vdash \Delta$ is provable. Hence, $\Gamma \vdash \Delta$ is already tautological and we can drop all the $l:\vee$-rules applied to A_n^a (and even the instantiation rules). Thus, there is a proof with smaller quantifier, inference, and symbol complexity which contradicts the assumption that φ_n was already minimal in these terms. Hence, there is no such instantiation. □

Remark 5 (See Remark 4 of [LL18]) *Note that Lemma 8 does not describe a general property of minimal proofs. Consider, for instance, the proof*

$$\cfrac{\cfrac{\cfrac{}{\forall x.P(x) \wedge Q(x), P(a), Q(a) \vdash P(a)}\,Ax}{\forall x.P(x) \wedge Q(x), P(a) \wedge Q(a) \vdash P(a)}\,l:\wedge}{\forall x.P(x) \wedge Q(x) \vdash P(a)}\,l:\forall$$

of the sequent $\forall x.P(x) \wedge Q(x) \vdash P(a)$. The proof is minimal but the atom $Q(a)$ is not active.

In Lemma 8, we showed that for each instantiated formula A_n, B, C_n, or D the corresponding atoms have to be active in some axiom. Now, we show that each of these formulas are instantiated at least once.

Lemma 9 (See Lemma 7 of [LL18]) *Let φ_n be a proof of the sequent S_n. Then the formulas*

$$\begin{aligned}
&P(a, f_1 a) \vee \ldots \vee P(a, f_n a),\\
&P(b_1, b_2) \rightarrow P(b_1, f b_2), \text{ and}\\
&(P(c_1, f c_2) \wedge P(f c_2, f c_3) \wedge \ldots \wedge P(f c_{n-1}, f c_n)) \rightarrow P(c_1, g c_n)
\end{aligned}$$

with some proof-specific terms a, b_1, b_2, c_1, …, c_n appear on the left side of some sequents in φ_n and the formula

$$P(d_1, g d_2)$$

with some proof-specific terms d_1, d_2 appears on the right side of some sequent φ_n.

Proof:
Note that at least one formula has to be instantiated. Otherwise, there cannot be a valid proof. By showing that an instantiation of an arbitrary formula enforces all other formulas to be instantiated at least once we will complete the proof. This can easily be seen by Lemma 8 and the facts that all potential atoms of A_n can only build valid axioms with potential atoms of B ($\Gamma, P(a, f_i a) \vdash P(b_1, b_2), \Delta$ with $a = b_1$ and $f_i a = b_2$),

all potential atoms of B have to build axioms with A_n and C_n, and so on. In the end, we have to instantiate A_n, B, C_n, and D. $\square$

Now we can describe sets of instantiations that belong to a minimal proof of S_n. We will not write down the whole proof because of its large size. But by proving the minimality of the number of instantiations we will implicitly give a sketch of the proof and show its validity.

Lemma 10 (See Theorem 13 of [LL18]) *Let $n \geq 2$. Then the sets*

$$\begin{aligned}
\mathbf{A}_n^1 &\stackrel{def}{=} \{a\},\\
\mathbf{A}_n^2 &\stackrel{def}{=} \{fh_1a \mid h_1 \in \{f_1, \ldots, f_n\}\},\\
\mathbf{A}_n^i &\stackrel{def}{=} \{fh_{i-1} \ldots fh_1a \mid h_1, \ldots, h_{i-1} \in \{f_1, \ldots, f_n\}\},\\
\mathbf{A}'_n &\stackrel{def}{=} \bigcup_{i=1}^{n-1} \mathbf{A}_n^i,\\
\mathbf{B}' &\stackrel{def}{=} \{(r, f_ir) \mid r \in \mathbf{A}'_n \wedge i \in \mathbb{N}_n\},\\
\mathbf{C}'_n &\stackrel{def}{=} \{ (r_1, \ldots, r_n, r_1, r_n) \mid r_1 = a \wedge r_2 = h_1r_1 \wedge r_3 = h_2fr_2 \wedge \ldots\\
&\qquad \ldots \wedge r_n = h_{n-1}fr_{n-1} \wedge h_1, \ldots, h_{n-1} \in \{f_1, \ldots, f_n\} \}, \text{ and}\\
\mathbf{D}' &\stackrel{def}{=} \{(a, r) \mid r = r_n \wedge \exists r_1, \ldots, r_{n-1}\, (r_1, \ldots, r_{n-1}, r_n) \in \mathbf{C}'_n\}
\end{aligned}$$

are instantiations of the formulas A_n, B, C_n, and D such that the corresponding fully instantiated sequent $S_n^{\downarrow}$ is tautological and there is a minimal (in terms of quantifier, inference, or symbol complexity) proof φ_n of S_n with the midsequent $S_n^{\downarrow}$.

Proof:
By Lemma 9 we can assume an instantiation $(r_1, \ldots, r_n)$ of C_n. Let $a =_{def} r_1$. As atomic subformulas of an instantiated formula in a minimal proof have to be active (see Lemma 8) we know that $P(a, fr_2)$ of

$$(P(a, fr_2) \wedge P(fr_2, fr_3) \wedge \ldots \wedge P(fr_{n-1}, fr_n)) \rightarrow P(a, gr_n)$$

has to be active in an axiom. In an axiom, $P(a, fr_2)$ appears on the right side of the sequent and hence, the only formula that can become $P(a, fr_2)$ on the left side of the sequent is $P(x, fy)$ of

$$B = \forall x, y\, (P(x, y) \rightarrow P(x, fy)).$$

Then x has to be equal to a and y has to be equal to r_2. By applying Lemma 8 again we have to find the counterpart of $P(x, y) = P(a, r_2)$. Hence, there has to be an instantiation of

$$A_n = \forall x\, (P(x, f_1x) \vee \ldots \vee P(x, f_nx)),$$

i.e.

$$P(a, f_1 a) \vee \ldots \vee P(a, f_n a).$$

Given that this is the only possibility we can conclude that there have to be instantiations of B where x is equal to a and y is equal once to $f_1 a$, ..., once to $f_{n-1} a$, and once to $f_n a$ and instantiations of

$$C_n = \forall x_1, \ldots, x_n (Z_n \rightarrow P(x_1, g x_n))$$
$$Z_n = P(x_1, f x_2) \wedge P(f x_2, f x_3) \wedge \ldots \wedge P(f x_{n-1}, f x_n)$$

such that $x_1 = a$ and x_2 takes the same n terms as y in B. The term r_2 of the first assumed instantiation might actually be one of these instantiations, i.e. it might be $f_1 a$, ..., $f_{n-1} a$, or $f_n a$.

So far we described $\mathbf{A}_n^1$, the parts of $\mathbf{B}'$ where $\mathbf{A}'_n$ is replaced with $\mathbf{A}_n^1$, the first two elements of the tuples in $\mathbf{C}'_n$, and the first element of the tuples in $\mathbf{D}'$. With the second elements $f_1 a, \ldots, f_n a$ of the tuples in $\mathbf{C}'_n$ we have to go through the same procedure as we did with a. That is, we will get new instantiations of A_n, i.e. $\mathbf{A}_n^2$, a new part of $\mathbf{B}'$ and the third elements of the tuples in $\mathbf{C}'_n$. After n applications of this procedure, the sets of the statement are fully constructed. Moreover, all atoms of the instantiated formulas appear as an active formula in an axiom and we cannot drop a single instantiation without making the proof invalid. Thus, the instantiations correspond to a minimal proof of S_n in terms of quantifier complexity. As all proofs contain at least as many instantiations as the given one there also has to be a corresponding minimal proof in terms of inference and symbol complexity. □

Finally, we can compute a lower bound for the various complexities of a minimal cut-free proof of S'_n. The lower bound is guaranteed by the minimal number of terms occurring in a cut-free proof.

Theorem 20 *Let φ_n be a minimal cut-free proof of S'_n. Then*

$$|\varphi_n|_\zeta \geq n^n$$

for $\zeta \in \{i, q, s\}$.

Proof:
In Lemma 7, we proved that every minimal cut-free proof of S'_n is larger with respect to all the considered complexity measures in comparison to a minimal cut-free proof χ of S_n. Hence, we only have to show that

$$|\chi|_\zeta \geq n^n$$

for $\zeta \in \{i, q, s\}$. By Proposition 2, we get

$$|\chi|_q \geq n^n$$

implies

$$|\chi|_i \geq n^n \text{ and } |\chi|_s \geq n^n.$$

Therefore, we consider the sets of instantiations of Lemma 10 and the corresponding Herbrand sequent $S_n^{\downarrow}$:

$$|S_n^{\downarrow}| = \sharp\mathbf{A}'_n + \sharp\mathbf{B}' + \sharp\mathbf{C}'_n + \sharp\mathbf{D}'.$$

By

$$\begin{aligned}
\sharp\mathbf{A}'_n &= \sum_{i=1}^{n} n^{i-1}, \\
\sharp\mathbf{B}' &= (n+1) \cdot \sum_{i=1}^{n} n^{i-1}, \\
\sharp\mathbf{C}'_n &= n^{n-1} + \sum_{i=1}^{n} n^{i-1}, \text{ and} \\
\sharp\mathbf{D}' &= n^{n-1} + 1
\end{aligned}$$

the complexity sums up to

$$|S_n^{\downarrow}| = n^n + 6 \cdot n^{n-1} + 4 \cdot n^{n-2} + \ldots + 4 \cdot n + 5 > n^n$$

for $n \geq 3$ and

$$|S_n^{\downarrow}| = n^n + 6 \cdot n^{n-1} + 5 > n^n$$

for $n = 2$. In any case, it follows that

$$|\chi|_\zeta \geq n^n$$

and thereby

$$|\varphi_n|_\zeta \geq n^n$$

for $\zeta \in \{i, q, s\}$. $\square$

4.9.2 A Proof Schema with a Π_2 Cut

After computing a lower bound for the complexity of a minimal cut-free proof of S'_n for a fixed n we want to generate a cut formula by the method of Section 4.7 and analyse the complexity of the corresponding proof schema with cut. Therefore, we define a schema of $\mathbf{S}\Pi_2$-**G**s $\mathcal{G}_n$. Following the intuition described in the beginning of this section, the end sequent is represented with two terms: with h_{F_n} and with h_{G_n}. The first part contains the eigenvariable/nonterminal α and will be used to prove the cut formula and the second part contains the eigenvariables/nonterminals $\beta_1, \ldots, \beta_{n-1}$ and will be proven by the cut formula.

Definition 48 (Schema of SΠ_2-Gs) *Let*

$$N_n \stackrel{def}{=} \{\tau, \alpha, \beta_1, \ldots, \beta_n\}$$

and

$$\mathrm{Pr}_n \stackrel{def}{=} \begin{array}{ll} \{ & \tau \to h_{F_n}(\alpha, \alpha, f_1\alpha) \mid \ldots \mid h_{F_n}(\alpha, \alpha, f_n\alpha), \\ & \tau \to h_{G_n}(a, \beta_1, \ldots, \beta_{n-1}, a, \beta_{n-1}), \\ & \alpha \to f\beta_{n-1} \mid \ldots \mid f\beta_1 \mid a, \\ & \beta_{n-1} \to f_1 f\beta_{n-2} \mid \ldots \mid f_n f\beta_{n-2}, \\ & \quad \vdots \\ & \beta_2 \to f_1 f\beta_1 \mid \ldots \mid f_n f\beta_1, \\ & \beta_1 \to f_1 a \mid \ldots \mid f_n a \qquad \} \end{array}$$

where h_{F_n} and h_{G_n} are function symbols that correspond to the λ-terms $\lambda\vec{x}.A'_n \wedge B'$ and $\lambda\vec{y}.C'_n \vee D'$, respectively. We define the schema of grammars $\mathcal{G}_n =_{def} \langle \tau, N_n, \Sigma_n, \mathrm{Pr}_n \rangle$.

Remark 6 *Note that the language $\mathcal{L}(\mathcal{G}_n)$ of $\mathcal{G}_n$ for a fixed number n covers the Herbrand term set of $S_n^{\downarrow}$ of the previous section.*

Definition 49 (Schema of Π_2-SEHSs) *Let n be a fixed number and $\mathcal{G}_n$ as in Definition 48. Then*

$$\mathcal{S}_n \stackrel{def}{=} \begin{array}{c} [(\lambda\vec{x}.A'_n \wedge B')\, \alpha\alpha f_i\alpha]_{i=1}^n, \\ \bigvee_{k=1}^{n} X\alpha f_k\alpha \to Xfa\beta_1 \wedge \bigwedge_{k=2}^{n-1} Xf\beta_{k-1}\beta_k \vdash (\lambda\vec{y}.C'_n \vee D')\, a\beta_1 \ldots \beta_{n-1} a\beta_{n-1} \end{array}$$

*is a Π_2-**SEHS** for S'_n with the reduced representation*

$$R_n \stackrel{def}{=} [(\lambda\vec{x}.A'_n \wedge B')\, \alpha\alpha f_i\alpha]_{i=1}^n \vdash (\lambda\vec{y}.C'_n \vee D')\, a\beta_1 \ldots \beta_{n-1} a\beta_{n-1}.$$

The Definitions 48 and 49 describe a problem to which we can apply the method of Section 4.7 (see also Subsection 5.3.1). First, we apply a maximal **G3c** derivation φ_n to

the reduced representation R_n. Then we collect all leaves

$$\begin{array}{l}\{\ P\left(\alpha, h\alpha\right), \\ \quad \{\neg P\left(\alpha, f_i\alpha\right)\ \mid i \in S\}, \{P\left(\alpha, ff_i\alpha\right)\ \mid i \in \mathbb{N}_n \backslash S\}, \\ \quad \{\neg P\left(a, f\beta_1\right)\ \mid j = 1\}, \\ \quad \{\neg P\left(f\beta_1, f\beta_2\right)\ \mid j = 2\}, \ldots, \{\neg P\left(f\beta_{n-2}, f\beta_{n-1}\right)\ \mid j = n-1\}, \\ \quad \{P\left(a, g\beta_{n-1}\right)\ \mid j = n\}, \\ \quad \neg P\left(a, g\beta_{n-1}\right) \vdash \\ \quad \mid h \in \{f_1, \ldots, f_n\}, S \subseteq \mathbb{N}_n, j \in \mathbb{N}_n \qquad\qquad \}\end{array}$$

and all non-tautological ones

$$\begin{array}{rl} & \{\ P\left(\alpha, f_k\alpha\right), \\ & \quad \{\neg P\left(\alpha, f_i\alpha\right)\ \mid i \in S\}, P\left(\alpha, ff_k\alpha\right), \{P\left(\alpha, ff_i\alpha\right)\ \mid i \in \mathbb{N}_n \backslash \left(S \cup \{k\}\right)\}, \\ \mathit{DNTA}\left(\varphi_n\right) = & \quad \{\neg P\left(a, f\beta_1\right)\ \mid j = 1\}, \\ & \quad \{\neg P\left(f\beta_1, f\beta_2\right)\ \mid j = 2\}, \ldots, \{\neg P\left(f\beta_{n-2}, f\beta_{n-1}\right)\ \mid j = n-1\}, \\ & \quad \neg P\left(a, g\beta_{n-1}\right) \vdash \\ & \quad \mid k \in \mathbb{N}_n, S \subseteq \left(\mathbb{N}_n \backslash \{k\}\right), j \in \mathbb{N}_{n-1} \qquad\qquad \}. \end{array}$$

In each leaf there is a least one conjunct of A'_n (first line of $\mathit{DNTA}\left(\varphi_n\right)$). Hence, if we branch B' with the corresponding term of the chosen disjunct only the branch containing the succedent of this B' is not a tautology (second line of $\mathit{DNTA}\left(\varphi_n\right)$). Given that each leaf contains the instantiation of D' (fifth line of $\mathit{DNTA}\left(\varphi_n\right)$) we have to look at the branch containing the antecedent of the instantiation of C'_n. Otherwise the leaf is a tautology. The antecedent is a conjunction that moves to the right of the sequent after branching C'_n and therefore, we have to pick an arbitrary conjunct (third and fourth line of $\mathit{DNTA}\left(\varphi_n\right)$).

