FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Automatische Generierung Analytischer Kalküle für Involutative Logiken

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Logic and Computation

eingereicht von

Ardit Ymeri
Matrikelnummer 01229540

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof.Dr Agata Ciabattoni
Mitwirkung: Lara Spendier, PhD

Wien, 7. Oktober 2019

Ardit Ymeri	Agata Ciabattoni

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Automatic Generation of Analytic Calculi for Involutive Logics.

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Logic and Computation

by

Ardit Ymeri
Registration Number 01229540

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof.Dr Agata Ciabattoni
Assistance: Lara Spendier, PhD

Vienna, 7th October, 2019

Ardit Ymeri | Agata Ciabattoni

Erklärung zur Verfassung der Arbeit

Ardit Ymeri
Address

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, 7. Oktober 2019

Ardit Ymeri

Danksagung

An erster Stelle möchte ich der Betreuerin meiner Masterarbeit, Prof. Agata Ciabattoni, einen großen Dank dafür aussprechen, daß sie mich mit dem Thema der Arbeit bekannt gemacht hat. Sie war unermüdlich darin, mir die Konzepte der Beweistheorie sowie versteckte Kniffe in komplizierten Beweisen zu erklären. Bei jedem Schritt auf dem Weg zu dieser Masterarbeit war sie uneingeschränkt da um mich zu betreuen, obwohl ich ihre Geduld auf eine harte Probe gestellt habe.

Weiters möchte ich mich bei Lara Spendier für ihre Durchsicht meines Programmcodes bedanken. Ihre frühere Arbeit auf demselben Gebiet bildete den Rahmen für meine Implementierung und bot Anschauungsmaterial, von dem ich viel lernen konnte.

Meine Familie spielte eine zentrale Rolle in meiner Ausbildung. Sie gab mir jede Unterstützung, die ich brauchte, und hat mich immer darin bestärkt weiterzumachen. Jedes Mitglied meiner Familie war mir ein gutes Beispiel.

Zu guter Letzt möchte ich allen meinen Freunden danken, die mit mir zusammen in Wien studiert haben. Es war mir eine Freude diese großartigen Menschen kennenzulernen, die immer das Allgemeinwohl im Sinne haben.

Acknowledgements

I want to give a big thank you to my professor and adviser Agata Ciabattoni for bringing me to this topic. She was never tired of explaining me the concepts of proof theory and clarifying the hidden tricks on puzzling proofs. She has always been there to supervise me on every step of my thesis. I have been a hard test for her patience and it is amazing how well she did on that.

I would like to express my gratitude to Lara Spendier for having a second pair of eyes on my code. Her previous work on the same area has served as a framework for my implementation and also as state of the art examples where I learned a lot.

My family played a vital role in my education. They provided me all the support I needed and motivated me to always go further. Each member of my family has been a good example for me to follow.

Last but not least, I would like to thank all my friends who studied with me in Vienna. It has been a pleasure to get to know such good people that are always aiming to contribute to the common good.

Kurzfassung

Die Nachfrage nach nichtklassischen Logiken ist im Laufe der letzten Jahrzehnte stark gestiegen. Insbesondere fanden derartige Logiken Anwendungen in verschiedenen Bereichen der Informatik, wie bespielsweise der Künstlichen Intelligenz, der Programmverifikation, der Wissensrepräsentation oder der Funktionalen Programmierung. Im Gegensatz zur Klassischen Logik erlauben sie oft die Behandlung von Situationen mit inkonsistenter oder unscharfer Information. Von besonderem Interesse für die Informatik sind substrukturelle Logiken, also axiomatische Erweiterungen des Lambek-Kalküls **FL**. Sie verdanken ihren Namen der Tatsache, daß ihnen in der Darstellung im Sequentialkalkül gewisse strukturelle Regeln fehlen (Vertauschung, Verdünnung und Zusammenziehung). Die Familie der substrukturellen Logiken umfasst eine große Bandbreite an nichtklassischen Logiken, wie beispielsweise intuitionistische Logik, lineare Logik, Fuzzylogik oder intermediäre Logiken.

Eine gebräuchliche Art der Beschreibung von nichtklassischen Logiken ist als Erweiterung bekannter Logiken durch Hilbert Axiome. Die Anwendbarkeit einer Logik ist allerdings stark abhängig von dem Vorhandensein eines entsprechenden analytischen Kalküls, also eines Schlußkalküls, in welchem Beweissuche durch schrittweise Zerlegung der zu beweisenden Formeln erfolgt. Diese Eigenschaft ermöglicht die Implementierung analytischer Kalküle als Grundlage automatischer Beweissuchprogramme, sowie den Beweis (meta-)mathematischer Eigenschaften wie Widerspruchsfreiheit und Entscheidbarkeit. Typischerweise wird ein analytischer Kalkül für eine Logik in zwei Schritten gewonnen: i) Zunächst wird der passende Kalkül gefunden und Vollständigkeit sowie Korrektheit bewiesen; ii) Dann wird mittels eines Schnitteliminationsbeweises gezeigt, daß der Kalkül analytisch ist. Beide Schritte sind oft sehr arbeitsintensiv, weshalb computergestützte Methoden zur Einführung analytischer Kalküle sehr gefragt sind.

Eine systematische Methode zur automatischen Erzeugung von analytischen Kalkülen für eine Vielzahl nichtklassischer Logiken wurde in [CGT08] vorgestellt. Die Methode wurde in Folge auf substrukturelle, intermediäre und parakonsistente Logiken erweitert [CLSZ13, CMS13] und als verschiedene in dem System Tools for the Investigation of Nonclassical Logics (TINC) implementiert. In der vorliegenden Arbeit erweitern wir TINC um das neue Programm InvAxiomCalc. Dieses erlaubt die Behandlung einer großen Klasse von Erweiterungen der multiplikativen additiven linearen Logik MALL, also eines Fragmentes der klassischen linearen Logik ohne exponentials. InvAxiomCalc basiert

auf dem Artikel [CST09], in welchem eine systematische Methode zur Konstruktion von Kalkülen für axiomatische Erweiterungen von MALL sowie ein allgemeiner Ansatz für Schnitteliminationsbeweise eingeführt wurden. InvAxiomCalc akzeptiert als Eingabe ein Axiom in der Sprache von MALL und generiert, wenn möglich, einen mittels LaTeXgesetzten Artikel, welcher den analytischen Kalkül für die axiomatische Erweiterung von MALL mit dem Eingabeaxiom beinhaltet.

Abstract

During the last decades there have been an increasing demand for non-classical logics. They have found application in various fields of computer science like Artificial Intelligence, Software Verification, Knowledge Representation or Functional Programming. Unlike classical logic, they are often capable of reasoning in situations with inconsistent or vague information. Of particular interest in computer science are substructural logics, which are axiomatic extensions of full Lambek Calculus **FL**. Their name is due to the fact that when expressed as sequent calculi, they lack some of the structural rules (exchange, weakening and contraction). The family of substructural logics encompasses a wide set of nonclassical logics such as intuitionistic, linear, fuzzy, intermediate logics, and more.

A common way of defining a nonclassical logic is by adding Hilbert axioms to well known systems. The applicability of a logic depends heavily on the existence of a corresponding analytic calculus, i.e. a deductive system where the proof search is performed by stepwise decomposition of the formulas to be proved. Analytic calculi are suitable for being implemented in computational proof search algorithms and for establishing mathematical properties of a logic such as consistency or decidability. Typically, an analytic calculus for a logic is obtained in two steps: i) finding the calculus that represent the features of the logic and showing that it is sound and complete, ii) showing that the calculus is analytic by providing a proof of the cut elimination theorem. These steps often require a laborious investigation, and therefore computer support tools for introducing analytic calculi are very desirable.

A systematic procedure which allows for automated generation of analytic calculi for a wide range of propositional nonclassical logics is introduced in [CGT08]. It is followed by a series of theoretical research over substructural, intermediate and paraconsistent logics [CLSZ13, CMS13] whose results are implemented in the Tools for the Investigation of Nonclassical Logics (TINC) system. In this thesis we extend TINC with a new tool called InvAxiomCalc, which provides support for a large set of logics extending Multiplicative Additive Linear Logic (MALL), i.e. a fragment of classical linear logic without exponentials. InvAxiomCalc is developed along the guidelines of [CST09] which provides a systematic procedure for generating calculi that represent the features of axiomatic extensions of MALL and a general approach for proving the cut elimination theorem. InvAxiomCalc accepts an axiom in MALL and generates, if possible, a paper written in LaTeX with the analytic calculus of MALL extended with the input axiom.

Contents

CHAPTER 1

Introduction

Nonclassical logics are deviations of classical logic which allow for reasoning in presence of inconsistencies, vague information, multiple truth values, dealing with time, resources, and more. Of particular interest in computer science are substructural logics. They encompass a large family of nonclassical logics such as intuitionistic, fuzzy, linear, intermediate, relevant logics, ect. During the recent decades, substructural logics have been intensively investigated as they provide languages for modeling data structures and resources. Additionally, computer support tools have been developed to make the theoretical results more accessible to researches and practitioners. For instance, MUltlog [BFSZ96] is a tool that allows for generation of calculi for many-valued logics. TINC [CS14] (Tools for the Investigation of Nonclassical Logics) is another system developed along the lines of MUltlog and it consists of a set of tools that allow for generating sequent-style calculi for large classes of substructural, intermediate and paraconsistent logics.