We will not construct the whole set of unification candidates because most of them are obviously not $\mathcal{G}^*$-unifiable. The only interactive literals are $P\left(\alpha, ff_i\alpha\right)$ and $\neg P\left(a, f\beta_1\right)$ or $P\left(\alpha, ff_i\alpha\right)$ and $\neg P\left(f\beta_j, f\beta_{j+1}\right)$ with $i \in \mathbb{N}_n$, $j \in \mathbb{N}_{n-2}$. The maximal set of $\mathcal{G}^*$-unified literals is independent from the chosen non-tautological leaf and consists of the single literal $P\left(x, fy\right)$. Then the starting set for $\mathcal{G}^*$-unifiable sequents is

$$\mathcal{U}(R_n) = \{S \mid S \subseteq \{P\left(x, fy\right)\}\} = \{\emptyset, P\left(x, fy\right)\}$$

accordingly. We can ignore the empty set because R_n is not a tautology. The set of possible sets of clauses (see Definition 35) is $Cl(\mathcal{U}(R_n)) = \{\{P\left(x, fy\right)\}\}$ and the set of solution candidates (see Definition 37) is

$$Sol(\mathcal{U}(R_n)) = \{\{P\left(x, fy\right)\}\}.$$

Figure 4.11: Sketch of a schema of proofs with Π_2 cut

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{\vdots}{\Gamma, \left[\left(\lambda\vec{x}.A'_n \wedge B'\right)\alpha\alpha f_i\alpha\right]_{i=1}^{n} \vdash \left[P\left(\alpha, f f_i\alpha\right)\right]_{i=1}^{n}, \exists y P\left(\alpha, fy\right), \exists\vec{y}.\overline{C'_n} \vee D'}\, l:\forall
}{\forall\vec{x}.A'_n \wedge B' \vdash \left[P\left(\alpha, f f_i\alpha\right)\right]_{i=1}^{n}, \exists y P\left(\alpha, fy\right), \exists\vec{y}.\overline{C'_n} \vee D'}\, r:\exists
}{\forall\vec{x}.A'_n \wedge B' \vdash \forall x \exists y P\left(x, fy\right), \exists\vec{y}.\overline{C'_n} \vee D'}\, r:\forall
}{\forall\vec{x}.A'_n \wedge B' \vdash \exists\vec{y}.\overline{C'_n} \vee D'}\, Cut \qquad \chi
$$

with

$$
\chi \overset{def}{=}
\dfrac{
\dfrac{
\dfrac{
\dfrac{\Gamma', P\left(a, f\beta_1\right), \left[P\left(f\beta_i, f\beta_{i+1}\right)\right]_{i=1}^{n-2} \vdash \left(\lambda\vec{y}.C'_n \vee D'\right) a\beta_1 \ldots \beta_{n-1} a\beta_{n-1}, \Delta}{\vdots}\, r:\exists
}{\Gamma', P\left(a, f\beta_1\right), \left[P\left(f\beta_i, f\beta_{i+1}\right)\right]_{i=1}^{n-2} \vdash \exists\vec{y}.\overline{C'_n} \vee D'}\, r:\exists
}{\vdots}\, l:\exists
}{\forall\vec{x}.A'_n \wedge B', \forall x \exists y P\left(x, fy\right) \vdash \exists\vec{y}.\overline{C'_n} \vee D'}\, l:\forall
$$

where Γ, Γ', and Δ are redundant formulas created during the process of instantiating the quantifiers.

Note that the set of solution candidates is independent from n. By Theorem 15 follows that $P\left(x, fy\right)$ is a solution and by Theorem 16 we find a proof with a single Π_2 cut that is $\forall x \exists y P\left(x, fy\right)$. Then we can define a proof of S'_n as sketched in Figure 4.11.

In order to check its correctness we investigate the two sequents

$$
\begin{aligned}
\left[\left(\lambda\vec{x}.A'_n \wedge B'\right)\alpha\alpha f_i\alpha\right]_{i=1}^{n} &\vdash \left[P\left(\alpha, f f_i\alpha\right)\right]_{i=1}^{n} \text{ and} \\
P\left(a, f\beta_1\right), \left[P\left(f\beta_i, f\beta_{i+1}\right)\right]_{i=1}^{n-2} &\vdash \left(\lambda\vec{y}.C'_n \vee D'\right) a\beta_1 \ldots \beta_{n-1} a\beta_{n-1}.
\end{aligned}
$$

This corresponds to the α-problem and the β-problem. Note that $DNTA\left(\varphi_n\right)$ can be divided into two sets that are the non-tautological leaves of maximal derivations of $\left[\left(\lambda\vec{x}.A'_n \wedge B'\right)\alpha\alpha f_i\alpha\right]_{i=1}^{n} \vdash$ and $\vdash \left(\lambda\vec{y}.C'_n \vee D'\right) a\beta_1 \ldots \beta_{n-1} a\beta_{n-1}$. More precisely:

$$
\begin{aligned}
\{\ & P\left(\alpha, f_k\alpha\right), \\
& \{\neg P\left(\alpha, f_i\alpha\right) \mid i \in S\}, P\left(\alpha, f f_k\alpha\right), \{P\left(\alpha, f f_i\alpha\right) \mid i \in \mathbb{N}_n \backslash \left(S \cup \{k\}\right)\} \\
& \mid k \in \mathbb{N}_n, S \subseteq \left(\mathbb{N}_n \backslash \{k\}\right) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \}
\end{aligned}
$$

is the set of non-tautological leaves of the first sequent and

$$\begin{aligned}
\{ \; & \{\neg P(a, f\beta_1) \mid j = 1\}, \\
& \{\neg P(f\beta_1, f\beta_2) \mid j = 2\}, \ldots, \{\neg P(f\beta_{n-2}, f\beta_{n-1}) \mid j = n-1\}, \\
& \neg P(a, g\beta_{n-1}) \vdash \\
& \mid j \in \mathbb{N}_{n-1} \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \}.
\end{aligned}$$

is the set of non-tautological leaves of the second sequent. The first set always contains a literal of the form $P(\alpha, ff_k\alpha)$ with $k \in \mathbb{N}_n$ that occurs also in $[P(\alpha, ff_i\alpha)]_{i=1}^n$. Therefore,

$$[(\lambda\vec{x}.A'_n \wedge B')\,\alpha\alpha f_i\alpha]_{i=1}^n \vdash [P(\alpha, ff_i\alpha)]_{i=1}^n$$

is a tautology. Moreover, one of the literals of $P(a, f\beta_1), [P(f\beta_i, f\beta_{i+1})]_{i=1}^{n-2}$ occurs in the second set of non-tautological leaves proving that

$$P(a, f\beta_1), [P(f\beta_i, f\beta_{i+1})]_{i=1}^{n-2} \vdash (\lambda\vec{y}.C'_n \vee D')\,a\beta_1 \ldots \beta_{n-1}a\beta_{n-1}$$

is a tautology. Since the considered sequents are subsequents of the sequents occurring in the proof sketch with Π_2 cut, we approve the correctness of the proof.

Finally, we can compute the size of a proof with a single cut and the cut-formula $\forall x\exists y P(x, fy)$.

Theorem 21 *Let n be a natural number, $\mathcal{G}_n$ be as in Definition 48, and $\mathcal{S}_n$ be as in Definition 49. Then there is a solution of $\mathcal{S}_n$ such that the corresponding proof φ'_n fulfils the following properties:*

$$\begin{aligned}
|\varphi'_n|_q &= 4 \cdot n + 3, \\
|\varphi'_n|_i &= 9 \cdot n + 3, \\
|\varphi'_n|_s &\leq 2 \cdot p(n) \cdot |\varphi'_n|_i + |\varphi'_n|_i
\end{aligned}$$

where $p(\cdot)$ is a polynomial.

Proof:
First we count the number of instantiations in the proof φ'_n that was sketched above. Let $|A'_n \wedge B'|$ be the number of instantiations of $\forall\vec{x}A'_n \wedge B'$ of the left-hand side of the end sequent S'_n and let $|C'_n \vee D'|$ be the number of instantiations of the right-hand side of the end sequent S'_n. Then

$$\begin{aligned}
|A'_n \wedge B'| &= n + 2 \text{ and} \\
|C'_n \vee D'| &= n + 2.
\end{aligned}$$

The number of weak quantifier inferences applied to the cut formula or one of its ancestors is $2 \cdot n - 1$. Hence, the number of weak quantifier inferences is

$$\begin{aligned} & 1 \cdot n + 2 \\ + \; & 1 \cdot n + 2 \\ + \; & 2 \cdot n - 1 \\ = \; & 4 \cdot n + 3 \end{aligned}$$

proving the first property. For the second property, we count the separate **G3c**-rules, occurring in the proof, independently. Let $|\varphi|_\zeta$ with ζ being the label of a **G3c**-rule be the number of appearances of ζ in φ. Then

$$\begin{aligned} |\varphi'_n|_{l:\forall} + |\varphi'_n|_{r:\exists} &= 4 \cdot n + 3, & |\varphi'_n|_{r:\vee} &= 1, \\ |\varphi'_n|_{r:\forall} &= 1, & |\varphi'_n|_{l:\to} &= n, \\ |\varphi'_n|_{l:\exists} &= n - 1, & |\varphi'_n|_{r:\to} &= 0, \\ |\varphi'_n|_{l:\wedge} &= n, & |\varphi'_n|_{l:\neg} &= 0, \\ |\varphi'_n|_{r:\wedge} &= n - 1, & & \text{and} \\ |\varphi'_n|_{l:\vee} &= n - 1, & |\varphi'_n|_{r:\neg} &= 1 \end{aligned}$$

and the number of inferences in φ'_n sums up to

$$|\varphi'_n|_i = 9 \cdot n + 3.$$

To give an upper bound on the symbol complexity we have to compute the maximal symbol complexity of the sequents appearing in the proof φ'_n. Therefore, we will assume a polynomial $p(\cdot)$ of one argument such that the maximal size of each sequent in the proof is smaller than $p(n)$. The existence of such a function can easily be proven. Note that, considering the sequents in the premises and in the conclusion of a **G3c**-rule, only in the weak quantifier rules the symbol complexity of the conclusion is smaller than the symbol complexity of one of the premises. Therefore, the maximal symbol complexity of a sequent occurring in φ'_n can be bounded by the symbol complexity of a sequent containing $\forall \vec{x} A'_n \wedge B', \forall x \exists y P(x, fy) \vdash \forall x \exists y P(x, fy), \exists \vec{y} C'_n \vee D'$ and all of its $|\varphi'_n|_{l:\forall} + |\varphi'_n|_{r:\exists} + |\varphi'_n|_{r:\forall} + |\varphi'_n|_{l:\exists}$ instantiations occurring in φ'_n. Thus, the degree of $p(\cdot)$ is 2. Given $p(\cdot)$ we can define the upper polynomial bound

$$2 \cdot p(n) \cdot |\varphi'_n|_i + |\varphi'_n|_i,$$

i.e. for each of the $|\varphi'_n|_i$ different **G3c**-rules there are at most 2 sequents as premises plus the rules itself. □

While the complexity in terms of logical inferences, in terms of weak quantifier inferences, or in terms of symbol complexity is bigger than n^n for the cut free proofs the introduction of the Π_2 cut decreases the complexity by an exponential factor.

CHAPTER 5

Implementation and Experiments

The method described in Section 4.7 is implemented in the **GAPT** (**G**eneral **A**rchitecture for **P**roof **T**heory) framework that is a collection of data structures, algorithms, parsers, and other components common in proof theory and automated deduction [EHR+16]. The implementation can be found in all releases after version 2.5. **GAPT** is implemented in Scala, licensed under the GNU General Public License, and available at https://logic.at/gapt. At the web page of the **GAPT** system, a user manual is offered. The code can also be found at the GitHub repository https://github.com/gapt/gapt.

In this chapter, we will discuss the implementation of the unification method (Section 5.1), an algorithm for the construction of $\mathbf{S}\Pi_2$-**G**s (Section 5.2), and experiments (Section 5.3). In each section, we consider version 2.9. of **GAPT**, the current one at the time of writing.

5.1 An Implementation for the Construction of Π_2-Cut Formulas

In this section, we describe the most important parts of the implementation and necessary background of the underlying **GAPT** system. We compare at first the input of the `proveWithPi2Cut` method with Π_2-**SEHS**s and $\mathbf{S}\Pi_2$-**G**s and explain afterwards the `gStarUnify` method as well as the `LiteralWithIndexLists` class in more detail. The description touches only the computation of a cut formula. For information about the generation of $\mathbf{S}\Pi_2$-**G**s see Section 5.2.

5.1.1 The Pi2SeHs Type

The main function of the program is the

```
proveWithPi2Cut( endSequent: Sequent[Formula],
```

```
seHs: Pi2SeHs,
y: Var, x: Var ): ( Option[LKProof] )
```

method. The `endSequent` object is of type `Sequent[Formula]` where `Formula` is a defined type of the **GAPT** system for the representation of formulas. Furthermore, `Sequent[A]` is also a defined type of the **GAPT** system that represents a sequent of objects of type `A`. Therefore, `endSequent` is just a representation of a provable sequent for which we want to find a proof with a Π_2 cut corresponding to the sequent S of Definition 31.

New to the system is the `Pi2SeHs` class. It collects the information of a single $\mathbf{S\Pi_2\text{-}G}$ $\mathcal{G}$ (see Definition 29) together with the reduced representation and defines several functions for it. In addition to `endSequent`, the two classes contain all information of a Π_2-**SEHS**. In order to define a new `Pi2SeHs` object we require

- the reduced representation R,
- the universal eigenvariable α,
- the list of the existential eigenvariables $\beta_1, \ldots, \beta_p$ (starting with the smallest index according to Definition 29, i.e. with the β_i not occurring in the production rule of another β_j),
- the list of terms $s_1, \ldots, s_p$ according to Definition 29, and
- the list of terms $r_1, \ldots, r_q$ with $\alpha \in \mathcal{V}(r_i)$ for $i \in \mathbb{N}_q$ according to Definition 29.

The last two arguments of the `proveWithPi2Cut` method are optional. They allow the user to name the quantified variables of the potential cut formula. The default values are `xCut` and `yCut` unless those names are occupied. If they are already used or if the names the user chose are already used, the algorithm will give them fresh similar names.

5.1.2 The Structure of the Algorithm

The `proveWithPi2Cut` method consists only of a call to the `introducePi2Cut` method and a proof building part where a proof with cut is constructed if a cut formula has been found. The `introducePi2Cut` method calls the `gStarUnify` method to compute a set of literals which are used to construct potential cut formulas. In this construction process, the `LiteralWithIndexLists` class is heavily used to check whether a valid cut formula was found. In the following, we describe the steps of the `introducePi2Cut` method.

introducePi2Cut

The input of the `introducePi2Cut` method is an object of type `Pi2SeHs` and two names for the variables in the cut formula. As the theory suggests, `Pi2SeHs`-objects allow us the computation of

- the $\forall$-multiplicity and the $\exists$-multiplicity,
- all substitutions for the cut formula,
- the production rules of the $\mathbf{S\Pi_2}$-**G**,
- the *DNTA* (φ) of a maximal **G3c**-derivation φ of the reduced representation, and
- the set of all literals occurring in *DNTA* (φ).

There is also a method to compute a Herbrand sequent. Note that, we store only non-tautological leaves if they are not a superset of another non-tautological leaf (Every solution of the subset is also a solution of the superset).

In a first step, `introducePi2Cut` calls the `gStarUnify` method asking for a set I of $\mathcal{G}^*$-unified literals corresponding to the $\mathbf{S\Pi_2}$-**G** $\mathcal{G}$ defined by the `Pi2SeHs` object. We postpone the discussion of `gStarUnify` to the next section and concentrate on the `introducePi2Cut` method.

In a next step, we compute index lists for each literal of I. They consist of pairs of numbers representing an index of a substitution and an index of a non-tautological leaf S such that the literal with this specific substitution makes S tautological. A list of the form $\{(2,3),(2,4)\}$ states that the leaf number 2 is tautological if we add the literal substituted with the substitution represented by 3 and 4. In the implementation the list is stored as $\{(2,(3,4))\}$. Moreover, we distinguish between two index lists: the index list with substitutions containing the eigenvariable α and the index list with substitutions containing some of the eigenvariables $\beta_1,\ldots,\beta_p$. Hence, one type of index list only considers the α-problem while the other type considers the β-problem.

Assume a literal L with the index lists I and J where

- I considers the α-problem,
- I contains a pair (i,S) with non empty S for each leaf i,
- J considers the β-problem, and
- J contains a pair (i,S) with non empty S for each leaf i.

Obviously, L itself can be used to construct a cut formula since it solves the α-problem and the β-problem. Usually, such a literal does not exist and we have to construct

clauses out of literals and sets of clauses out of clauses. The benefit of computing all the index lists is that we can directly compute whether a new constructed clause or set of clauses is a solution. Assume the two literals L_1, L_2 with the index lists I_1, J_1 and I_2, J_2, respectively where

- I_1, I_2 consider the α-problem,
- I_1 only contains a pair for the leaf with number 3, i.e. $I_1 = \{(3, S)\}$ with non empty S,
- I_2 only contains a pair for the leaf with number 4, i.e. $I_2 = \{(4, S)\}$ with non empty S,
- J_1, J_2 consider the β-problem,
- J_1 only contains a pair for the leaf with number 3, i.e. $J_1 = \{(3, S)\}$ with non empty S, and
- J_2 contains a pair (i, S) with non empty S for each leaf i.

Then we can assign new index lists I', J' to the clause $L_1 \wedge L_2$. On one hand, the clause makes a leaf of the β-problem tautological if one of the literals made it already tautological, i.e. $J' = J_1 \cup J_2$. In the given example, J' contains for each leaf i a pair (i, S) with non empty S. On the other hand, the clause only makes a leaf of the α-problem tautological if both literals made it already tautological, i.e. $I' = I_1 \cap I_2$. In the given example, I' is empty. In summary, we intersect the index lists for the α-problem and union the index lists for the β-problem in the construction procedure of a new clause.

When constructing sets of clauses, we can again assign new index lists. For the same reasons, we union the index lists for the α-problem and intersect the index lists for the β-problem in the construction procedure of a new clause, i.e. we exchange intersections with unions and vice versa. This is due to the symmetric behaviour of $r : \wedge$ and $l : \vee$ as well as $r : \vee$ and $l : \wedge$.

As soon as a set of clauses, a clause, or a literal is found with index lists that contains a pair (i, S) with non empty S for each non-tautological leaf i, the method terminates and outputs this formula with renamed variables.

gStarUnify

One of the major contributions of the paper [LL18] was the introduction of the so called $\mathcal{G}^*$-unifiability. When we consider cut-introduction in general, the only restriction to a potential cut formula is given by the signature. The $\mathcal{G}^*$-unifiability gives us a method at hand that yields a small number of literals that might occur in the cut formula. Thereby, we lose the universality of the solution, i.e. we will only find a cut formula if there is a balanced solution.

In the `gStarUnify` method, we begin by partitioning all literals occurring in the non-tautological leaves of the reduced representation in the sets `alphaPos`, `alphaNeg`, `betaPos`, `betaNeg`, `neutralPos`, and `neutralNeg`. As the names suggest, they consist of the literals that occur positively (`Pos`) or negatively (`Neg`) in a non-tautological leaf and contain the eigenvariable α (`alpha`), one of the eigenvariables $\beta_1, \ldots, \beta_p$ (`beta`), or none of them (`neutral`).

Afterwards, we build the unification candidates as in Definition 39. But instead of constructing them naively, we only build pairs of a positive (`Pos`) and a negative (`Neg`) literal. Moreover, we do not pair two neutral literals.

The implementation of the unification is straight forward, as we check all differences occurring at any position of the literals of a unification candidate and unify them according to the given grammar if possible. All $\mathcal{G}^*$-unified literals are collected in a set which is then returned.

5.2 Computing $\mathbf{S\Pi_2}$-Gs

The focus of this thesis was to find Π_2-cut formulas for an already known $\mathbf{S\Pi_2}$-**G**. In order to introduce cuts into cut-free proofs, we have to compute a covering grammar first. The grammar computation is a difficult problem on its own and we only give a survey over the possible solutions including a first prototype (cf. also Section 5.3).

For a better presentation, we first describe how the prototype addresses the problem of finding a 'good' $\mathbf{S\Pi_2}$-**G**, i.e. a grammar with a minimal size with respect to a fixed number of nonterminals $\{\alpha, \beta_1, \ldots, \beta_p\}$. The core of the algorithm is described in [EEH17] and was developed for Π_1-cut introduction with many cuts (therefore, the number of eigenvariables/nonterminals is not necessarily one). It computes totally rigid acyclic tree grammars by reducing this problem in polynomial time to the **MaxSAT** optimization problem (for a list of **MaxSAT** solvers see [ALMP08]). Thereby, we can enforce the algorithm to only compute totally rigid acyclic tree grammars that also fulfill the requirements of being a $\mathbf{S\Pi_2}$-**G**. Moreover, we ask for at least one production rule of the form $\alpha \to r$ where α is the eigenvariable of the universally quantified variable of the potential cut formula. Then the algorithm finds a minimal $\mathbf{S\Pi_2}$-**G** for a fixed number of nonterminals.

Even though the described approach computes $\mathbf{S\Pi_2}$-**G**s, there are several problems that are of future interest:

- **The number of nonterminals.**
  As mentioned, the algorithm needs a fixed number of nonterminals for which it finds an optimal grammar. Whether the number of nonterminals is optimal can, so far, only be determined by brute-force search. For practical applications it might be interesting to investigate the statistical impact of the size of the cut-free proof on the optimal number of nonterminals. For a general theoretic solution, it seems

to be most promising to define a new method that is able to increase the number of nonterminals in the process of computing the grammar.

- **The additional requirements of $\mathbf{S\Pi_2}$-Gs.** (Cf. Subsection 5.3.1)
  As $\mathbf{S\Pi_1}$-**G**s for many cuts, $\mathbf{S\Pi_2}$-**G**s have many eigenvariables. But there are two types of eigenvariables: the one introduced for the universal quantifier (α) and the ones introduced for the existential quantifier ($\beta_1, \ldots, \beta_p$). The experiments showed that the algorithm finds much more solutions if we ask for at least one production rule of the form $\alpha \to r$. This suggests a closer investigation of the role of the different eigenvariables for computing solvable Π_2-**SEHS**s.

- **Non-solvable Π_2-SEHSs.**
  The existence of non-solvable Π_2-**SEHS**s as well as the restrictions of the current implementation, which was designed to compute balanced solutions, lead to the following non-exhaustive list of questions: Are there relevant fragments of $\mathbf{S\Pi_2}$-**G**s that are always solvable? Which properties increase the probability of the solvability/existence of balanced solutions? Is there useful information in non-solvable grammars that can be used in the computation of a new grammar?

5.3 Experiments

When testing the implementation of the methods described in this thesis, one has to make several choices. This starts already with the implementation of a $\mathbf{S\Pi_2}$-**G** finding procedure, even though the thesis focuses on the search for a cut formula. If we only want to test $\mathcal{G}^*$-unification, we do not only need a provable end sequent; we also need a reduced representation and the corresponding $\mathbf{S\Pi_2}$-**G**, i.e. a Π_2-**SEHS**. Unfortunately, this is not feasible for large scale experiments. For the experiments, we have to consider already existing databases of proofs and those do not contain any information about $\mathbf{S\Pi_2}$-**G**s. Hence, we can test the $\mathcal{G}^*$-unification only for a few examples or together with an implementation of a grammar finding method.

We distinguish between two kinds of experiments in this section. The first concentrates on experiences about different aspect of the implementation for a few examples while the second one describes a run of the full algorithm – with different parameters – based on examples from a large scale database. The partitioning follows the structure of the Subsections 3.5.2 and 3.5.3.

5.3.1 Experiences with the Automated Introduction of Π_2 Cuts

Checking the applicability of the $\mathcal{G}^*$-unification method of Section 4.7 is difficult, since we require a $\mathbf{S\Pi_2}$-**G** and a provable end sequent. Computing $\mathbf{S\Pi_2}$-**G**s by hand is too time consuming, but nonetheless, we check the method in some interesting cases. These tests gave us the confidence that the method behaves well and is able to find interesting cut formulas. But they do not tell anything about the practice.

In the thesis, we have already introduced one schema of examples for which $\mathbf{S}\Pi_2$-**G**s exist, i.e. the sequents S'_n of Section 4.9 with the schema of $\mathbf{S}\Pi_2$-**G**s of Definition 48. Moreover, we showed that, in theory, the method should find the cut formula for each instance. In order to check our code, we implemented some instances of the schema with the corresponding grammars. Indeed, we also checked some simple modifications.

These examples can be found in the the file:

```
gapt/tests/src/test/scala/gapt/cutintro/
     IntroducePiCutTest.scala
```

One instance is depicted in Figure 5.1. The two outermost functions "`"This" should`" and "`"be computed correctly" in`" can be ignored. They only ensure that this tests can be executed in a special mode for checking the correctness of the major parts of the algorithm. The actual implementation of the example are the definitions that start with `val`. `A3`, `B1`, `B2`, `B3`, `C3`, and `D` encode the formulas occurring in the reduced representation `Rere`. Other than in the theoretical part of the thesis the eigenvariables are `x` and `y1`, `y2`. This is due to the **GAPT** framework which requires variables to start with one of the letters `u`, `v`, `w`, `x`, `y`, `z`. The functions `f1`, `f2`, and `f3` correspond to the functions f_1, f_2, and f_3 in S'_n telling us that n is actually 3. `seHs` is of type `Pi2SeHs` and needs – apart from the reduced representation – information about the grammar. Hence, we find as input the eigenvariables and terms on which the eigenvariables are mapped to. The last three lines define the names of the variables in a potential cut formula and call the `introducePi2Cut` method. The result is directly checked, as it should be

```
( Option( fof"P($xName,f($yName))" ), yName, xName )
```

Above the example, we find the following lines:

```
//
// Number of non-tautological leaves
//   24
// Number of unified literals
//   1
// No 'allowed clauses' were computed
// Number of checked Formulas
//   1
```

These lines are comments and give us information about the performance of the example. The number of non-tautological leaves without counting supersets is 24. For these, we computed only 1 $\mathcal{G}^*$-unified literal. For this simple example, no allowed clause had to be computed. Note that we do not count unit clauses, since they are checked separately.