Nonclassical logics are often defined as systems of reasoning based on a set of axioms and inference rules which compose a *proof system* or a *calculus*. The valid statements of a logic are the set of all formulas that can be proved from its calculus. A convenient framework for describing a logic is the Hilbert system which usually consists of a large number of axioms and *modus ponens* as the only inference rule. A logic may have more than one calculus. For instance, sequent calculus introduced by Gentzen in [Gen35], is a widely used framework usually consisting of a single axiom and a large number of inference rules. Unlike Hilbert systems, the inference rules in sequent calculus operate over sets of formulas, called sequents, rather than single formulas. A generalised version of the sequent calculus is the hypersequent calculus [Avr87] where the expressive power is extended by letting the inference rules operate over sets of sequents instead of single sequents. Sequent and hypersequent calculi also contain a reformulation of the modus ponens, called the *cut rule* that is the only rule breaking the subformula property, which states that every formula in the premises is a subformula in the conclusion of the rule.

If all proofs in a calculus consist of step-wise decomposition of the formula to be proved, the calculus is called *analytic*. The applicability of a logic depends heavily on the availability of a corresponding analytic calculi. While a Hilbert system is convenient for describing the features of a logic, an analytic calculus is suitable when it comes to automated proof reasoning or the investigation of mathematical properties of a logic such as decidability or consistency. Generally, an analytic calculus can be obtained from a sequent calculus by showing that the cut rule is eliminable, i.e. every rule in the system can be transformed into a proof that does not use the cut rule. Finding a suitable calculus, showing that it is sound and complete, and then converting it to an analytic calculus by proving cut-elimination, often requires a lot of effort. Therefore, it is desirable to have an automated way to generate calculi for logics having already the analyticity property.

A systematic procedure for generating analytic calculi for a wide set of extensions of classical propositional logics is introduced in [CGT08, CGT12]. The procedure is based on the substructural hierarchy which is a classification of the Hilbert axioms according to the polarity of their logical connectives. In the first step, the procedure accepts an axiom up to a certain level of the substructural hierarchy and generates an equivalent set of (hyper)sequent rules. In a second step, the set of (hyper)sequent rules is transformed into equivalent analytic rules. The generated analytic rules together with the (hyper)sequent calculus of the base logic under consideration consist the (hyper)sequent calculus of the extended logic. The systematic procedure mentioned above is implemented in one of the tools in TINC called AxiomCALC which transforms any suitable axiomatic extension of Full Lambek calculus with exchange and weakening FLew (i.e. intuitionistic linear logic with weakening) into a cut-free (hyper)sequent calculus.

The subject of this thesis is to extend TINC with the implementation of the procedure introduced in [CST09] for transforming axiomatic suitable extensions of Multiplicative Additive Linear Logic without exponentials [Gir87] (i.e. classical linear logic without exponentials) into a cut-free sequent or hypersequent calculus. The implementation consists of a new tool in TINC called InvAxiomCalc which is developed in Prolog.

1.1 Overview

The thesis is organized as follows. Chapter 2 summarizes the preliminary concepts that are used throughout the thesis. We start with a brief description of the language and semantics of linear logic. Then, we proceed with the notions of Gentzen-style proof theory like sequent and hypersequent calculus as well as the cut elimination theorem. In Chapter 3 we provide a more detailed description of the algorithm in [CST09] to be implemented. Each step is illustrated with examples that cover a reasonable amount of corner cases and also suggests the implementation techniques. We start by describing the algorithm that accepts an axiom in the language of MALL and determines the class in the substructural hierarchy where the axiom belongs to. Afterwards, we provide the details of all the required steps for transforming an axiom to a set of analytic rules.

This will be possible only if the axiom resides into certain classes of the hierarchy. Chapter 4 starts with a gentle introduction to Prolog and the language features that we use in the implementation of InvAxiomCalc. We proceed with an architectural description of the existing tools in TINC, and then we provide detailed information about the implementation of InvAxiomCalc. The chapter concludes with a set of related logic engineering tools. Chapter 5 provides a summary of the thesis contents and the implemented tool InvAxiomCalc.

CHAPTER 2

Preliminaries.

The purpose of this chapter is to provide an introduction to the basic concepts of the Gentzen style proof theory of a fragment of classical linear logic without exponentials, also known as Multiplicative Additive Linear Logic (MALL). We start by introducing the language of MALL [Gir87, CST09] and briefly describing its logical connectives. Afterwards, in Section 2.2 we describe the notions of the sequent calculus LJ and the hypersequent calculus HLJ as introduced in [Gen35, Avr87] for Intuitionistic Logic (IL). Section 2.3 gives a sequent and a hypersequent calculus for MALL [CST09, Gir87]. We close this chapter with Section 2.4 which discusses analytic calculi and how to obtain them.

2.1 The Syntax of MALL

The language that we use for MALL, consists of infinitely many (possibly indexed) *propositional variables* $\mathcal{V} = \{a, b, c, \ldots\}$, their duals $\mathcal{V}^\perp = \{a^\perp, b^\perp, c^\perp, \ldots\}$, the constants $\{\bot, \top, 1, 0\}$, and the logical connectives $\{\&, \parr, \otimes, \oplus, \multimap\}$. The formulas on this language are generated by the grammar:

$$\mathcal{F} ::= \mathcal{V} \mid \mathcal{V}^\perp \mid \bot \mid \top \mid 1 \mid 0 \mid \mathcal{F} \parr \mathcal{F} \mid \mathcal{F} \,\&\, \mathcal{F} \mid \mathcal{F} \otimes \mathcal{F} \mid \mathcal{F} \oplus \mathcal{F} \mid \mathcal{F} \multimap \mathcal{F} \tag{2.1}$$

The capital letters A, B, C, ... are used to denote the formulas generated by the above grammar. We write $A \circ\!\!-\!\!\circ B$ as an abbreviation of $(A \multimap B) \,\&\, (B \multimap A)$, and $A_{\&1}$ as an abbreviation of $A \,\&\, 1$. The Greek letters ϕ, ψ, ... are used to denote schematic formulas (schemes), which are generated by a grammar similar to (2.1), but starting from formula variables instead of propositional variables. A scheme ϕ is called atomic, if $\phi = A$ or $\phi = A^\perp$ for some formula variable A. The capital letters A, B, C, ... are also used to denote atomic schemes.

2.1.1 The Linear Implication and the Exponentials

In classical and intuitionistic logic, by knowing that the propositions A and $A \supset B$ both hold, we can derive B without affecting the truth values of any of the premises A or $A \supset B$. The truth values which are not affected by the application of the logical connectives are also called stable truths, see Chapter 1 in [Gen35]. However, in real world, the implications are often *causal*, i.e. the condition is modified after its use. The process of modification of the premises is known as *reaction*.

For example, let A represent the action of spending 1€ on a Coke and B represent the action of getting it. In this case, the reaction of $A \supset B$ is that 1€ is spent during the process and cannot be used a second time. The symbol $\multimap$ (*linear implication*) is used in linear logic for representing the causal implication.

Linear logic is also equipped with two connectives "!" (*of course*) and "?" (*why not*), called *exponentials*, for expressing the iterability of the actions. These exponentials roughly correspond to the modalities $\Box$ and $\Diamond$, eg [CZ97]. We can think of "!" as a free duplication of an action and "?" as a discarding thereof. For instance $!A$ would mean to spend as many Euros as one needs [Gir87]. In this thesis we will not consider "!" and "?".

The intuitionistic implication $A \supset B$ can still be written by using $\multimap$ and the exponentials as $(!A) \multimap B$, which means that B is caused by some iterations of A. However, the subject of this thesis considers only the fragment of Linear Logic which excludes the exponentials. In this way, the syntax still looks similar to Intuitionistic Logic, but having additional logical connectives as explained in the following sections.

2.1.2 The Two Conjunctions

There are two conjunctions in Linear Logic, namely $\otimes$ (times) and $\&$ (with). They correspond to different meanings of the word "*and*". Both represent the availability of two actions. The difference is that in the case of $\otimes$ both of the actions will be performed, whereas in case of $\&$ only one of them will. We can chose to perform either of the actions, but not both of them simultaneously.

Let, A, B and C be three actions such that $A \multimap B$ and $A \multimap C$. Given only this information, there is no way of getting $A \multimap B \otimes C$ (we can only get $A \otimes A \multimap B \otimes C$). However, it is possible to get the action $A \multimap B \& C$, which is the superimposition of both B and C. Hence, the connective $\&$ has also disjunctive features, but it is technically not a disjunction, as $A \& B \multimap A$ and $A \& B \multimap B$ are both provable, see Example 2.3.2.

2.1.3 The Two Disjunctions

In analogy with the two conjunctions described above, linear logic is also equipped with two disjunctions:

- $\oplus$ (*plus*) is the dual of & (*with*) and expresses a *non-deterministic* choice of one action between two possible types. Note that in & we are able to chose which action shall be performed, but in $\oplus$ the choice is non-deterministic.
- $\parr$ (*par*) is the dual of $\otimes$ (*times*) and expresses the dependency between two types of actions. $A \parr B$ can either be read as $A^\perp \multimap B$ or $B^\perp \multimap A$, which means that $\parr$ can be seen as the symmetric form of the linear implication $\multimap$.