Figure 5.1: Implemented instance of S'_n of Section 4.9

```
"This" should {
  "be computed correctly" in {
    val A3 = fof"P(x,f1(x))|P(x,f2(x))|P(x,f3(x))"
    val B1 = fof"P(x,f1(x))->P(x,f(f1(x)))"
    val B2 = fof"P(x,f2(x))->P(x,f(f2(x)))"
    val B3 = fof"P(x,f3(x))->P(x,f(f3(x)))"
    val C3 = fof"(P(c,f(y1))&P(f(y1),f(y2)))->P(c,g(y2))"
    val D = fof"P(c,g(y2))"
    val Rere = A3 +: B1 +: B2 +: B3 +: C3 +: Sequent() :+ D
    val seHs = new Pi2SeHs( Rere, fov"x",
                            List( fov"y1", fov"y2" ),
                            List( fot"c", fot"f(y1)" ),
                            List( fot"f1(x)",
                                  fot"f2(x)",
                                  fot"f3(x)" ) )
    val xName = fov"xName"
    val yName = fov"yName"
    introducePi2Cut( seHs, yName, xName ) must_==
                ( ( Option( fof"P($xName,f($yName))" ),
                            yName,
                            xName ) )
  }
}
```

Altogether, we checked exactly one formula, i.e. `P(xName,f(yName))`, which gave us already a solution.

This example shows that – given the correct grammar – we are able to find the cut formula with the current implementation. The same file contains also S'_4 that behaves similarly to the lower instance, but produces 96 non-tautological leaves. In order to check the robustness, we modified the example and replaced the binary predicate symbol $P(x, y)$ with the binary formulas $P(x) \vee Q(y)$ or $(P(x) \wedge Q(y)) \vee (P(y) \wedge Q(x))$. We checked several instances of these formulas as well. All tests were successful, but the performance changed. The costs to construct a formula such as $(P(x) \wedge Q(y)) \vee (P(y) \wedge Q(x))$ is much higher than producing a single literal. More precisely, for the most complex example, we computed 6 $\mathcal{G}^*$-unified literals which allow in general 2^6 different clauses. Depending on the instance (the highest was 3), we computed up to 7 allowed clauses and checked up to 14 formulas until we found the solution.

Altogether, these examples show that the $\mathcal{G}^*$-unification method reduces the search space significantly and allows us to search for Π_2-cut formulas in complex frameworks, even

Figure 5.2: Pigeonhole principle

$$\forall n \in \mathbb{N}, f\colon \mathbb{N} \to \mathbb{N}_n \exists i \leq n \in \mathbb{N} \forall m \in \mathbb{N} \exists j \geq m \in \mathbb{N}. f\,(j) = i$$

(a) The infinite pigeonhole principle

$$\forall f\colon \mathbb{N} \to \mathbb{N}_2 \exists i, j \leq 2 \in \mathbb{N}. i < j \wedge f\,(i) = f\,(j)$$

(b) A finite variant of the pigeonhole principle

though, it does not guarantee finding a solution.

Another example in the same file is the reduced representation of one Π_2 cut of a variant of the pigeonhole principle. The pigeonhole principle is an important problem in the proof theory community [Tao07]. In its most general version, it says that if we distribute an infinite number of pigeons over a finite number of holes, we will find a hole containing an infinite number of pigeons. In order to prove this, we introduce lemmas telling us that a fixed hole either contains an infinite number of pigeons or we can ignore it for the rest of the reasoning. Since there is a finite number of holes, we will eventually find such a hole containing infinitely many pigeons.

Formally, we can represent the pigeonhole principle as depicted in Figure 5.2a. In this form, we would require an induction rule in order to proof it. For this reason, we consider an instance of it, i.e. we restrict the number of holes and look only for two distinct pigeons appearing in the same hole (see Figure 5.2b). Moreover, we consider f as an uninterpreted function in order to avoid the quantification over a higher type. This variant is taken from [AHL15] and was also used for applying analytic methods to it (see [BHL$^+$04, BBS97, Her95, Urb00]). One proof of this variant relies on two Π_2 cuts with the cut formulas

$$\forall x \exists y. x \leq y \wedge f\,(y) = 0 \text{ and}$$
$$\forall x \exists y. x \leq y \wedge f\,(y) = 1.$$

In the corresponding framework (we need associativity of "=", a binary maximum function, and some further theory), we can show that

$$\vdash \forall x \exists y. x \leq y \wedge f\,(y) = 0, \forall x \exists y. x \leq y \wedge f\,(y) = 1$$

is provable. Moreover, both formulas imply the variant of Figure 5.2b. Now, we can extract the Π_2-**EHS** of the uppermost cut and forget about the cut formula itself. This gives us a Π_2-**SEHS** and its encoding is depicted in Figure 5.3.

Since our current implementation cannot handle equality, we introduced the predicate `Pg` (`g` for "Gleichheit", a German expression denoting equality) with the necessary axioms instead. The predicate symbol `Pkl` (`kl` for "kleiner", a German expression denoting less) says that the first argument is less than the second. The symbol `Pklg` (`klg` for "kleiner

Figure 5.3: Π_2-**SEHS** corresponding to the pigeonhole principle

```
val T1 = fof"Pkl(0,y1)&Pg(f(0),f(y1))"
val T2 = fof"Pkl(y1,y2)&Pg(f(y1),f(y2))"
val I0 = fof"Pklg(c,M(c,x))&Pg(f(M(c,x)),0)"
val Gamma1 = fof"Pklg(c,M(c,x))&Pklg(x,M(c,x))"
val Gamma21 = fof"Pg(f(M(c,x)),0)|Pg(f(M(c,x)),s(0))"
val Gamma22 = fof"Pg(f(0),0)|Pg(f(0),s(0))"
val Gamma23 = fof"Pg(f(y1),0)|Pg(f(y1),s(0))"
val Gamma24 = fof"Pg(f(y2),0)|Pg(f(y2),s(0))"
val Delta11 = fof"(Pg(f(y1),0)&Pg(0,f(y2)))->
                    Pg(f(y1),f(y2))"
val Delta12 = fof"(Pg(f(y1),s(0))&Pg(s(0),f(y2)))->
                    Pg(f(y1),f(y2))"
val Delta13 = fof"(Pg(f(0),0)&Pg(0,f(y1)))->
                    Pg(f(0),f(y1))"
val Delta14 = fof"(Pg(f(0),s(0))&Pg(s(0),f(y1)))->
                    Pg(f(0),f(y1))"
val Delta21 = fof"Pklg(s(0),y1)->Pkl(0,y1)"
val Delta22 = fof"Pklg(s(y1),y2)->Pkl(y1,y2)"
val Ref1 = fof"Pg(f(y1),0)->Pg(0,f(y1))"
val Ref2 = fof"Pg(f(y2),0)->Pg(0,f(y2))"
val Ref3 = fof"Pg(f(y1),s(0))->Pg(s(0),f(y1))"
val Ref4 = fof"Pg(f(y2),s(0))->Pg(s(0),f(y2))"
val Rere = Ref1 +: Ref2 +: Ref3 +: Ref4 +:
           Gamma1 +:
           Gamma21 +: Gamma22 +: Gamma23 +: Gamma24 +:
           Delta11 +: Delta12 +: Delta13 +: Delta14 +:
           Delta21 +: Delta22 +:
           Sequent() :+ T1 :+ T2 :+ I0
val seHs = new Pi2SeHs( Rere, fov"x",
                        List( fov"y1", fov"y2" ),
                        List( fot"0", fot"s(y1)" ),
                        List( fot"M(c,x)" ) )
val xName = fov"xName"
val yName = fov"yName"
introducePi2Cut( seHs, yName, xName )
```

gleich", a German expression denoting less than or equal) consequentially allows the two arguments to be equal as well. As already mentioned, we also add a binary maximum function `M`. The eigenvariables are `x` for the universal eigenvariable and `y1` and `y2` for the two existential eigenvariables. The productions starting with an eigenvariable are:

$$\begin{aligned} \texttt{x} &\to \texttt{0} \mid \texttt{s(y1)}, \\ \texttt{y2} &\to \texttt{M(c,s(y1))}, \\ \texttt{y1} &\to \texttt{M(c,s(0))}. \end{aligned}$$

In the test run, we checked whether the algorithm produces

```
Pklg(xName,yName)&Pg(f(yName),s(0))
```

or a permutation of it as cut formula. For this reason we add to `introducePi2Cut( seHs, yName, xName ):`

```
must beOneOf(
  ( Option( fof"Pklg($xName,$yName)&Pg(f($yName),s(0))" ),
    yName,
    xName                                                    ),
  ( Option( fof"Pg(f($yName),s(0))&Pklg($xName,$yName)" ),
    yName,
    xName                                                    )  )
```

The test was successful and the method found the required cut formula. During the construction, it computed only three $\mathcal{G}^*$-unified literals, one allowed clauses, and had to check only one formula. Altogether, this first experiences show that the implementation finds cut formulas for well defined $\mathbf{S}\Pi_2$-**G**s.

In a next step, we wanted to see whether our prototype implementation of the grammar finding method can compute the $\mathbf{S}\Pi_2$-**G**s of Definition 48. For this reason, we have incorporated the sketched proofs of Figure 4.11 in the **GAPT** system. In order to run the examples within **GAPT**, we have to run two commands that are depending on the instance `Inst`:

```
import examples.ExponentialCompression._
val proof = instantiateProof.Instantiate( le"preOmega Inst" )
```

`Inst` has to be a natural number encoded as successors of `0`, for instance `(s 0)`, `(s (s 0))`, `(s (s (s 0)))`, etc. Due to the **GAPT** framework and the schematic implementation, there are slight differences to the proofs in Figure 4.11. Since we only use it as a starting point to which we apply a cut-elimination algorithm, we only have to ensure that the Herbrand sequents of the corresponding cut-free proofs are as expected.

This is indeed the case and we can apply our method. For running a cut-elimination algorithm, **GAPT** offers several methods. Starting with the fastest, they are:

```
val cfProof = normalizeLKt.lk( proof )
val cfProof = eliminateCutsET( LKToExpansionProof( proof ) )
val cfProof = cutNormal( proof )
```

Note that the second command produces an expansion proof. In general, we could transform it back to a sequent-calculus proof, but our method can also be applied to expansion proofs. The last cut-elimination method is very slow and should only be applied to low instances.

Before starting the cut-introduction algorithm, we have to define the eigenvariables and have to decide how many we want to consider. In this particular case, we know exactly the number of eigenvariables, i.e. `(Inst+1)` many existential eigenvariables. The number of universal eigenvariables is independent of the example and always 1. In order to define them, we run the two commands:

```
val alpha = FOLVar( "xa" )
val betas = for ( i <- 1 to (Inst+1) ) yield FOLVar( s"y$i" )
```

where `(Inst+1)` has to be replaced with the actual number. Note that **GAPT** requires strings for variables to start with one of the characters `u,v,w,x,y,` or `z`. Finally, we can run the method by the call

```
Pi2CutIntroduction( cfProof,alpha,betas.toVector )
```

Before discussing the performance of the cut-introduction method, we provide some information about the sizes of the proofs. For simplicity, we only give the number of occurring sequents. The instantiations of `preOmega`, i.e. the proofs we apply cut elimination to, are growing linearly with the instantiation:

$$32, 65, 90, 115, 140, 165, \ldots$$

The cut-free proofs are much larger, but there size also depends on the used cut-elimination algorithm. We get the following sizes:

$$\texttt{normalizeLKt.lk}: \ 18, 91, 696, 7419, 101558, \ldots$$
$$\texttt{cutNormal}: \ 18, 89, 654, 6735, \ldots$$

As expected, the size of the cut-free proofs is increasing fast.

When running the `Pi2CutIntroduction` command, we can get further information by running it together with the `verbose` function:

```
verbose( Pi2CutIntroduction( cfProof,alpha,beta.toVector ) )
```

For the first instance, i.e. `Inst` being `0`, we get the following output:

```
[Pi2CutIntro] Language size: 4
start computing grammar
[Pi2CutIntro] Found grammar of size: 6
x0 -> 'a1:P(x,fn(0,x))'(f(a))
x0 -> 'a2:P(x,y)->P(x,f(y))'(f(a), fn(0, f(a)))
x0 -> 'a0:P(f(x),f(z))->P(f(x),g(z))'(a, fn(0, f(a)))
x0 -> 's0:P(x,g(y))'(f(a), fn(0, f(a)))
xa -> fn(0, f(a))
y1 -> f(a)
start computing cut formula
[Pi2CutIntro] Could not find cut formula.
res0: Option[gapt.proofs.lk.LKProof] = None
```

The lines `start computing...` only inform the user in which phase the algorithm is. The language size gives us the number of terms in the Herbrand term set. Here, it found a grammar of size `6`, i.e. there are 6 production rules, where `xa` and `y1` are – as defined before – the eigenvariables and `x0` is the starting symbol. Even though `->` seems to be the symbol of a production rule, it is not. For the first four lines starting with `x0`, it can be read as such, but the production rules for the eigenvariables are `xa` $\rightarrow$ `f(a)` and `y1` $\rightarrow$ `fn(0, f(a))`. This is due to technical reasons. Anyhow, in this particular case, the productions starting with an eigenvariable are irrelevant anyway. They do not occur on the right side of any other production rule. They only occur in the grammar because we force the algorithm to compute a grammar containing at least one production starting with the universal eigenvariable. This is motivated by much better results during the experiments (see Subsection 5.3.2). In order to be able to read the right side of such a line, we look at

```
x0 -> 'a2:P(x,y)->P(x,f(y))'(f(a), fn(0, f(a)))
```

The right side can be split in two parts: `'a2:P(x,y)->P(x,f(y))'` and `(f(a), fn(0, f(a)))`. The first is the term that correspond to the formula $P(x,y) \rightarrow P(x,f(y))$. There are two variables occurring in the formula: x and y. In the second part, we find a tuple containing two terms: `f(a)` and `fn(0 ,f(a))`. They have to be substituted for the variables in the first part such that we get the term

```
'a2:P(f(a),fn(0 ,f(a)))->P(f(a),f(fn(0 ,f(a))))'
```

representing the formula

$$P(f(a), f_0(f(a))) \rightarrow P(f(a), f(f_0(f(a)))).$$

The string a2 is just a label. For better readability and better comparison to Definition 29, we introduce the term representations $h_{\mathtt{s0}}$,$h_{\mathtt{a0}}$,$h_{\mathtt{a1}}$, and $h_{\mathtt{a2}}$. They represent the formulas on the right side of the production rules with x0 on the left side such that – for instance – $h_{\mathtt{a2}}(fa, f_0fa)$ translates to

$$P(f(a), f_0(f(a))) \to P(f(a), f(f_0(f(a)))).$$

Altogether, we get the $\mathbf{S\Pi_2}$-$\mathbf{G}$ $\mathcal{G}(0)$ with the nonterminals x0,xa,y1 and the production rules

$$\begin{aligned}
&\mathtt{x0} \to h_{\mathtt{s0}}(fa, f_0fa) \mid h_{\mathtt{a0}}(a, f_0fa) \mid h_{\mathtt{a1}}(fa) \mid h_{\mathtt{a2}}(fa, f_0fa),\\
&\mathtt{xa} \to fa\\
&\mathtt{y1} \to f_0fa.
\end{aligned}$$

Of course, we could not find a cut formula for this grammar.

The next instantiation – Inst being (s 0) – is already much more interesting. Thereby, we had surprisingly two different outcomes depending on the machine we run the method on, once on the laptop of Grabriel Ebner from the Institute of Discrete Mathematics and Geometry at TU Wien and once on the author's computer at the university. Both use a Debian Linux system as operating system and there are no relevant difference that explain the different behaviour. The outcomes are:

```
[Pi2CutIntro] Language size: 17
[Pi2CutIntro] Found grammar of size: 9
x0 -> 'a2:P(x,y)->P(x,f(y))'(f(xa), fn(0, f(xa)))
x0 -> 'a0:P(f(x),f(y))&P(f(y),f(z))->P(f(x),g(z))'(a, y1, y2)
x0 -> 'a2:P(x,y)->P(x,f(y))'(f(xa), fn(s(0), f(xa)))
x0 -> 'a1:P(x,fn(0,x))|P(x,fn(s(0),x))'(f(xa))
x0 -> 's0:P(x,g(y))'(f(a), y1)
xa -> fn(s(0), f(xa))
xa -> fn(0, f(xa))
y1 -> y2
y2 -> a
[Pi2CutIntro] Cut formula: P(f(xCut), f(yCut)): o
```

and

```
[Pi2CutIntro] Language size: 17
[Pi2CutIntro] Found grammar of size: 9
x0 -> 'a0:P(f(x),f(y))&P(f(y),f(z))->P(f(x),g(z))'(a, y1, y2)
x0 -> 's0:P(x,g(y))'(f(a), y1)
x0 -> 'a1:P(x,fn(0,x))|P(x,fn(s(0),x))'(f(y2))
x0 -> 'a2:P(x,y)->P(x,f(y))'(f(y2), y1)
```

```
xa -> fn(s(0), f(xa))
xa -> fn(0, f(xa))
xa -> a
y1 -> y2
y2 -> a
[Pi2CutIntro] Could not find cut formula.
```

Both produce a $\mathbf{S\Pi_2}$-**G**, but only for one of them the method finds a cut formula. In order to understand the differences, we translate the outcomes to the corresponding $\mathbf{S\Pi_2}$-**G**s. For a better comparison to the grammars of Definition 48, we also translate the symbols. The starting symbol `x0` occurs in the grammars as τ and the other nonterminals `xa,y1,y2` are represented by α, β_2, β_1, respectively. Note that `y1` corresponds to β_2. The representations of terms corresponding to the formulas will be abbreviated by h_ζ where ζ is `s0,a0,a1,` or `a2`. Thus, $h_{\mathtt{s0}}$ represents `'s0:P(x,g(y))'` and $h_{\mathtt{s0}}(fa)\,\beta_2$ represents `'s0:P(x,g(y))'(f(a), y1)`. The productions of the first $\mathbf{S\Pi_2}$-**G** $\mathcal{G}'$ are

$$\begin{aligned}
\tau &\to h_{\mathtt{s0}}(fa, \beta_2) \mid h_{\mathtt{a0}}(a, \beta_2, \beta_1) \mid h_{\mathtt{a1}}(f\alpha) \mid h_{\mathtt{a2}}(f\alpha, f_0 f\alpha) \mid h_{\mathtt{a2}}(f\alpha, f_1 f\alpha),\\
\alpha &\to \beta_1 \mid a,\\
\beta_2 &\to f_0 f\beta_1 \mid f_1 f\beta_1,\\
\beta_1 &\to f_0 fa \mid f_1 fa.
\end{aligned}$$

When comparing with $\mathcal{G}_1$ of Definition 48, there are only small differences. Since the implementation does not require the end sequent to be in the form $\forall\vec{x}F \vdash \exists\vec{y}G$, we get different term representations. The only non technical difference is the function f that occurs once in the τ-productions and once in the α-productions. This affects also the computed cut formula such that in the one case, the cut formula is $\forall x \exists y P(fx, fy)$ and in the other case, it is $\forall x \exists y P(x, fy)$.

The second grammar $\mathcal{G}''$ has the productions

$$\begin{aligned}
\tau &\to h_{\mathtt{s0}}(fa, \beta_2) \mid h_{\mathtt{a0}}(a, \beta_2, \beta_1) \mid h_{\mathtt{a1}}(f\beta_1) \mid h_{\mathtt{a2}}(f\beta_1, \beta_2),\\
\alpha &\to \beta_1 \mid a,\\
\beta_2 &\to f_0 f\beta_1 \mid f_1 f\beta_1 \mid a,\\
\beta_1 &\to f_0 fa \mid f_1 fa \mid a.
\end{aligned}$$

Note that all productions from the starting symbol contain β_1 or β_2, but α does not occur anywhere. In fact, every literal occurring in the corresponding reduced representation

$$\begin{aligned}
P(\beta_1, f_0\beta_1) \vee P(\beta_1, f_1\beta_1), P(f\beta_1, \beta_2) \to P(f\beta_1, f\beta_2),\\
P(fa, f\beta_1) \wedge P(f\beta_1, f\beta_2) \to P(fa, g\beta_2) \vdash P(fa, g\beta_2)
\end{aligned}$$

contains β_1 or β_2. Hence, there cannot be a $\mathcal{G}^*$-unified literal and our method fails to find a cut formula. Indeed, the considered grammar could be read as a grammar for the introduction of a Σ_1 formula with two existential quantifiers, since the α-productions

are irrelevant. By dropping the α-productions, the grammar becomes even smaller. This suggests that the concept of a minimal $\mathbf{S}\Pi_2$-$\mathbf{G}$ is not a good criteria for Π_2-cut introduction.

Altogether, we have seen that the method is able to find cut formulas when the $\mathbf{S}\Pi_2$-$\mathbf{G}$ is chosen correctly. The current implementation suffers of the absence of a suited grammar computing algorithm. Since the used variant was designed for Π_1-cut introduction where every $\mathbf{S}\Pi_1$-$\mathbf{G}$ leads to a proof with Π_1 cuts, it concentrates on finding minimal grammars. Even though, this works well for the Π_1 case (see [EHL$^+$18, HLR$^+$14]), in the Π_2 case, the computed grammars produce non $\mathcal{G}^*$-unifiable reduced representations. Thus, we cannot find a solution.

5.3.2 Experiments with Thousands of Solutions from Theorem Provers

Apart from checking single examples, we executed the full algorithm – grammar computation and cut-formula construction – on the **TSTP** library (**T**housands of **S**olutions of **T**heorem **P**rovers; see [Sut09]). More precisely, we imported the prover9-part of the **TSTP** library into **GAPT**, similarly to the experiments of [HLR$^+$14]. As in [HLR$^+$14], the main motivation is the (comparatively) easy import due to the simple and clean proof output format Ivy. This library contains 6394 resolution proofs (53 more have been added since the publication [HLR$^+$14]). Of those, 37 could not be parsed and 3512 contain equality reasoning. The rest is translated into sequent calculus proofs. Moreover, 1488 of the proofs without equality have a trivial language, since every term in the corresponding Herbrand term set has its own root symbol. Hence, there are 1357 relevant proofs to which our algorithm was applied.

The performance comparison was conducted on a Debian Linux system with an Intel i5-4570 CPU and 8 GiB RAM. The timeout was set to 5 minutes and the experiments were executed in version 2.11 of **GAPT**. Since we were mainly interested in the performance of the cut-formula construction (cf. Subsection 5.3.1), we did not consider whether the found grammar of a problem has smaller size than the Herbrand term set. Nonetheless, we computed the number of problems for which a compressing grammar was found. Depending on the number of nonterminals 325 to 365 compressions were found. This corresponds to approximately $25,23\%$ of all cases. Since this number is relatively small, we applied the algorithm to all 1357 problems and tried to introduce cuts even when the size of the found grammar was larger than the size of the Herbrand term set. For the course of this subsections, when speaking about an application, we refer to a single attempt to introduce a cut for one of these 1357 problems with our algorithm. A successful application is consequentially an application of our method to one problem where the result is a proof with a cut that corresponds to the automatically computed grammar. Moreover, the algorithm will be applied in different modes and so we might speak about the successful applications referring to those applications that were successfully conducted in the current mode.

	Number of solutions	**%**	**% without timeouts**
$\mathbf{S\Pi_1}$-**G**	32	$2,36\%$	$3,07\%$
$\mathbf{S\Pi_2}$-**G** with one β	68	$5,01\%$	$6,37\%$
$\mathbf{S\Pi_2}$-**G** with two β's	137	$10,10\%$	$13,44\%$
$\mathbf{S\Pi_2}$-**G** with three β's	135	9.96%	$13,61\%$
$\mathbf{S\Pi_2}$-**G** with four β's	142	$10,47\%$	$14,29\%$
$\mathbf{S\Pi_2}$-**G** with five β's	138	$10,18\%$	$14,33\%$
Arbitrary grammar	252	$18,58\%$	$21,36\%$

Table 5.1: Performance of the full algorithm on the **TSTP** library

As already mentioned, we applied the method in different modes. In a first execution, we added no special constraints. The grammar finding algorithm requires a number of nonterminals/eigenvariables. There is always one universal eigenvariable. We decided to apply the algorithm with different numbers of existential eigenvariables; with "1 β", "2 β's", "3 β's", and "4 β's". This means that the algorithm searches only for $\mathbf{S\Pi_2}$-**G**s with the nonterminals $\{\tau, \alpha, \beta_1, \ldots, \beta_n\}$ where τ is the designated starting symbol, α is the nonterminal corresponding to the universal eigenvariable, n is the number of β's, and β_i is the nonterminal corresponding to the existential eigenvariable. In the execution without further constraints, we found only 16 cut formulas.

After some further experiments, we ensured that the computed grammar contains at least a single production with the universal eigenvariable as left side. The performance improved considerably. Table 5.1 shows how many cut formulas were computed depending on the number of β's. The first column depicts the absolute number of solutions, the second column the percentage of successful applications, and the third column the percentage of successful applications without regarding timeouts ($32 \mid 32/1357 \approx 2,36\% \mid 32/(1357 - 313) \approx 3,07$). A special case is the first line in which results for $\mathbf{S\Pi_1}$-**G**s are depicted. $\mathbf{S\Pi_1}$-**G**s containing only the nonterminals τ and α can be extended to $\mathbf{S\Pi_2}$-**G**s by adding dummy productions and dummy nonterminals. For this reason, we also computed the minimal $\mathbf{S\Pi_1}$-**G** for introducing a single cut and translated it into a $\mathbf{S\Pi_2}$-**G**. Then we applied the $\mathcal{G}^*$-unification method as we did for the other cases. The last line of the table are the combined results, i.e. at least one application found a cut formula and at least one application had not a timeout.

Compared to the 16 solutions of the first application, we were able to increase the number of solutions by a factor of more than 15. Note that we only ensured that there is a production with the universal eigenvariable on the left side. In total, we found cut formulas for $18,58\%$ of the 1357 problems. Unfortunately, the cut formulas are almost all purely universal or purely existential. The only actual Π_2-cut formulas – that were found during the tests on the **TSTP** library – have the quantifier free bodies

```
truth( isa( x,knight ) ) & truth( isa( other,y ) )
```

```
| truth( isa( other,y ) )  & truth( isa( x,knave ) )
```

and

```
  truth( isa( x,knave ) )  & truth( isa( other,y ) )
| truth( isa( x,knight ) ) & truth( isa( other,y ) )
```

where `x` is the universally quantified variable, `y` is the existentially quantified variable, `&` encodes $\wedge$, and `|` encodes $\vee$. They were found for the problems

`PUZ/PUZ035-5/Prover9---1109a.UNS-Ref.s`

(see Appendix A.1) and

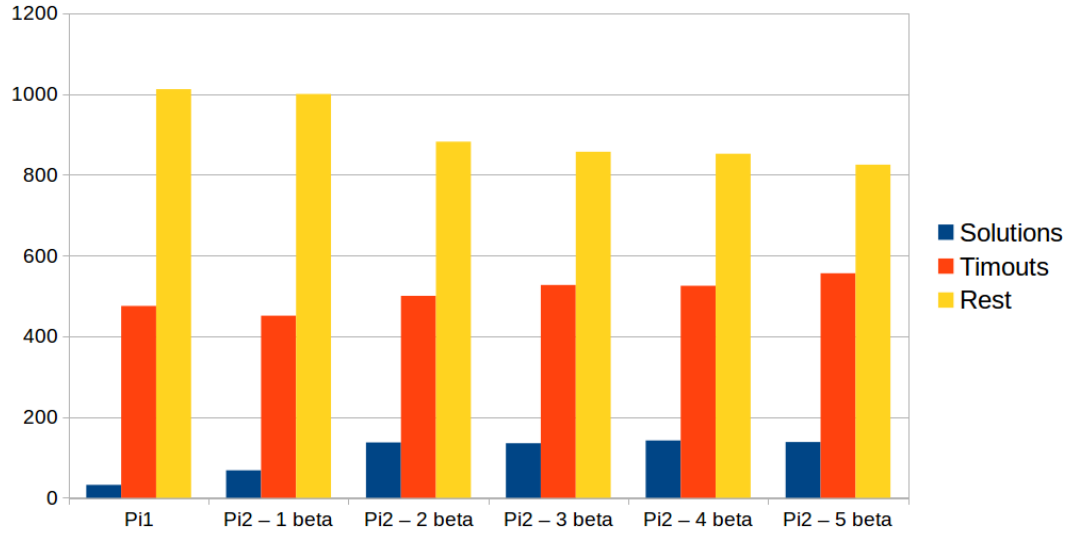
`PUZ/PUZ035-6/Prover9---1109a.UNS-Ref.s`

(see Appendix A.2) of [Sut09].