2.1.4 The Linear Negation

An important connective in MALL is the linear negation $(\cdot)^\perp$ (*nil*) which is the only negative operation of the logic.

nil is an involutive[1] operation which means that $A^{\perp\perp}$ is equivalent to A, similarly as in classical logic. Therefore, provides possibilities to apply the De Morgan-like laws to all of the connectives of the language. For example:

$$\begin{array}{rcl} (A \,\&\, B)^\perp & \Leftrightarrow & A^\perp \oplus B^\perp \\ (A \parr B)^\perp & \Leftrightarrow & A^\perp \otimes B^\perp \end{array}$$

2.2 Gentzen Style Proof Theory

Proof theory studies the formalization and the structure of mathematical proofs. One of the initiators of the proof theory is David Hilbert. In his attempts to provide a proof for the consistency of mathematics, Hilbert suggested proof systems consisting of a set of axioms (or axiom schemes) and a small number of inference rules, e.g. modus ponens [Bus98]. However, Hilbert systems are not suitable for computational proof search, as they are not analytic (see Section 2.4). In [Gen35], Gentzen introduced the sequent calculi LK and LJ as formal proof systems for classical and intuitionistic logic. Being analyitc, such proof systems turned out to be more suitable for automated proof search. This section provides a short introduction to the sequent and hypersequent calculus as introduced by Gentzen [Gen35] and Avron [Avr87].

2.2.1 The Sequent Calculus

The sequent calculus was introduced by Gentzen [Gen35] as a formalization of proofs in classical and intuitionistic logic. The main difference from the other proof systems (like Natural Deduction and Hilbert-style calculus) is that, instead of manipulating a single formula, sequent calculus operates on structures consisting of sets of formulas, called sequents:

Definition 2.2.1. *A sequent is an expression of the form* $\Gamma \vdash \Delta$ *where* Γ *and* Δ *are multisets of formulas.* Γ *is called the antecedent and* Δ *the succedent of the sequent. If the succedent of the sequent consist of at most one formula, then the sequent is called single-conclusion. Otherwise, it is called multi-conclusion sequent.*

[1] In Mathematics, a function is called involutive if it is the inverse of its own.

$$\phi \vdash \phi \qquad \frac{\Gamma \vdash \phi_i}{\Gamma \vdash \phi_1 \vee \phi_2}\,(\vee_i, r) \qquad \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \supset \psi}\,(\supset, r) \qquad \frac{\Gamma \vdash \phi \qquad \Gamma, \psi \vdash \Pi}{\Gamma, \phi \supset \psi \vdash \Pi}\,(\supset, l)$$

$$\bot \vdash \phi \qquad \frac{\Gamma, \phi_i \vdash \Pi}{\Gamma, \phi_1 \wedge \phi_2 \vdash \Pi}\,(\wedge_i, l) \qquad \frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\,(\wedge, r) \qquad \frac{\Gamma, \phi \vdash \Pi \qquad \Gamma, \psi \vdash \Pi}{\Gamma, \phi \vee \psi \vdash \Pi}\,(\vee, l)$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \phi}\,(w, r) \qquad \frac{\Gamma \vdash \Pi}{\Gamma, \phi \vdash \Pi}\,(w, l) \qquad \frac{\Gamma, \phi, \phi \vdash \Pi}{\Gamma, \phi \vdash \Pi}\,(c, l) \qquad \frac{\Gamma \vdash \phi \qquad \phi, \Delta \vdash \Pi}{\Gamma, \Delta \vdash \Pi}\,(cut)$$

Table 2.1: The sequent calculus LJ for intuitionistic logic.

In classical and intuitionistic logic, a sequent is intuitively understood as the implication of the conjunction of all formulas in the antecedent to the disjunction of all formulas in the succedent.

A sequent calculus consists of a set of axioms and a set of rules. A rule in sequent calculus consists of a set of sequents $\{S_1, \ldots, S_n\}$ called the *premises* and a single sequent S called the *conclusion*. It written as

$$[\text{rule name}]\ \frac{S_1 \quad \ldots \quad S_n}{S}$$

and it means that the conclusion S is inferred from the sequents of the premises $S_1, \ldots, S_n$. Table 2.1 shows the rules and the axioms of the first sequent calculus for intuitionistic logic LJ, in [Gen35]. The notion of derivation is also extended to sequents:

Definition 2.2.2. *A derivation in a sequent calculus is a finite labelled tree with nodes labelled by sequents and a single root (called end sequent). The leaves are labelled with axioms and each node is connected with the (immediate) successor nodes (if any) according to the inference rules.*

Let C be a sequent calculus. We say that the sequent S is derivable if there is a derivation in C having S as the end sequent.

In a sequent calculus we distinguish between different types of rules:

- *Logical rules* are rules that introduce a logical connective. The logical rules in Table 2.1 are $(\supset, l)$, $(\supset, r)$, $(\wedge_i, l)$, $(\wedge, r)$, $(\vee, l)$, $(\vee_i, r)$. For example, considering the $(\supset, r)$ rule from LJ:

  $$(\supset, l)\ \frac{\Gamma \vdash \phi \qquad \Gamma, \psi \vdash \Pi}{\Gamma, \phi \supset \psi \vdash \Pi}$$

  A new formula $\phi \supset \psi$ is introduced in the left side of the conclusion, having $\supset$ as a logical connective, whose left and right hand side are formulas taken respectively

from each premise. The formula $\phi \supset \psi$, in $(\supset, l)$, is called the principal formula. The formulas ϕ and ψ in the premises from which the principal formula is derived, are called active formulas. The rest of the components in the premises that are not changed by the rule application, i.e. Γ, Π, are referenced to as respectively the left and the right *context* of the rule.

- *Structural rules* do not introduce logical connectives. They operate on the occurrences and positions of the formulas in a sequent. Examples of structural rules from LJ are weakening (w, l), (w, r); contraction (c, l) and exchange, for instance:

$$(e, l)\ \frac{\Gamma, \phi, \psi \vdash \Pi}{\Gamma, \psi, \phi \vdash \Pi}$$

- *The cut rule* - an important characteristic of this rule, is that it eliminates a formula (called the *cut formula*) from the premises. It corresponds to the introduction of intermediate steps to derivations, and it is often helpful to shorten the proofs. The *cut* rule can also simulate modus ponens (MP):

$$(MP)\ \frac{\phi \qquad \phi \supset \psi}{\psi}$$

to show this, we have to derive that $\vdash \psi$ from $\vdash \phi$ and $\vdash \phi \supset \psi$ by using *cut*. Indeed:

$$(cut)\ \frac{\vdash \phi \supset \psi \qquad (cut)\ \dfrac{\vdash \phi \qquad (\supset, l)\ \dfrac{\phi \vdash \phi \qquad (w, l)\ \dfrac{\psi \vdash \psi}{\phi, \psi \vdash \psi}}{\phi, \pi \supset \psi \vdash \psi}}{\phi \supset \psi \vdash \psi}}{\vdash \psi}$$

Generally in inference rules, a proof of the premises implies a proof of the conclusion. On some cases, a proof of the conclusion, also implies a proof for each premise. Such rules, are called invertible rules:

Definition 2.2.3. *Let $S_1, \ldots, S_n, S$ be sequents and r be a rule. r is invertible, if for each instance*

$$\frac{S_1 \quad \ldots \quad S_n}{S}$$

of r, whenever S is derivable then each S_i for $n = 1, \ldots, n$ is also derivable.

2.2.2 The Hypersequent calculus

The sequent calculus has been one of the most preferred frameworks for defining analytic calculi. However, it is not powerful enough to capture all interesting logics. This has motivated the introduction of many generalizations of the sequent calculus in the last 30 years. A prominent example is the hypersequent calculus introduced by Avron in [Avr87]. Instead of single sequents, it operates on sets of sequents called *hypersequents*:

Internal	$\dfrac{G \mid \Gamma \vdash}{G \mid \Gamma \vdash \phi}\,(w,r)$	$\dfrac{G \mid \Gamma \vdash \Pi}{G \mid \Gamma, \phi \vdash \Pi}\,(w,l)$	$\dfrac{G \mid \Gamma, \phi, \phi \vdash \Pi}{G \mid \Gamma, \phi \vdash \Pi}\,(c,l)$
External	$\dfrac{G}{G \mid \Gamma \vdash \Pi}\,(ew)$		$\dfrac{G \mid \Gamma \vdash \Pi \mid \Gamma \vdash \Pi}{G \mid \Gamma \vdash \Pi}\,(ec)$

Table 2.2: Internal and external structural rules in HLJ

Definition 2.2.4. *A hypersequent is a finite multiset* $\Gamma_1 \vdash \Delta_1 | \ldots | \Gamma_n \vdash \Delta_n$ *where each* $\Gamma_i \vdash \Delta_i$ *for* $i = 1, \ldots, n$ *is a sequent, called a component of the hypersequent. A hypersequent is called single-conclusion if all of its components are single-conclusion. Otherwise, it is called multiple conclusion.*

In classical and intuitionistic logic, the structural connective "|" that is used to separate the sequents is interpreted as a disjunction at the meta-level.

Similarly as in sequent calculus, the rules of a hypersequent calculus consist of axioms, logical rules, structural rules and the cut rule. The key difference, which also extends the expressibility of a hypersequent calculus compared to the sequent calculus, is that we can have rules that manipulate not only the formulas within one component of a hypersequent (*internal rules*), but also rules that manipulate the components of a hypersequent (*external rules*).

The hypersequent calculus HLJ for the intuitionistic logic can be obtained by adding a hypersequent context G to every rule in LJ Table 2.1 and by adding the external structural rules (ew) and (ec). Table 2.2 shows the internal and external structural rules of HLJ.

2.3 The hypersequent calculus HMALL

In Section 2.2 we described the notions of sequent and hypersequent calculus as introduced by Gentzen and Avron in [Gen35, Avr87] for intuitionistic logic LJ. However, the main topic of this thesis is the fragment of linear logic without the exponentials. This section provides a translation to linear logic of the notions given in Section 2.2. Further details, can be found in [Gir87, CST09].

Hypersequent calculus for Multiplicative Additive Linear Logic (HMALL) is the hypersequent system consisting of the set of inference rules shown in Table 2.3. Each rule in HMALL consists of sequents of formulas in the language of linear logic without exponentials. By removing ew and ec, and by dropping the hypersequent context $\mathcal{H}$ from every rule in HMALL, we obtain the sequent system MALL.

Each HMALL rule consists of a set of single sided multiple conclusion sequents. A definition of the single sided (hyper)sequents and their interpretation is given in Definition 2.3.1. Note that by sticking to the *single sided multiple conclusion* setting, we do not lose any expressive power of the *two sided single conclusion* setting (i.e. sequents of the form $\Gamma \vdash A$). The latter can be embedded to the former by using left/right polarities.

Definition 2.3.1. *[CST09] A single sided sequent is a finite multiset of formulas, usually written as* $\vdash A_1, ..., A_n$. *A single sided hypersequent* $\mathcal{H}$ *is a finite multiset of sequents written as* $\vdash \Gamma_1 | \ldots | \vdash \Gamma_n$. *The interpretation* $(\Gamma)^I$ *of a sequent* $\vdash \Gamma := \vdash A_1, \ldots, A_n$ *is the formula* $A_1 ⅋ \ldots ⅋ A_n$ *and* $(\vdash \Gamma)^I = \bot$, *if* $n = 0$. *The interpretation of the hypersequent* $\mathcal{H} = \vdash \Gamma_1 | \ldots | \vdash \Gamma_n$ *is defined as* $(\mathcal{H})^I = (\vdash \Gamma_1)^I_{\&1} \oplus \ldots \oplus (\vdash \Gamma_n)^I_{\&1}$.

From now on, the Greek letters Γ, Δ, Σ, ... are used to denote multisets of formulas, the calligraphic letters $\mathcal{H}$, $\mathcal{G}$, ... to denote hypersequents and the capital letters A, B, C, ... to denote single formulas. In inference rules, the letters Γ, Δ, Σ, ... will be referred to as *multiset variables*, while the letters A, B, C, ... will be referred to as *formula variables*.

Similarly as in the hypersequent calculus HLJ, the rules in HMALL are classified as:

- the *ax* rule - represents the identity principle, as it is equivalent to the two sided hypersequent rule: $\overline{\mathcal{H} | A \vdash A}$.
- *logical* rules - introducing a logical connective. Similarly as in HLJ, the logical rules of HMALL can be partitioned into *invertible rules*: $\&$, $⅋$, $\top$, $\bot$ and non-invertible rules: $\otimes$, $\oplus$, 1, 0.
- *structural* rules - the only structural rules included in HMALL are the external structural rules of weakening *ew* and contraction *ec*. Since linear logic emphasizes the role of the formulas as resources, the internal structural rules of contraction and weakening are in general *not allowed*.
- the *cut* rule - looks slightly different from the *cut* rule in HLJ (see Table 2.1) since in HMALL we only deal with single sided sequents. In HMALL, the cut formulas occur on the same side of the sequents, but they have different polarities. Similarly as in HLJ, the *cut* rule in HMALL also eliminates a formula from the premises.

The notation $\vdash_S A$ is used to denote that the formula A is derivable in the system S. Similarly, $\vdash_S \Gamma$ or $\vdash_S \mathcal{H}$ denote that the sequent Γ or the hypersequent $\mathcal{H}$ is derivable in the system S. If R is a set of rules, then we write $S + R$ to denote the system S extended with the rules in R. Furthermore, we write $\vdash_{S+R} A$ if the formula A is derivable in the system $S + R$.

Example 2.3.2. *The derivation of* $\vdash_{MALL} (A^\bot \oplus B^\bot) ⅋ A$ *is written as follows:*

$$ax\ \frac{}{\mathcal{H}|\vdash A, A^{\perp}} \qquad cut\ \frac{\mathcal{H}|\vdash \Gamma, A \qquad \mathcal{H}|\vdash A^{\perp}, \Delta}{\mathcal{H}|\vdash \Gamma, \Delta} \qquad ew\ \frac{\mathcal{H}}{\mathcal{H}|\vdash \Gamma}$$

$$ec\ \frac{\mathcal{H}|\vdash \Gamma|\vdash \Gamma}{\mathcal{H}|\vdash \Gamma} \qquad \otimes\ \frac{\mathcal{H}|\vdash \Gamma, A \qquad \mathcal{H}|\vdash B, \Delta}{\mathcal{H}|\vdash \Gamma, A \otimes B, \Delta} \qquad 1\ \frac{}{\mathcal{H}|\vdash 1}$$

$$\oplus_1\ \frac{\mathcal{H}|\vdash \Gamma, A}{\mathcal{H}|\vdash \Gamma, A \oplus B} \qquad \oplus_2\ \frac{\mathcal{H}|\vdash \Gamma, B}{\mathcal{H}|\vdash \Gamma, A \oplus B} \qquad \top\ \frac{}{\mathcal{H}|\vdash \Gamma, \top}$$

$$\parr\ \frac{\mathcal{H}|\vdash \Gamma, A, B}{\mathcal{H}|\vdash \Gamma, A\ \parr\ B} \qquad \&\ \frac{\mathcal{H}|\vdash \Gamma, A \qquad \mathcal{H}|\vdash \Gamma, B}{\mathcal{H}|\vdash \Gamma, A\ \&\ B} \qquad \perp\ \frac{\mathcal{H}|\vdash \Gamma}{\mathcal{H}|\vdash \Gamma, \perp}$$

Table 2.3: Hypersequent system HMALL

$$\parr\ \frac{\oplus_1\ \dfrac{ax\ \dfrac{}{\vdash A^{\perp}, A}}{\vdash A^{\perp} \oplus B^{\perp}, A}}{\vdash (A^{\perp} \oplus B^{\perp}) \parr A} \tag{2.2}$$

Similarly, we can show the derivability of $\vdash (A^{\perp} \oplus B^{\perp}) \parr B$ *by using* $\oplus_2$ *instead of* $\oplus_1$

Proposition 2.3.3 shows that if a sequent Γ is provable in HMALL extended with an axiom $\mathcal{G}$ (i.e. a rule having empty premises and the hypersequent $\mathcal{G}$ in the conclusion), then Γ is also provable in HMALL extended with the axiom $\mathcal{G}^I$ (i.e. the rule with empty premises having the sequent $\vdash \mathcal{G}^I$ in the conclusion).

Proposition 2.3.3. *[CST09] For any sequent* Γ *and hypersequent* $\mathcal{G}$*, we have that*

$$\vdash_{HMALL+\mathcal{G}} \Gamma \qquad \textit{iff} \qquad \vdash_{HMALL+\mathcal{G}^I} \Gamma.$$

The equivalence between two sets of rules S_1 and S_2 is defined based on the set of sequents that can be proved by extending a base (hyper)sequent system with S_1 and S_2. In particular:

Definition 2.3.4. *[CST09] Two sets of inference rules* S_1 *and* S_2 *are equivalent in* $(H)MALL$ *iff* $(H)MALL + S_1$ *and* $(H)MALL + S_2$ *prove the same sequents.*

Since an axiom ϕ can be considered as a rule without premises, the definition 2.3.4 applies also to the sets of axioms.

2.4 Analytic calculi and Cut elimination

An important rule in (hyper)sequent calculus systems is the *cut* rule:

$$cut\ \frac{\Gamma \vdash \phi \qquad \phi, \Delta \vdash \Pi}{\Gamma, \Delta \vdash \Pi} \qquad cut\ \frac{\mathcal{H}| \vdash \Gamma, A \qquad \mathcal{H}| \vdash A^{\perp}, \Delta}{\mathcal{H}| \vdash \Gamma, \Delta}$$

(a) – *cut* in LJ $\qquad\qquad$ (b) – *cut* in HMALL

cut is often useful to shorten the proofs. However, *cut* is the only rule which does not satisfy the subformula property which states that: every formula in the premises occurs as a subformula in the conclusion. Therefore, in a derivation tree having no occurrences of *cut*, every formula occurring on a leaf is a subformula on each of its ancestors, and therefore, also a subformula of the root, i.e. the end sequent. Such a property, is the key for the development of the automated reasoning methods and proof-theoretic applications. Therefore, it becomes important to remove all instances of the cut from derivations of a calculus system. If every rule in a calculus system satisfies the subformula property, then the system is referred to as *analytic*.