We visualized the results for different types of $\mathbf{S\Pi_2}$-**G**s in Figure 5.4. The blue bars represent the number of solutions found, as in Table 5.1. While the applications with more than one β produce a similar amount of solutions, we get only few solutions for the simple grammars containing either only one existential eigenvariable or being a transformed $\mathbf{S\Pi_1}$-**G**. We can also see that the number of timeouts increases with the number of β's. As we will see below, this is mainly due to the grammar computation. The higher amount of timeouts might also effect the number of found solutions. Nonetheless, we believe that for the **TSTP** library the consideration of even more β's would not effect the performance considerably. A high number of β's makes only sense when the considered example has a large term set. If the considered language is large and also the number of nonterminals is high, the algorithm will most likely end in a stack overflow error. The last bar contains all other possible outcomes which most often means that for the considered grammar no balanced solution exists. In few cases – each below 5% – we either run out of memory, had a stack overflow, or had some other exception.

Figure 5.5 presents how much time each part of the algorithm required. It shows only the relative times for completed executions. Hence, an execution that run into a timeout is not considered, because we could not compute the ratios between the various parts in case of a timeout. In each subfigure, we can see a cake diagram with five colours: Blue represents the amount of time spent for the construction of the cut formula, i.e. the construction of the $\mathit{DNTA}(\varphi)$ (for some φ; see Section 5.1), the $\mathcal{G}^*$ unification, and the combining of the literals to formulas in **DNF**. Red represents the time spent for minimizing the "stable" grammar while green represents the time spent for finding such a grammar. For the discussion of this section, we can put them together and consider them as the time spent for computing a grammar. Yellow shows the time spent for parsing

Figure 5.4: Number of solutions compared to timeouts and other unsolved problems; Subsection 5.3.2



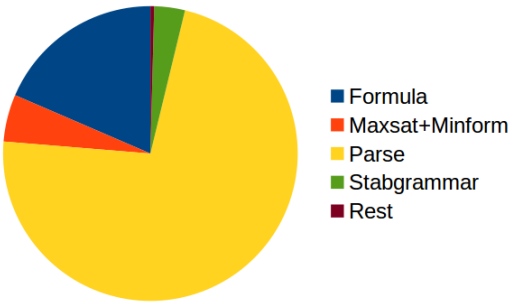
the problem and brown is the rest. All ratios are the average over all executions (for the considered type of grammar) for which the algorithm terminated without timeout.

As mentioned above, the time spent for computing the grammar – more precisely for minimizing the stable grammar – increases with the number of existential eigenvariables. This confirms the result of [EEH17]. Interestingly, the ratio for the cut-formula construction shrinks, i.e. the absolute amount of time stays approximately constant.

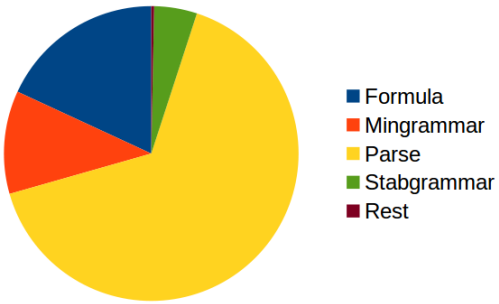
In Figure 5.6 and Figure 5.7, we compare the times spent for the formula construction with the times spent for the grammar computation according to the size of the Herbrand term set. The axes are both logarithmic to base 10 and the y-axe gives the logarithmic time in milliseconds. Again, we can make a similar observation as in Figure 5.5: The more eigenvariables the grammar has, the more time the grammar algorithm needs while the formula construction stays almost constant. Moreover, there is an interesting behaviour for term-set sizes around $1,5$ (around 31 terms). Independent from the considered grammar, the cut-formula construction is expensive for those problems. Surprisingly, this changes completely when the term-set sizes are larger than 100 (in the figures at 2). This is probably due to the grammars that are computed for large term-set size. Since the grammars are minimal, we find most likely only a few $\mathcal{G}^*$-unified literals and hence, the formula construction is relatively cheap. This explanation is also supported by the fact, that the biggest term-set size for which a cut-formula was computed is 44.

In summary, we can say that there is a potential for an efficient implementation of cut introduction via the $\mathcal{G}^*$-unification method. It reduces the search space considerably and is already able to compute a significant amount of cuts, even though the computed

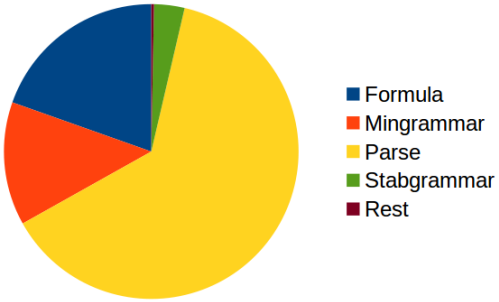
Figure 5.5: Distribution of spent time (relative)



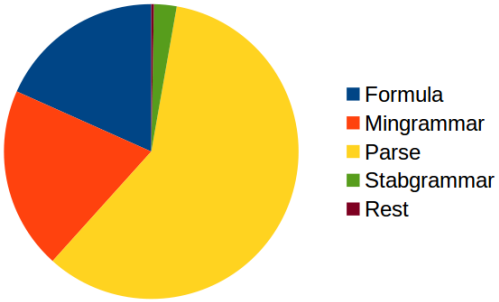
(a) Distribution for $\mathbf{S\Pi_1}$-**G**s



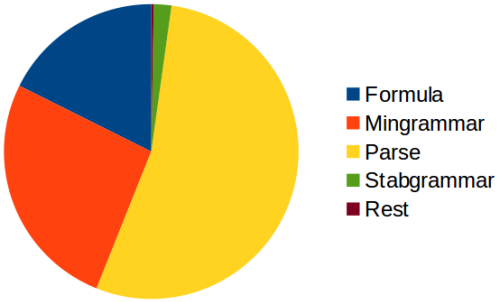
(b) Distribution for $\mathbf{S\Pi_2}$-**G**s with one β



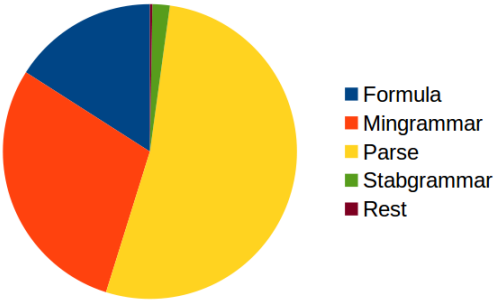
(c) Distribution for $\mathbf{S\Pi_2}$-**G**s with two β's



(d) Distribution for $\mathbf{S\Pi_2}$-**G**s with three β's

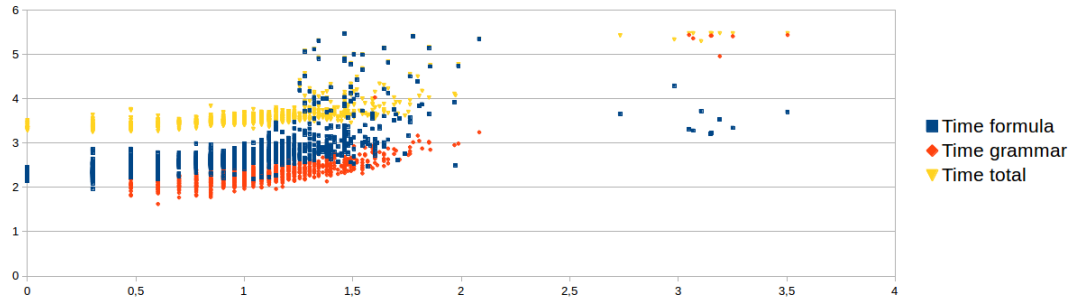


(e) Distribution for $\mathbf{S\Pi_2}$-**G**s with four β's

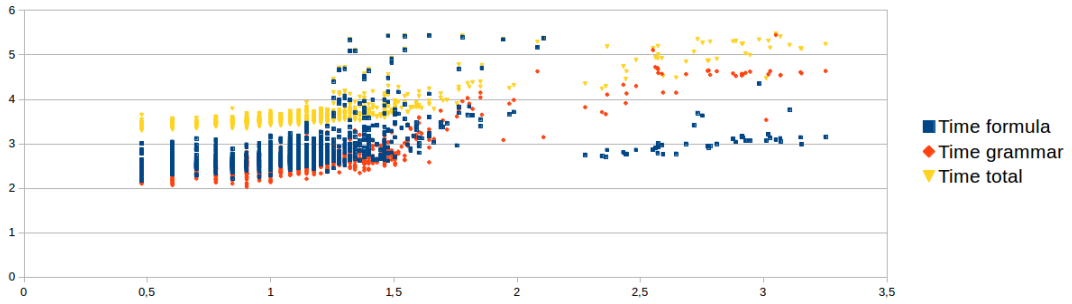


(f) Distribution for $\mathbf{S\Pi_2}$-**G**s with five β's

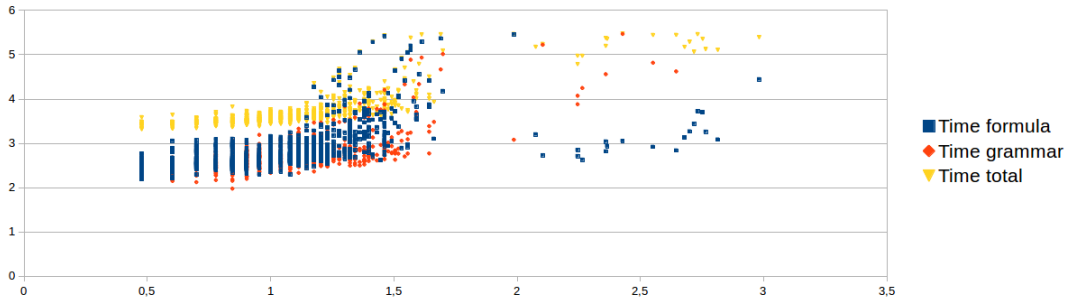
Figure 5.6: Spent time compared to the size of the term set I; Subsection 5.3.2



(a) Distribution for $\mathbf{S}\Pi_1$-**G**s

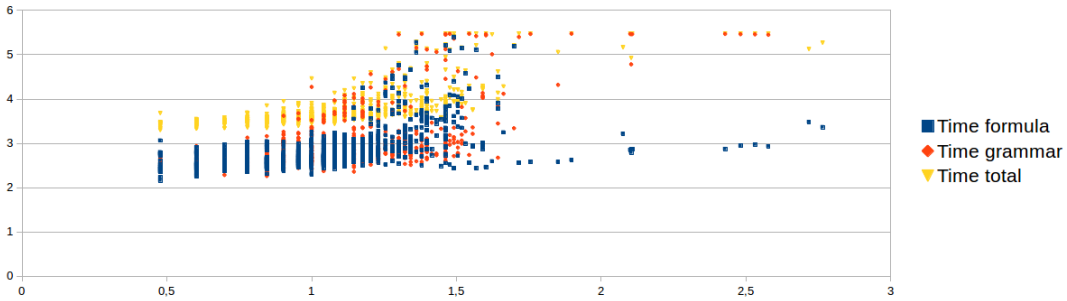


(b) Distribution for $\mathbf{S}\Pi_2$-**G**s with one β

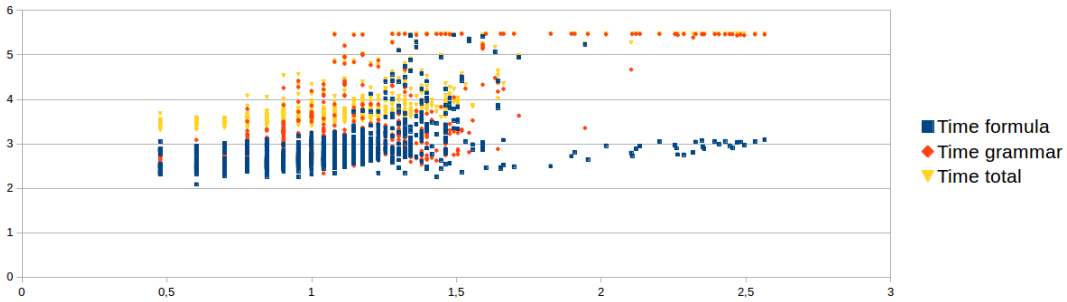


(c) Distribution for $\mathbf{S}\Pi_2$-**G**s with two β

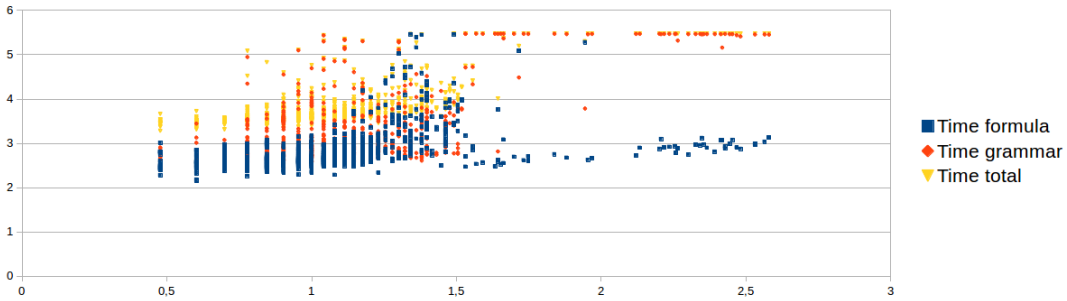
Figure 5.7: Spent time compared to the size of the term set II; Subsection 5.3.2



(a) Distribution for $\mathbf{S}\Pi_2$-**G**s with three β



(b) Distribution for $\mathbf{S}\Pi_2$-**G**s with four β



(c) Distribution for $\mathbf{S}\Pi_2$-**G**s with five β

grammars are not suited for this setting (for Π_1-cut introduction they are well suited). Unfortunately, the short comings of the current implementation do not allow us to find interesting cut formulas. Apart from the grammars, this is also due to the relatively small sizes of term sets for which a solution was found.

In order to overcome these short comings, there are two possible directions of research. One way would be to allow an interactive construction of grammars such that the user can verify whether a grammar is likely being solvable. Easy non-solvable cases as in the previous subsection could be avoided. Another idea would be the development of a new guideline for the grammar computation. We saw already that minimality is not a good criterion for $\mathbf{S}\Pi_2$-**G**s. Instead, the grammar computation should increase the probability that there are $\mathcal{G}^*$-unified literals. The condition that the computed grammars contain at least one production with the universal eigenvariable as nonterminal on the left side is a property increasing this probability. The existence of such a production makes it more likely that the corresponding reduced representation contains the universal eigenvariable, since the production affects the size of the grammar anyway, no matter if it is used or not. In order to get a minimal grammar, the probability increases that the grammar makes use of this production and replaces a term in the reduced representation with the universal eigenvariable. Note that this way, we would also need more time for the formula construction; more $\mathcal{G}^*$-unified literals have to be constructed and can be combined to a formula. Nonetheless, it would still be completely automatic.

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

In Chapter 2 and Chapter 3, we have presented definitions and notations required for an understanding of the problems in the realm of Π_2-cut introduction. Afterwards, we have revisited Π_1-cut introduction which also serves as a basis to understand major concepts such as extended Herbrand sequents, schematic grammars, and schematic extended Herbrand sequents (Sections 3.1, 3.2, and 3.3, respectively). In Section 3.4, we showed that Π_1-cut introduction via schematic grammars is always solvable and can be extended to more general frameworks, for instance cut introduction within first-order logic with equality or the introduction of several cut formulas with blocks of quantifiers at once (cf. [EH15, EHL$^+$18, HLW12, HLRW14, HLR$^+$14]). Moreover, in Section 3.5 we have seen applications of cut introduction, i.e. the introduction of meaningful lemmas and the compression of proofs.

The main chapter of this work is Chapter 4. We have introduced Π_2-**EHS**s, **S**Π_2-**G**s, and Π_2-**SEHS**s which are generalizations of Π_1-**EHS**s, **S**Π_1-**G**s, and Π_1-**SEHS**s, respectively. The question approached is whether there is a proof with a Π_2 cut whenever there is a **S**Π_2-**G** for a given cut-free proof. Section 4.5 provides two counterexamples, which imply that in general we are not able to construct a proof with Π_2 cut. Hence, we have given a full characterization of the Π_2-cut introduction problem in Section 4.6 reducing the problem to the definition of a suitable starting set. A starting set is a collection of literals that might occur in the cut formula. This characterization allows us to decide whether a combination of these literals is correct (Theorem 14 and Theorem 15). In Chapter 4, we have proceeded with a practical approach: The unification method presented in Section 4.7 is able to find a solution for a Π_2-**SEHS** whenever there is a balanced solution (see Definition 43 and Theorem 17). Even though the balanced solutions are a proper fragment of all solutions, they are the most promising in terms of inference and symbol complexity, since they do not contain interactive literals (see Definition 35 and Definition

37). Moreover, we have seen that the corresponding starting set can be constructed in polynomial time. The problem of finding out whether there exists a finite starting set with an inherent correct combination of literals for the general case is not treated in this work. Section 4.8 has shown that the methods, apart from the tedious notation, can easily be extended to Π_2-cut formulas with blocks of quantifiers. A demonstration of the compressive power of balanced solutions (Section 4.9) concludes the chapter. The presented unification method is able to find Π_2-cut formulas that compress a sequence of cut-free proofs exponentially.

An implementation of the unification method of Section 4.7 is available in the **GAPT** framework since version 2.5. In Section 5.1, we have presented how we implemented the method. In order to find suitable grammars, we have used the grammar generation algorithm developed in [EEH17] which, as discussed in Section 5.2, leaves room for improvement. Section 5.3 then summarizes the results achieved with this implementation. While the construction of a cut formula is feasible, the implementation is lacking a suitable grammar computing method (even though the current method behaves well in the Π_1 case).

6.2 Future Work

Comparing Π_1-cut introduction with Π_2-cut introduction provides a natural guide for further research. While Π_1-cut introduction can be performed in the presence of equality and we are able to introduce several Π_1 cuts at once, the Π_2-cut introduction is still restricted to the introduction of a single cut. On the one hand, the inclusion of equality may provide better solutions to Π_2-**SEHS**s or even solutions for Π_2-**SEHS**s that were not solvable before (see Remark 3). On the other hand, the introduction of several Π_2 cuts at once seems less promising since already the introduction of a single Π_2 cut can fail. The latter gives also a hint on how likely Π_3-cut or even Π_n-cut introduction find cut formulas for a given schematic grammar. Introducing a single Π_2 cut corresponds to the introduction of several Π_1 cuts where the cut formulas of all these Π_1 cuts are equal up to some terms. Analogously, introducing a single Π_3 cut corresponds to several Π_2 cuts and introducing a Π_n cut corresponds to several Π_{n-1} cuts. However, with an increasing complexity of the cut formula also the possible proof compression and the probability of meaningful lemmas increase. Especially in the context of inductions, techniques for introducing complex cut formulas might provide methods for the search of induction invariants.

Since the decidability of the Π_2-cut introduction problem has not been tackled, the following question arises: Is there an algorithm that takes as input a Π_2-**SEHS** and outputs a cut formula whenever there is one or a negative answer otherwise? In order to answer this question, the introduction of a normal form for cut formulas with respect to a fixed set of quantifier rules seems promising. The major idea is to restrict the literals potentially occurring in a cut formula. The restriction itself should not only be based on the signature, but on some notion of *nesting degree* that allows us the limit the number

of occurring terms and thereby, restricts the number of literals to a finite set.

A drawback of the presented methods for introducing cuts is that both, Π_1-cut and Π_2-cut introduction, require a cut-free proof. It would be desirable to construct lemmas in the process of proving a theorem. While resolution based provers are already capable of producing lemmas that are simple universally closed disjunctions of literals, there have been several attempts to integrate the cut rule into tableau provers (see [LMG94] and [Häh01]). The major problem is to decide when an application of the cut rule is reasonable and how to restrict the cut formula. [LP18] is a first step of the incorporation of grammar related cut-introduction techniques into theorem provers. By merging literals sharing a similar shape, all usual refinements of tableau based methods are able to reduce the proof size up to an exponential factor. In schematic grammars, the nonterminals represent eigenvariables of a proof with cut that are generalizations or abstractions of terms occurring in designated formulas in a cut-free proof. Assume a clause $\{P(a, fx), P(b, fx), P(c, fx)\}$. Following this idea, we can represent the same clause by $\{P(\alpha, fx)\}$ where α abstracts the terms a, b, and c. In [LP18], we propose a tableau calculus that builds proofs using such abstractions, we show that the calculus is compatible with the usual tableau methods, and that we can achieve an exponential compression of proofs. Note that while in the present work we use **DNF**s, in [LP18]we use **DNF**s. Beside improvements of this tableau method, integrations of this idea into resolution provers are worth investigating.

One aim of lemma generation is the construction of induction invariants. While first-order theorems are always provable without cut, this is no more the case when the system contains an induction rule. In order to automate provers capable of induction or increase the amount of automation in such provers, we have to provide the provers abstraction techniques. One line of future research is a combination of the approach of [LP18] and an extension of the $\mathcal{S}i$**LK** calculus presented in [CL17]. The $\mathcal{S}i$**LK** calculus gives a framework to construct *proof schemata*, i.e. a system for first-order logic with induction (see [LPW17] and also [ACP09, ACP10, ACP11a, ACP11b, AP11, AEP13, Cer14, DLRW13] for further information), implicitly enforcing the soundness of the links of the proof schemata. In order to use this calculus for proof search, we suggest a rule to abstract terms of sequents: Let $S(r)$ be a sequent where the term r occurs. Then we can try to prove $S(\alpha)$ where α might be unified with a term t that is smaller than r. Afterwards, we try to prove $S(t + sn)$ where s is the successor function and $S(t + n)$ can appear as leaf in the corresponding proof tree. This simulates an induction over n. First work on this has been presented at the PARIS workshop on July 7 & 8, 2018 in Oxford, UK where we defined a more suitable calculus and a first draft of an abstraction rule.

As mentioned in Section 5.2, the algorithm for computing $\mathbf{S}\Pi_2$-**G**s is only a prototype and requires further optimizations. In particular, due to the existence of non solvable Π_2-**SEHS**s, the discovery of properties of $\mathbf{S}\Pi_2$-**G**s that guarantee the solvability of the corresponding Π_2-**SEHS**s would improve the presented methods decisively. Moreover, a more reasonable implementation of the algorithm should exploit interaction between the grammar computation and the computation of a cut formula.

APPENDIX A

Problems of the TSTP

A.1 PUZ/PUZ035-5/Prover9—1109a.UNS-Ref.s

```
%--------------------------------------------------------------
% File     : PUZ035-5 : TPTP v7.2.0. Released v2.0.0.
% Domain   : Puzzles
% Problem  : Knights and Knaves #36
% Version  : [Sto95] axioms.
%            Theorem formulation : Definite answer "yes".
% English  : On an island, there live exactly two types of
%            people: knights and knaves. Knights always tell the
%            truth and knaves always lie. I landed on the
%            island, met two inhabitants, asked one of them: "Is
%            one of you a knight?" and he answered me. What can
%            be said about the types of the asked and the other
%            person depending on the answer I get?

% Refs     : [Smu78] Smullyan (1978), What is the Name of This
%                    Book? The Ri
%          : [Sto95] Stolzenburg (1995), Email to Geoff
%                    Sutcliffe.
%          : [BFS95] Baumgartner et al. (1995), Model
%                    Elimination, Logic Pr
%          : [BFS97] Baumgartner et al. (1997), Computing
%                    Answers with Mode
% Source   : [Sto95]
% Names    :
```

```
% Status   : Unsatisfiable
% Rating   : 0.00 v7.1.0, 0.17 v7.0.0, 0.12 v6.3.0, 0.00 v2.1.0
% Syntax   : Number of clauses     :    9 (   2 non-Horn;
%                                         1 unit;   6 RR)
%            Number of atoms       :   20 (   0 equality)
%            Maximal clause size   :    3 (   2 average)
%            Number of predicates  :    2 (   0 propositional;
%                                         1-2 arity)
%            Number of functors    :    6 (   4 constant;
%                                         0-2 arity)
%            Number of variables   :   14 (   4 singleton)
%            Maximal term depth    :    3 (   2 average)
% SPC      : CNF_UNS_RFO_NEQ_NHN

% Comments : Query allows for disjunctive answer
%            X/Y = knave/knave ; knight/knave ; knight/knight
%--------------------------------------------------------------------------
%----Everyone's either a knight or a knave
cnf(everyone_a_knight_or_knave,axiom,
    ( truth(isa(P,knight))
    | truth(isa(P,knave)) )).

cnf(not_both_a_knight_and_knave,axiom,
    ( ~ truth(isa(P,knight))
    | ~ truth(isa(P,knave)) )).

cnf(knights_make_true_statements1,axiom,
    ( truth(S)
    | ~ truth(isa(P,knight))
    | ~ says(P,S) )).

cnf(knights_make_true_statements2,axiom,
    ( truth(isa(P,knight))
    | ~ truth(S)
    | ~ says(P,S) )).

%----Definitions for or
cnf(or1,axiom,
    ( truth(A)
    | truth(B)
    | ~ truth(or(A,B)) )).

cnf(or2,axiom,
```

```
    ( truth(or(A,B))
    | ~ truth(A) )).

cnf(or3,axiom,
    ( truth(or(A,B))
    | ~ truth(B) )).

cnf(says_yes,axiom,
    ( says(asked,or(isa(asked,knight),isa(other,knight))) )).

cnf(query,negated_conjecture,
    ( ~ truth(isa(asked,X))
    | ~ truth(isa(other,Y)) )).

%--------------------------------------------------------------------
```

A.2 PUZ/PUZ035-6/Prover9—1109a.UNS-Ref.s

```
%--------------------------------------------------------------------
% File     : PUZ035-6 : TPTP v7.2.0. Released v2.0.0.
% Domain   : Puzzles
% Problem  : Knights and Knaves #36
% Version  : [Sto95] axioms.
%            Theorem formulation : Definite answer "no".
% English  : On an island, there live exactly two types of
%            people: knights and knaves. Knights always tell the
%            truth and knaves always lie. I landed on the
%            island, met two inhabitants, asked one of them: "Is
%            one of you a knight?" and he answered me. What can
%            be said about the types of the asked and the other
%            person depending on the answer I get?

% Refs     : [Smu78] Smullyan (1978), What is the Name of This
%                    Book? The Ri
%          : [Sto95] Stolzenburg (1995), Email to Geoff
%                    Sutcliffe.
%          : [BFS95] Baumgartner et al. (1995), Model
%                    Elimination, Logic Pr
%          : [BFS97] Baumgartner et al. (1997), Computing
%                    Answers with Mode
% Source   : [Sto95]
% Names    :
```

```
% Status   : Unsatisfiable
% Rating   : 0.00 v7.1.0, 0.17 v7.0.0, 0.12 v6.3.0, 0.14 v6.2.0,
%            0.00 v2.1.0
% Syntax   : Number of clauses     :   11 (   3 non-Horn;
%                                         1 unit;   7 RR)
%            Number of atoms       :   24 (   0 equality)
%            Maximal clause size   :    3 (   2 average)
%            Number of predicates  :    2 (   0 propositional;
%                                         1-2 arity)
%            Number of functors    :    7 (   4 constant;
%                                         0-2 arity)
%            Number of variables   :   16 (   4 singleton)
%            Maximal term depth    :    4 (   2 average)
% SPC      : CNF_UNS_RFO_NEQ_NHN

% Comments : Query allows for definite answer
%            X/Y = knight/knave.
%--------------------------------------------------------------------
%----Everyone's either a knight or a knave
cnf(everyone_a_knight_or_knave,axiom,
    ( truth(isa(P,knight))
    | truth(isa(P,knave)) )).

cnf(not_both_a_knight_and_knave,axiom,
    ( ~ truth(isa(P,knight))
    | ~ truth(isa(P,knave)) )).

cnf(knights_make_true_statements1,axiom,
    ( truth(S)
    | ~ truth(isa(P,knight))
    | ~ says(P,S) )).

cnf(knights_make_true_statements2,axiom,
    ( truth(isa(P,knight))
    | ~ truth(S)
    | ~ says(P,S) )).

%----Definitions for or
cnf(or1,axiom,
    ( truth(A)
    | truth(B)
    | ~ truth(or(A,B)) )).
```

```
cnf(or2,axiom,
    ( truth(or(A,B))
    | ~ truth(A) )).

cnf(or3,axiom,
    ( truth(or(A,B))
    | ~ truth(B) )).

%----Axioms for not
cnf(not1,axiom,
    ( truth(C)
    | truth(not(C)) )).

cnf(not2,axiom,
    ( ~ truth(C)
    | ~ truth(not(C)) )).

cnf(says_yes,axiom,
    ( says(asked,not(or(isa(asked,knight),isa(other,knight))))
      )).

cnf(query,negated_conjecture,
    ( ~ truth(isa(asked,X))
    | ~ truth(isa(other,Y)) )).

%--------------------------------------------------------------------
```

List of Figures

Index

Bibliography

[ACP09] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A Schemata Calculus for Propositional Logic. In Martin Giese and Arild Waaler, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 5607 of *Lecture Notes in Computer Science*, pages 32–46. Springer Berlin Heidelberg, 2009.