One way to show that a (hyper)sequent calculus system is analytic, is to show that the *cut* rule in this system is *admissible*. Which means that the presence of the *cut* does not add new derivable sequents in the system, and hence, it is closed under *cut*.

Another way to show that a system is analytic, is to prove the *cut elimination theorem* by showing that any derivation containing an application of a *cut* can be transformed into a cut-free derivations. Gentzen introduced in [Gen35] a proof of the cut elimination for LJ, which proceeds by a double induction on the complexity of the cut formula (i.e. the number of the logical connectives in it) and the number of consecutive sequents in the derivation that contain the cut formula. A general proof of the cut elimination theorem for the extensions of HMALL is given in [CST09].

CHAPTER 3

From Axioms to Analytic Rules

In this chapter we give a description of the systematic procedure introduced in [CST09] for transforming Hilbert axioms in the language of classical linear logic without exponentials into equivalent analytic inference rules in the multiple-conclusion (hyper)sequent calculi, which enjoys cut-elimination under certain conditions described through the chapter. The procedure is based on the substructural hierarchy defined over Hilbert axioms and uses HMALL as a base hypersequent system, see Section 2.3. Section 3.1 provides the definition of the substructural hierarchy. Sections 3.2 and 3.3 describe the systematic procedure respectively for the class $\mathcal{N}_2$ and $\mathcal{P}_3'$ of the substructural hierarchy. The final step of the transformation which converts a rule constructed from 3.2 and 3.3 into an analytic one, is given in Section 3.4.

3.1 Substructural Hierarchy

The substructural hierarchy is defined based on the invertibility of the logical inference rules in HMALL and consists of sets of axioms $\mathcal{P}_n$, $\mathcal{N}_n$ for $n \geq 0$. Recall that the rules $\&$, $\parr$, $\bot$ and $\top$ in Table 2.3 are invertible, whereas $\oplus$, $\otimes$, 1 and 0 are not.

Let $\mathcal{A}$ denote single proposition variables and their duals. The classes $\mathcal{P}_n$ and $\mathcal{N}_n$ of the substructural hierarchy are defined by the following grammar:

$$\begin{array}{ll} \mathcal{P}_0 ::= \mathcal{A} & \mathcal{P}_{n+1} ::= \mathcal{N}_n \mid \mathcal{P}_{n+1} \otimes \mathcal{P}_{n+1} \mid \mathcal{P}_{n+1} \oplus \mathcal{P}_{n+1} \mid 1 \mid 0 \\ \mathcal{N}_0 ::= \mathcal{A} & \mathcal{N}_{n+1} ::= \mathcal{P}_n \mid \mathcal{N}_{n+1} \,\&\, \mathcal{N}_{n+1} \mid \mathcal{N}_{n+1} \parr \mathcal{N}_{n+1} \mid \top \mid \bot \end{array} \tag{3.1}$$

Note that according to the grammar (3.1) we have $\mathcal{P}_0 = \mathcal{N}_0 = \mathcal{A}$. Furthermore, $\mathcal{N}_n \subseteq \mathcal{P}_{n+1}$ and $\mathcal{P}_n \subseteq \mathcal{N}_{n+1}$ for every $n \geq 0$. Proposition 3.1.1 formalizes the relation between the classes of different levels of the substructural hierarchy.

Axiom in intuitionistic and classical version		Name	Rule	Class
i:	$A \multimap 1, \quad \bot \multimap A$	weakening	w	$\mathcal{N}_2^i$
c:	$A \parr 1$			$\mathcal{N}_2$
i:	$A \multimap A \otimes A$	contraction	c	$\mathcal{N}_2^i$
c:	$A^\bot \parr A \otimes A$			$\mathcal{N}_2$
i:	$A \oplus (A \multimap \bot)$	excluded middle	em	$\mathcal{P}_2^i$
c:	$A \oplus A^\bot$			$\mathcal{P}_1$
i:	$(A \multimap B)_{\&1} \oplus (B \multimap A)_{\&1}$	linearity	com	$\mathcal{P}_3^{i'}$
c:	$(A^\bot \parr B)_{\&1} \oplus (B^\bot \parr A)_{\&1}$			$\mathcal{P}_3'$
i:	$((A^{\otimes 2} \multimap B) \& ((B \multimap \bot)^{\otimes 2} \multimap (A \multimap \bot))) \multimap (A \multimap B)$	Nelson	nel	$\mathcal{N}_3^i$
c:	$((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)) \parr A^\bot \parr B^\bot$			$\mathcal{N}_2$
i:	$A \oplus (A^{\otimes n} \multimap \bot)$	n-excluded middle	n-em	$\mathcal{P}_2^i$
c:	$A \oplus (A^\bot)^{\parr n}$			$\mathcal{P}_2$

$A^{\otimes n}$ is an abbreviation for $A \otimes \ldots \otimes A$, and $A^{\parr n}$ is an abbreviation for $A \parr \ldots \parr A$

Table 3.1: Axioms and their corresponding class in substructural hierarchy [CST09]

Proposition 3.1.1. *[CST09] Every axiom belongs to some $\mathcal{P}_n$ and some $\mathcal{N}_n$, and for all n, we have $\mathcal{P}_n \subseteq \mathcal{P}_{n+1}$ and $\mathcal{N}_n \subseteq \mathcal{N}_{n+1}$. Furthermore, $A \in \mathcal{P}_n$ iff $A^\bot \in \mathcal{N}_n$.*

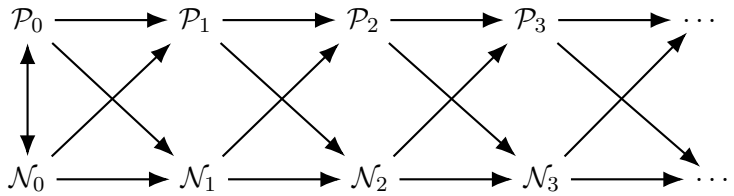

Figure 3.1: The Substructural Hierarchy [CST09]

Figure 3.1 depicts the hierarchy and the relation between classes where the arrow $\rightarrow$ stands for the set inclusion $\subseteq$.

A similar substructural hierarchy was first introduced in [CGT08] for formulas in the language of Intuitionistic Multiplicative Additive Linear Logic (IMALL) where the multiplicative *or* $A^\bot \parr B$ is expressed through the linear implication $A \multimap B$ [Gir87]. The substructural hierarchy $(\mathcal{P}_n^i, \mathcal{N}_n^i)$ for IMALL is defined as:

$$\begin{aligned} \mathcal{P}_0^i &::= \mathcal{A} \qquad & \mathcal{P}_{n+1}^i &::= \mathcal{N}_n^i \mid \mathcal{P}_{n+1}^i \otimes \mathcal{P}_{n+1}^i \mid \mathcal{P}_{n+1}^i \oplus \mathcal{P}_{n+1}^i \mid 1 \mid 0 \\ \mathcal{N}_0^i &::= \mathcal{A} \qquad & \mathcal{N}_{n+1}^i &::= \mathcal{N}_n^i \mid \mathcal{N}_{n+1}^i \,\&\, \mathcal{N}_{n+1}^i \mid \mathcal{P}_{n+1}^i \multimap \mathcal{N}_{n+1}^i \mid \top \mid \bot \end{aligned} \tag{3.2}$$

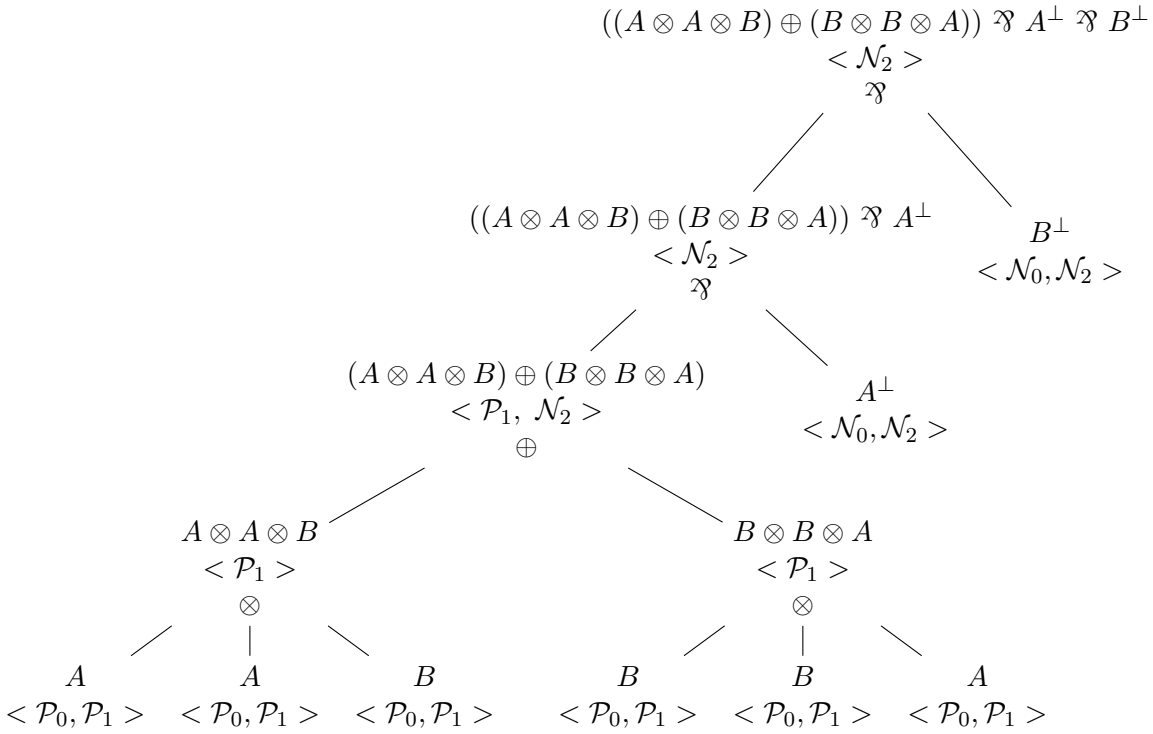

Figure 3.2: Identifying the class of *Nelson axiom*

where $\mathcal{A}$ is the set of positive (i.e. without negation) atomic axioms. Table 3.1 shows some examples of axioms and their classes according to the substructural hierarchy defined for intuitionistic linear logic (3.2) and classical linear logic (3.1). Note that some *intuitionistic* axioms belong to higher level in the hierarchy then their equivalent classical linear logic axioms. For example, the *excluded middle* and the *Nelson axiom* belong respectively to classes $\mathcal{P}_2^i$ and $\mathcal{N}_3^i$ according to grammar (3.2) but are brought to levels $\mathcal{P}_1$ and $\mathcal{N}_2$ according to grammar (3.1).

To identify the class where a formula ϕ belongs to, one has to decompose the formula into subformulas according to the most binding logical connectives until reaching atomic formulas. If we know the classes of all subformulas of ϕ, the class of ϕ can be determined by utilizing the grammar (3.1). For example, Figure 3.2 shows a tree representing the decomposition of the *Nelson Axiom* into subformulas. Each node of the tree is labeled with: the subformula corresponding to the node, the class in the substructural hierarchy where the subformula belongs to (enclosed in angle brackets <>) and the most binding logical connective in the subformula. The root node is the *Nelson axiom* and the leaves are atomic formulas. Since the classes of atomic formulas $\mathcal{A}$ are already defined in grammar (3.1), we can determine the classes in the rest of the nodes by traversing the tree bottom-up and applying the rules defined by grammar (3.1).

Proposition 3.1.2 formalizes another important observation of the structure of axioms

$$w\ \frac{\vdash \Gamma}{\vdash \Gamma, \Delta} \qquad c\ \frac{\vdash \Gamma, \Delta, \Delta}{\vdash \Gamma, \Delta} \qquad em\ \frac{\vdash \Gamma, \Gamma}{\vdash \Gamma} \qquad com\ \frac{\mathcal{H}| \vdash \Gamma, \Theta \qquad \mathcal{H}| \vdash \Sigma, \Delta}{\mathcal{H}\ | \vdash \Gamma, \Delta\ | \vdash \Sigma, \Theta}$$

$$nel\ \frac{\vdash \Gamma, \Sigma, \Sigma, \Delta \qquad \vdash \Gamma, \Sigma, \Delta, \Delta}{\vdash \Gamma, \Delta, \Sigma} \qquad n\text{-}em\ \frac{\mathcal{H}| \vdash \Gamma, \Sigma_1 \qquad \ldots \qquad \mathcal{H}| \vdash \Gamma, \Sigma_n}{\mathcal{H}| \vdash \Gamma| \vdash \Sigma_1, \ldots, \Sigma_n}$$

Table 3.2: Rules generated the from axioms in Table 3.1 [CST09]

belonging to certain classes of the substructural hierarchy. This observation will be used later on for the syntactic transformations from axioms to analytic rules.

Proposition 3.1.2. *[CST09] Every axiom $\phi \in \mathcal{P}_{n+1}$ is equivalent to an axiom of the form $\oplus_i(\otimes_j \psi_{i,j})$ where $\psi_{i,j} \in \mathcal{N}_n$ for each i, j. And every axiom $\phi \in \mathcal{N}_{n+1}$ is equivalent to an axiom of the form $\&_i(\parr_j \psi_{i,j})$ where $\psi_i, j \in \mathcal{P}_n$ for each i, j.*

To facilitate the transformation procedure, the axioms belonging to class $\mathcal{N}_2$, are previously translated into a normal form as defined in Definition 3.1.3:

Definition 3.1.3. *[CST09] An axiom ϕ is called $\mathcal{N}_2 - normal$ if it is of the shape $\phi = \parr_k(\oplus_i(\otimes_j A_{k,i,j}))$ where each $A_{i,j,k}$ is atomic.*

From Proposition 3.1.2 it follows that every axiom in $\mathcal{N}_2$ can be transformed into a finite additive conjunction (&) of $\mathcal{N}_2 - normal$ axioms. Note that every $\mathcal{N}_2$ axiom in Table 3.1 is already in $\mathcal{N}_2 - normal$ form.

In absence of weakening, only a subset $\mathcal{P}_3' \subset \mathcal{P}_3$, defined by the grammar rule (3.3), is covered by the transformation procedure which is described in the following sections. An example of an axiom in $\mathcal{P}_3'$ class is the *linearity axiom* in Table 3.1. Figure 3.3 shows a tree representing the decomposition of the *linearity axiom* into subformulas and the class in the substructural hierarchy where each subformula belongs to. The classes of inner nodes in the tree are determined similarly as in the example in Figure 3.2 but the grammar rule (3.3) is also used for determining the class $\mathcal{P}_3'$.

$$\mathcal{P}_3' ::= \mathcal{N}_2\ \&\ 1 \mid \mathcal{P}_3' \otimes \mathcal{P}_3' \mid \mathcal{P}_3' \oplus \mathcal{P}_3' \mid 1 \mid 0 \tag{3.3}$$

Proposition 3.1.4 is used to transform an axiom $\phi \in \mathcal{P}_3'$ into an equivalent (w.r.t. Definition 2.3.4) finite set of axioms $\{\psi_1, \ldots, \psi_n\}$. This transformation will be used as a preprocessing step in Section 3.3.

Proposition 3.1.4. *[CST09] Every axiom $\phi \in \mathcal{P}_3'$ is equivalent to a finite set $\{\psi_1, \ldots, \psi_n\}$ of axioms such that $\psi_i = \oplus_{j=1}^{m_i}(\xi_{i,j})_{\&1}$ where $\xi_{i,j}$ is $\mathcal{N}_2 - normal$ for all i, j.*

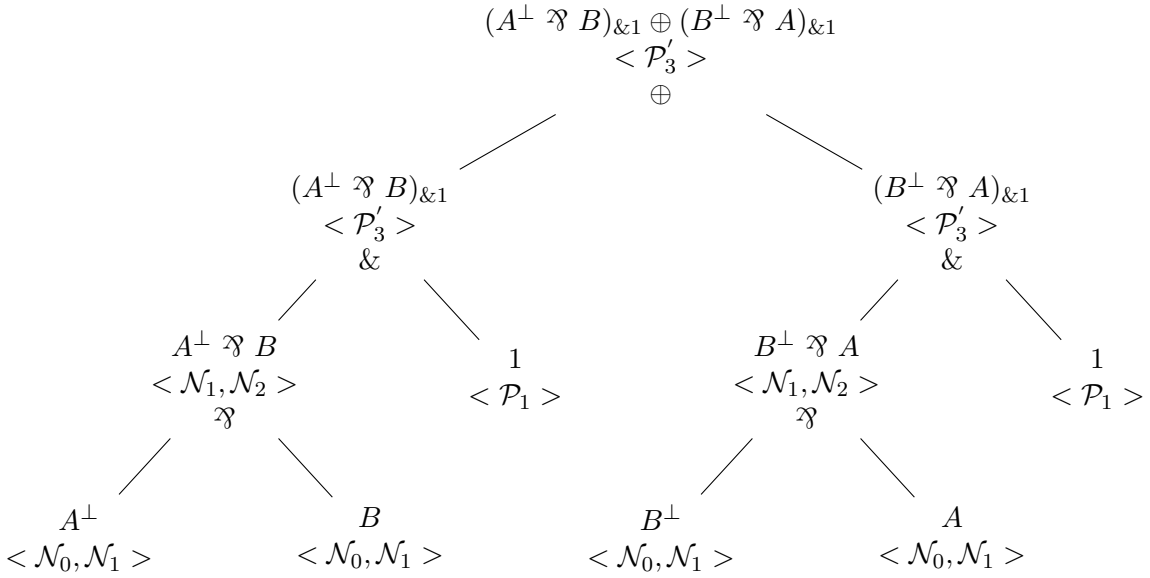

Figure 3.3: Identifying the class of the linearity axiom. Note that $A_{\&1}$ is an abbreviation for $A\&1$, see Section 2.1.

3.2 From $\mathcal{N}_2$-axioms to Analytic Sequent Rules

This section provides a description of the algorithm introduced in [CST09] for transforming the axioms within the class $\mathcal{N}_2$ into equivalent sequent calculus structural rules. The description below is a reformulation, with more details, of the results in [CST09]. The described algorithm extends the one introduced in [CGT08] for axioms of class $\mathcal{N}_2^i$ as defined by grammar (3.2). The transformation given by Lemma 3.2.1 (also known as the *Ackerman Lemma*) is the most important step of the algorithm. The complete algorithm is given in the proof of Theorem 3.2.2.