[ACP10] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A Decidable Class of Nested Iterated Schemata. In *Proceedings of the 5th international conference on Automated Reasoning*, IJCAR'10, pages 293–308, Berlin, Heidelberg, 2010. Springer-Verlag.

[ACP11a] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Decidability and Undecidability Results for Propositional Schemata. *Journal of Artificial Intelligence Research*, 40(1):599–656, 2011.

[ACP11b] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Linear Temporal Logic and Propositional Schemata, Back and Forth. In *Proceedings of the 2011 Eighteenth International Symposium on Temporal Representation and Reasoning*, TIME '11, pages 80–87, Washington, DC, USA, 2011. IEEE Computer Society.

[AEP13] Vincent Aravantinos, Mnacho Echenim, and Nicolas Peltier. A Resolution Calculus for First-Order Schemata. *Fundam. Inform.*, 125(2):101–133, 2013.

[AHL15] Bahareh Afshari, Stefan Hetzl, and Graham Emil Leigh. Herbrand Disjunctions, Cut Elimination and Context-Free Tree Grammars. In *13th International Conference on Typed Lambda Calculi and Applications, TLCA 2015, July 1-3, 2015, Warsaw, Poland*, pages 1–16, 2015.

[ALMP08] Josep Argelich, Chu-Min Li, Felip Manya, and Jordi Planes. The First and Second Max-SAT Evaluations. *Journal on Satisfiability, Boolean Modeling and Computation*, 4:251–278, 2008.

[AP11] Vincent Aravantinos and Nicolas Peltier. Schemata of SMT-Problems. In *Proceedings of the 20th international conference on Automated reasoning with*

analytic tableaux and related methods, TABLEAUX'11, pages 27–42, Berlin, Heidelberg, 2011. Springer-Verlag.

[BBHI05] Alan Bundy, David Basin, Dieter Hutter, and Andrew Ireland. *Rippling: Meta-Level Guidance for Mathematical Reasoning*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2005.

[BBS97] Franco Barbanera, Stefano Berardi, and Massimo Schivalocchi. "Classical" programming-with-proofs in ${\lambda_{PA}}^{Sym}$: An analysis of non-confluence. In *Theoretical Aspects of Computer Software, Third International Symposium, TACS '97, Sendai, Japan, September 23-26, 1997, Proceedings*, pages 365–390, 1997.

[BHL+04] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Cut-Elimination: Experiments with CERES. In *Logic for Programming, Artificial Intelligence, and Reasoning, 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14-18, 2005, Proceedings*, pages 481–495, 2004.

[Bir67] Garrett Birkhoff. *Lattice Theory*, volume 25. American Mathematical Society, 1967.

[Bun01] Alan Bundy. The Automation of Proof by Mathematical Induction. In Andrei Voronkov and John Alan Robinson, editors, *Handbook of Automated Reasoning*, volume 1, pages 845–911. Elsevier, 2001.

[Bus95] Samuel R. Buss. On Herbrand's theorem. In *Logic and Computational Complexity*, pages 195–209. Springer, 1995.

[CDG+08] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree Automata: Techniques and Applications. *http://www.grappa.univ-lille3.fr/tata*, 2008.

[Cer14] David M. Cerna. A Tableaux-Based Decision Procedure for Multi-Parameter Propositional Schemata. In *Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings*, pages 61–75, 2014.

[Cho56] Noam Chomsky. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124, 1956.

[CL17] David M. Cerna and Michael Lettmann. Integrating a Global Induction Mechanism into a Sequent Calculus. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 278–294. Springer, 2017.

[Col01] Simon Colton. *Automated Theory Formation in Pure Mathematics*. PhD thesis, University of Edinburgh, 2001.

[Col02] Simon Colton. *Automated Theory Formation in Pure Mathematics*. Springer, 2002.

[Cra57] William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, 22(3):269–285, 1957.

[Ded87] Richard Dedekind. Was sind und was sollen die Zahlen? *Friedrich Vieweg & Sohn, Braunschweig*, 1939 (1887).

[DFG08] Marcello D'Agostino, Marcelo Finger, and Dov Gabbay. Cut-Based Abduction. *Logic Journal of the IGPL*, 16:537–560, 2008.

[DFG13] Marcello D'Agostino, Marcelo Finger, and Dov Gabbay. Semantics and proof-theory of depth-bounded Boolean logics. *Theoretical Computer Science*, 480:43–68, 2013.

[DLRW13] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Cut-Elimination and Proof Schemata. In *Logic, Language, and Computation - 10th International Tbilisi Symposium on Logic, Language, and Computation, TbiLLC 2013, Gudauri, Georgia, September 23-27, 2013. Revised Selected Papers*, pages 117–136, 2013.

[Dra87] Albert G. Dragálin. *Mathematical intuitionism: Introduction to proof theory*. American Mathematical Soc., 1987.

[EEH17] Sebastian Eberhard, Gabriel Ebner, and Stefan Hetzl. Algorithmic compression of finite tree languages by rigid acyclic grammars. *ACM Transactions on Computational Logic (TOCL)*, 18(4):26:1–26:20, 2017.

[EH15] Sebastian Eberhard and Stefan Hetzl. Inductive theorem proving based on tree grammars. *Annals of Pure and Applied Logic*, 166(6):665–700, 2015.

[EH18] Sebastian Eberhard and Stefan Hetzl. On the Compressibility of Finite Languages and Formal Proofs. *Information and Computation*, 259:191–213, 2018.

[EHL+18] Gabriel Ebner, Stefan Hetzl, Alexander Leitsch, Giselle Reis, and Daniel Weller. On the Generation of Quantified Lemmas. *Journal of Automated Reasoning*, pages 1–32, 2018.

[EHR+16] Gabriel Ebner, Stefan Hetzl, Giselle Reis, Martin Riener, Simon Wolfsteiner, and Sebastian Zivota. System description: GAPT 2.0. In *8th International Joint Conference on Automated Reasoning, IJCAR*, pages 293–301, 2016.

[Fan91] Gino Fano. *Sui postulati fondamentali della geometria proiettiva in uno spazio lineare a un numero qualunque di dimensioni*. 1891.

[FG07] Marcelo Finger and Dov Gabbay. Equal Rights for the Cut: Computable Non-analytic Cuts in Cut-based Proofs. *Logic Journal of the IGPL*, 15(5–6):553–575, 2007.

[Fre79] Gottlob Frege. *Begriffsschrift. Eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.* 1879.

[Fre84] Gottlob Frege. *Die Grundlagen der Arithmetik. Eine logisch mathematische Untersuchung über den Begriff der Zahl.* Verlag von Wilhelm Koebner, 1884.

[Gen35a] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1934-1935.

[Gen35b] Gerhard Gentzen. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112:493–565, 1935.

[Göd29] Kurt Gödel. *Über die Vollständigkeit des Logikkalküls.* PhD thesis, 1929.

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.

[GS97] Ferenc Gécseg and Magnus Steinby. Tree Languages. In *Handbook of formal languages*, pages 1–68. Springer, 1997.

[Häh01] Reiner Hähnle. Tableaux and Related Methods. In *Handbook of Automated Reasoning (in 2 volumes)*, pages 100–178. Elsevier, 2001.

[Her30] Jacques Herbrand. Recherches sur la théorie de la démonstration. 1930.

[Her95] Hugo Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de lambda-termes et comme calcul de stratégies gagnantes. (Computing with sequents: on the interpretation of sequent calculus as a calculus of lambda-terms and as a calculus of winning strategies).* PhD thesis, Paris Diderot University, France, 1995.

[Het11] Stefan Hetzl. Proofs as tree languages. 2011.

[Het12] Stefan Hetzl. Applying Tree Languages in Proof Theory. In Adrian-Horia Dediu and Carlos Martìn-Vide, editors, *Language and Automata Theory and Applications*, pages 301–312. Springer, 2012.

[Hil99] David Hilbert. *Grundlagen der Geometrie.* B. G. Teubner, 4th edition, 1913 (1899).

[Hil00] David Hilbert. Mathematische Probleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1900:253–297, 1900.

[HLR+14] Stefan Hetzl, Alexander Leitsch, Giselle Reis, Janos Tapolczai, and Daniel Weller. Introducing quantified cuts in logic with equality. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Automated Reasoning*, volume 8562 of *LNAI*, pages 240–254. Springer, 2014.

[HLRW14] Stefan Hetzl, Alexander Leitsch, Giselle Reis, and Daniel Weller. Algorithmic introduction of quantified cuts. *Theoretical Computer Science*, 549:1–16, 2014.

[HLW12] Stefan Hetzl, Alexander Leitsch, and Daniel Weller. Towards algorithmic cut-introduction. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 228–242. Springer, 2012.

[IB96] Andrew Ireland and Alan Bundy. Productive Use of Failure in Inductive Proof. *Journal of Automated Reasoning*, 16(1–2):79–111, 1996.

[JKV09] Florent Jacquemard, Francis Klay, and Camille Vacher. Rigid Tree Automata. In *LATA*, pages 446–457. Springer, 2009.

[Ken72] Hubert C Kennedy. The origins of modern axiomatics: Pasch to Peano. *The American mathematical monthly*, 79(2):133–136, 1972.

[Kle09] Stephen Cole Kleene. *Introduction to Metamathematics*. Ishi Press, 2009.

[Koh08] Ulrich Kohlenbach. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*. Springer Science & Business Media, 2008.

[Kun14] Kenneth Kunen. *Set theory: An introduction to independence proofs*, volume 102. Elsevier, 2014.

[KY00] John C Kieffer and En-Hui Yang. Grammar-Based Codes: a New Class of Universal Lossless Source Codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000.

[Lei15] Alexander Leitsch. On proof mining by cut-elimination. *Mathematical Logic and Foundations*, 55:173–200, 2015.

[LL18] Alexander Leitsch and Michael P. Lettmann. The problem of Π_2-cut-introduction. *Theor. Comput. Sci.*, 706:83–116, 2018.

[LM99] N. Jesper Larsson and Alistair Moffat. Off-Line Dictionary-Based Compression. In *Data Compression Conference, DCC 1999, Snowbird, Utah, USA, March 29-31, 1999.*, pages 296–305, 1999.

[LMG94] Reinhold Letz, Klaus Mayr, and Christoph Goller. Controlled Integration of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, 13(3):297–337, 1994.

[LP18] Michael P. Lettmann and Nicolas Peltier. A Tableaux Calculus for Reducing Proof Size. In *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, pages 64–80, 2018.

[LPW17] Alexander Leitsch, Nicolas Peltier, and Daniel Weller. CERES for First-Order Schemata. *J. Log. Comput.*, 27(7):1897–1954, 2017.

[NMW97] Craig G. Nevill-Manning and Ian H. Witten. Identifying Hierarchical Structure in Sequences: A Linear-Time Algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.

[Pas82] Moritz Pasch. Vorlesungen über neuere Geometrie, 1882.

[Pea89a] Giuseppe Peano. *Arithmetices Principia: Nova methodo exposita*. Fratres Bocca, 1889.

[Pea89b] Giuseppe Peano. *I principii di geometria logicamente esposti*. Fratelli Bocca, 1889.

[Sch77] Helmut Schwichtenberg. Proof Theory: Some Applications of Cut-Elimination. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 867–895. Elsevier, 1977.

[SS82] James A. Storer and Thomas G. Szymanski. Data Compression via Textual Substitution. *J. ACM*, 29(4):928–951, 1982.

[Sut09] Geoff Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[Tak67] Gaisi Takeuti. Consistency proofs of subsystems of classical analysis. *Annals of mathematics*, pages 299–348, 1967.

[Tak87] Gaisi Takeuti. *Proof theory*. Studies in Logic and the Foundations of Mathematics 81. North-Holland, 2nd edition, 1987.

[Tao07] Terence Tao. Soft analysis, hard analysis, and the finite convergence principle. *Essay posted May*, 23, 2007.

[TS96] Anne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, The Pitt Building, Trumpington Street, Cambridge, United Kingdom, second edition, 1996.

[Tse68] G. S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constrained Mathematics and Mathematical Logic*, 1968.

[Urb00] Christian Urban. *Classical Logic and Computation.* PhD thesis, 2000.

[VSU10] Jiři Vyskočil, David Stanovský, and Josef Urban. Automated Proof Compression by Invention of New Definitions. In E. M. Clark and A. Voronkov, editors, *Logic for Programming, Artifical Intelligence and Reasoning (LPAR-16)*, volume 6355 of *Lecture Notes in Computer Science*, pages 447–462. Springer, 2010.

[WP10] Bruno Woltzenlogel Paleo. Atomic Cut Introduction by Resolution: Proof Structuring and Compression. In E. M. Clark and A. Voronkov, editors, *Logic for Programming, Artifical Intelligence and Reasoning (LPAR-16)*, volume 6355 of *Lecture Notes in Computer Science*, pages 463–480. Springer, 2010.