Lemma 3.2.1 (Ackerman Lemma)**.** *[CST09] For any axiom ξ, the following two sequent rules are equivalent*

$$\frac{\vdash \Sigma_1 \quad \ldots \quad \vdash \Sigma_n}{\vdash \Gamma, \xi}\ (1) \qquad \text{and} \qquad \frac{\vdash \Sigma_1 \quad \ldots \quad \vdash \Sigma_n \quad \vdash \Delta, \xi^\perp}{\vdash \Gamma, \Delta}\ (2) \tag{3.4}$$

where Δ is a fresh metavariable standing for multisets of formulas.

Proof. The direction (2) $\Rightarrow$ (1) is shown by taking the premises of (1) and making use of (2) to get the conclusion of (1). Since (2) has one more premise (namely $\vdash \Delta, \xi^\perp$) than (1), we have to instantiate the multiset variable Δ in this premise in such a way that we can derive the conclusion of (1). By letting $\Delta = \xi$ and using the ax rule we have the following:

$$\frac{\vdash \Sigma_1 \quad \ldots \quad \vdash \Sigma_n \qquad ax\ \dfrac{}{\vdash \xi, \xi^\perp}}{\vdash \Gamma, \xi}\ (2)$$

The other direction (1) ⇒ (2) is shown by starting with the premises of (2) and using (1) and the *cut* rule to obtain the conclusion of (2):

$$\dfrac{\dfrac{\vdash \Sigma_1 \quad \dots \quad \vdash \Sigma_n}{\vdash \Gamma, \xi}\,(1) \qquad \vdash \Delta, \xi^{\perp}}{\vdash \Gamma, \Delta}\,cut$$

□

Corollary 3.2.1.1. *If $\xi \in \mathcal{P}_1$, then the following rules*

$$\dfrac{\vdash \Sigma_1 \quad \dots \quad \vdash \Sigma_n}{\vdash \Gamma, \xi}\,(1) \qquad \text{and} \qquad \dfrac{\vdash \Sigma_1 \quad \dots \quad \vdash \Sigma_n \quad \vdash \Delta, \xi^{\perp}}{\vdash \Gamma, \Delta}\,(2) \tag{3.5}$$

are equivalent to the rule:

$$\dfrac{\vdash \Sigma_1 \quad \dots \quad \vdash \Sigma_n \quad \vdash \Delta, A_{1,1}, \dots, A_{1,k_1} \quad \dots \quad \vdash \Delta, A_{m,1}, \dots, A_{m,k_m}}{\vdash \Gamma, \Delta}\,(3) \tag{3.6}$$

where each $A_{i,j}$ is atomic and $m, k_1, \dots, k_m \geq 0$.

Proof. The equivalence (1) ⇔ (3) is a special case of the equivalence (1) ⇔ (2) of the Ackerman Lemma 3.2.1. From Proposition 3.1.1, if $\xi \in \mathcal{P}_1$ then $\xi^{\perp} \in \mathcal{N}_1$, and therefore, $\xi^{\perp}$ is equivalent with an axiom of the form $\&_i(\parr_j A_{i,j})$. Then (3) is reduced to (2) by using the invertible rules $\parr, \&, \top, \perp$. □

Note that during the transformation of a sequent rule from (1) to (2) in Lemma 3.2.1, the formula variable ξ in the conclusion is replaced with the fresh multiset variable Δ and a new premise is introduced, consisting of both the multiset variable Δ and the negated formula variable $\xi^{\perp}$. In this way, the conclusion has one less formula variable, which is moved to the premises.

Theorem 3.2.2. *[CST09] Every $\mathcal{N}_2 - axiom$ can be transformed into a finite set of equivalent structural sequent rules, whose conclusion consist only of multiset variables.*

Proof. Let $\phi \in \mathcal{N}_2$. By Proposition 3.1.2, ϕ is equivalent to a set of axioms $\{\psi_1, \dots, \psi_n\}$ where each $\psi_i \in \mathcal{N}_2 - normal$. And therefore, each ψ_i is equivalent to a rule without premises:

$$\dfrac{}{\vdash \xi_{i,1}, \dots, \xi_{i,m_i}} \tag{3.7}$$

where each $\psi_{i,j} \in \mathcal{P}_1$. The claim follows by repeatedly applying the Asckerman Lemma 3.2.1 until every formula variable in the conclusion is replaced with a new multiset variable. In this way, for each ψ_i we obtain a sequent rule whose conclusion consist only of multiset variables. □

Example 3.2.3. *This example shows the steps of applying the procedure given in the proof of Theorem 3.2.2 to the contraction axiom $A^{\perp} \parr (A \otimes A)$ (see Table 3.1). The axiom is already in $\mathcal{N}_2-$normal form, hence, the algorithm will generate only one sequent rule.*

$$
\begin{array}{ll}
 & \dfrac{}{\vdash A^{\perp} \parr (A \otimes A)} \\[2ex]
\overset{\text{inverted } \parr}{\Longrightarrow} & \dfrac{}{\vdash A^{\perp}, A \otimes A} \\[2ex]
\overset{\text{Ackerman Lemma}}{\Longrightarrow} & \dfrac{\vdash \Gamma,\ (A^{\perp})^{\perp}}{\vdash \Gamma, A \otimes A} \\[2ex]
\overset{\text{Ackerman Lemma}}{\Longrightarrow} & \dfrac{\vdash \Gamma,\ (A^{\perp})^{\perp} \qquad \vdash \Delta, (A \otimes A)^{\perp}}{\vdash \Gamma, \Delta} \\[2ex]
\overset{\text{de Morgan}}{\Longrightarrow} & \dfrac{\vdash \Gamma,\ A \qquad \vdash \Delta, A^{\perp} \parr A^{\perp}}{\vdash \Gamma, \Delta} \\[2ex]
\overset{\text{inverted } \parr}{\Longrightarrow} & c' \dfrac{\vdash \Gamma,\ A \qquad \vdash \Delta,\ A^{\perp},\ A^{\perp}}{\vdash \Gamma, \Delta}
\end{array}
\tag{3.8}
$$

Note that the conclusion of the obtained rule c' consists only of multiset variables. Furthermore, by invoking the the invertible rules and the de Morgan laws, all of the logical connectives are eliminated from the premises. Section 3.4 provides the final step of the algorithm that converts c' into a rule that obeys the subformula property.

3.3 From $\mathcal{P}_3'$-axioms to Analytic Hypersequent Rules

We describe the algorithm introduced in [CST09] for transforming axioms within the class $\mathcal{P}_3'$ into equivalent structural rules in hypersequent calculus. In presence of weakening, this algorithm covers the whole class $\mathcal{P}_3$. Note that the Ackerman Lemma 3.2.1 can be extended to the hypersequent calculus while keeping the same proof:

Lemma 3.3.1. *[CST09] For any axiom ξ, the following hypersequent rules are equivalent*

$$
\frac{\mathcal{G}_1 \quad \dots \quad \mathcal{G}_n}{\mathcal{H} \mid \mathcal{H}' \mid \vdash \Gamma, \xi} \quad \text{and} \quad \frac{\mathcal{G}_1 \quad \dots \quad \mathcal{G}_n \quad \mathcal{H} \mid \vdash \Delta, \xi^{\perp}}{\mathcal{H} \mid \mathcal{H}' \mid \vdash \Gamma, \Delta} \tag{3.9}
$$

where Δ is a fresh metavariable standing for multisets of formulas.

Corollary 3.3.1.1. *If $\xi \in \mathcal{P}_1$, then the rules*

$$
\frac{\mathcal{G}_1 \quad \dots \quad \mathcal{G}_n}{\mathcal{H} \mid \mathcal{H}' \mid \vdash \Gamma, \xi} \quad \text{and} \quad \frac{\mathcal{G}_1 \quad \dots \quad \mathcal{G}_n \quad \mathcal{H} \mid \vdash \Delta, \xi^{\perp}}{\mathcal{H} \mid \mathcal{H}' \mid \vdash \Gamma, \Delta} \tag{3.10}
$$

are equivalent to the rule:

$$\frac{\mathcal{G}_1 \quad \dots \quad \mathcal{G}_n \quad \mathcal{H}|\vdash \Delta, A_{1,1}, \dots, A_{1,k_1} \quad \dots \quad \mathcal{H}|\vdash \Delta, A_{m,1}, \dots, A_{m,k_m}}{\mathcal{H} \mid \mathcal{H}' \mid \vdash \Gamma, \Delta} \tag{3.11}$$

where each $A_{i,j}$ is atomic and $m, k_1, \dots, k_m \geq 0$.

A hyperstructural rule is formally defined as follows:

Definition 3.3.2. *[CST09] A hypersequent structural rule or hyperstructural rule is*

$$\frac{\mathcal{H} \mid \vdash \Psi_1 \quad \dots \quad \mathcal{H} \mid \vdash \Psi_n}{\mathcal{H} \mid \vdash \Phi_1 \mid \dots \mid \vdash \Phi_m} \tag{3.12}$$

where each Φ_i and Ψ_j contains only multiset variables and formula variables.

Similar to Section 3.2, the algorithm is given in the proof of the following theorem:

Theorem 3.3.3. *[CST09] Every $\mathcal{P}_3'$-axiom is equivalent to a finite set of hyperstructural rules where each $\Phi_1, \dots, \Phi_n$ consist of mutually distinct multiset variables.*

Proof. Let $\phi \in \mathcal{P}_3'$. By Proposition 3.1.4, ϕ is equivalent (w.r.t. Definition 2.3.4), to a finite set of formulas $\{\psi_1, \dots, \psi_n\}$ where each ψ_i has the form $\psi_i = \oplus_j(\xi_{i,j})_{\&1}$ with $\xi_{i,j} \in \mathcal{N}_2 - normal$ for all i, j. By Proposition 2.3.3, each ψ_i is equivalent to:

$$\vdash \xi_{i,1,1}, \dots, \xi_{i,1,m_{i1}} | \dots | \dots | \vdash \xi_{i,k,1}, \dots, \xi_{i,1,m_{ik}} \tag{3.13}$$

where $\xi_{i,j,l} \in \mathcal{P}_1$. From the *ew*-rule, (3.13) is equivalent to:

$$\frac{}{\mathcal{H} \mid \vdash \xi_{i,1,1}, \dots, \xi_{i,1,m_{i1}} | \dots | \dots | \vdash \xi_{i,k,1}, \dots, \xi_{i,1,m_{ik}}} \tag{3.14}$$

On every component of (3.14) we can apply the Lemma 3.3.1 similarly as in the proof of Theorem 3.2.2. □

Example 3.3.4. *Consider the linearity axiom which belongs to $\mathcal{P}_3'$. The transformation*

steps given in the proof of Theorem 3.3.3 are as follows:

$$\frac{}{\vdash (A^\perp \parr B)_{\&1} \oplus (B^\perp \parr A)_{\&1}}$$

$$\Longrightarrow \qquad \frac{}{\mathcal{H} \mid\vdash A^\perp \parr B \mid\vdash B^\perp \parr A}$$

$$\overset{inv\ \parr}{\Longrightarrow} \qquad \frac{}{\mathcal{H} \mid\vdash A^\perp,\ B \mid\vdash B^\perp,\ A}$$

$$\overset{Ack.\ Lemma}{\Longrightarrow} \qquad \frac{\mathcal{H} \mid\vdash \Gamma, (A^\perp)^\perp}{\mathcal{H} \mid\vdash \Gamma,\ B \mid\vdash B^\perp,\ A}$$

$$\overset{Ack.\ Lemma}{\Longrightarrow} \qquad \frac{\mathcal{H} \mid\vdash \Gamma, (A^\perp)^\perp \qquad \mathcal{H} \mid\vdash \Delta, B^\perp}{\mathcal{H} \mid\vdash \Gamma,\ \Delta \mid\vdash B^\perp,\ A}$$

$\ldots$

$$\overset{Ack.\ Lemma}{\Longrightarrow} \qquad \frac{\mathcal{H} \mid\vdash \Gamma, (A^\perp)^\perp \qquad \mathcal{H} \mid\vdash \Delta, B^\perp \qquad \mathcal{H} \mid\vdash \Sigma, (B^\perp)^\perp \qquad \mathcal{H} \mid\vdash \Theta, A^\perp}{\mathcal{H} \mid\vdash \Gamma,\ \Delta \mid\vdash \Sigma,\ \Theta}$$

$$\overset{de\ Morgan}{\Longrightarrow} \qquad com' \frac{\mathcal{H} \mid\vdash \Gamma, A \qquad \mathcal{H} \mid\vdash \Delta, B^\perp \qquad \mathcal{H} \mid\vdash \Sigma, B \qquad \mathcal{H} \mid\vdash \Theta, A^\perp}{\mathcal{H} \mid\vdash \Gamma,\ \Delta \mid\vdash \Sigma,\ \Theta}$$

Similarly as in Example 3.2.3, by invoking the invertible rules and the de Morgan laws, all of the logical connectives are eliminated from the premises.

If the base calculus under consideration, HMALL, is equipped with the *weakening* rule w (Table 3.2), then the transformation procedure can cover the whole class $\mathcal{P}_3$:

Corollary 3.3.4.1. *[CST09] Every axiom $\phi \in \mathcal{P}_3$ is equivalent to a set of hyperstructural rules in the presence of weakening.*

Proof. Similarly as in Proposition 3.1.2, an axiom $\phi \in \mathcal{P}_3'$ is equivalent to an axiom of the form $\oplus_i(\otimes_j \psi_{i,j} \& 1))$ where $\psi_{i,j} \in \mathcal{N}_2$. In the presence of weakening, we have $\vdash_{\mathrm{MALL}\,+\,w} A \multimap A \,\&\, 1$ because[1]:

$$\dfrac{\dfrac{ax\ \dfrac{}{\vdash A^\perp, A} \qquad w\ \dfrac{1\ \dfrac{}{\vdash 1}}{\vdash A^\perp, 1}}{\&\ \dfrac{\vdash A^\perp,\ A \,\&\, 1}{\parr\ \dfrac{\vdash A^\perp \parr (A \,\&\, 1)}{}}} \qquad \dfrac{ax\ \dfrac{}{\vdash A,\ A^\perp}}{\oplus_1\ \dfrac{\vdash A,\ A^\perp \oplus\ 0}{\parr\ \dfrac{\vdash A \parr (A^\perp \oplus\ 0)}{}}}}{\&\ \dfrac{}{\vdash (A^\perp \parr (A \,\&\, 1)) \,\&\, (A \parr (A^\perp \oplus\ 0))}} \tag{3.15}$$

[1] $A \multimap A \,\&\, 1$, is equivalent to $(A^\perp \parr (A \,\&\, 1)) \,\&\, (A \parr (A^\perp \oplus\ 0))$

Hence $\phi \in \mathcal{P}_3'$ is also equivalent to an axiom of the form $\oplus_i(\otimes_j \psi_{i,j})$. Which means that (from Proposition 3.1.2) $\phi \in \mathcal{P}_3$. Therefore, in presence of weakening, $\mathcal{P}_3'$ is reduced to $\mathcal{P}_3$. Hence, the whole class $\mathcal{P}_3$ is covered by the Theorem 3.3.3.

□

3.4 Rule Completion

This section describes the final step of the procedure for transforming, if possible, the sequent rules obtained in Sections 3.2 and 3.3 into *completed rules* as defined by Definition 3.4.3. The condition that the rules need to satisfy before applying the final transformation step, is the *acyclicity*:

Definition 3.4.1. *[CST09] The cut-closure $CUT(r)$ of a (hyper)structural rule (r) is the minimal set which contains the premises of (r) and it is closed under application of the cut rule. A rule (r) is said to be cyclic if for some formula variable A, we have $\mathcal{H}| \vdash \Gamma, A, A^{\perp} \in CUT(r)$. Otherwise, (r) is acyclic. .*

Example 3.4.2. *The cut closure of the (c') rule obtained from the contraction axiom in Example 3.2.3 is $CUT(c') = \{\vdash \Gamma, A;\ \vdash \Delta, A^{\perp}, A^{\perp};\ \vdash \Gamma, \Delta, A^{\perp}\}$. And the cut closure of the (com') rule obtained in Example 3.3 is*

$$CUT(com') = \{\mathcal{H}| \vdash \Gamma, A;\ \mathcal{H}| \vdash \Delta, B^{\perp};\ \mathcal{H}| \vdash \Sigma, B;\ \mathcal{H}| \vdash \Theta, A^{\perp}; \mathcal{H}| \vdash \Gamma, \Theta; \mathcal{H}| \vdash \Delta, \Sigma\}$$

Since, no sequent of the form $\mathcal{H}| \vdash A, A^{\perp} \notin CUT(c')$ or $\mathcal{H}| \vdash A, A^{\perp} \notin CUT(com')$, then both (c') and (com') are acyclic. Examples of cyclic rules are:

$$cancel\ \frac{\vdash \Gamma, A, A^{\perp}}{\vdash \Gamma} \qquad and \qquad menace\ \frac{\vdash \Gamma, A^{\perp}, A^{\perp} \qquad \vdash \Delta, A, A}{\vdash \Gamma, \Delta}$$

The following definition formalizes the notion of a *completed rule*, and its properties:

Definition 3.4.3. *[CST09] A hyperstructural rule (r) is called completed if it satisfies the following properties:*

- *No Forumula Variable (NFV): The conclusion and all premises of (r) contain only multiset variables and hypersequent contexts.*
- *Linear Conclusion (LC): Each multiset variable occurs at most once in the conclusion of (r).*
- *Subformula Property (SP): Each multiset variable occurring in the premises of (r) also occurs in the conclusion.*

The rules resulting from the procedures in Sections 3.2 and 3.3 already satisfy the (LC) property, the (NFV) property only in their conclusion, and the (SP) property only for their multiset variables. To convert them into completed rules, it suffices to eliminate the formula variables from the premises. The proof of the following theorem gives also a procedure for generating completed rules out of (hyper)sequent rules form Sections 3.2 and 3.3 by eliminating the formula variables from the premises while still preserving the equivalence between them.

Theorem 3.4.4. *[CST09] Any acyclic (hyper)structural rule* (r) *generated by the procedures in Theorems 3.2.2 and 3.3.3 can be transformed into a completed rule.*

Proof. Let (r) be an *acyclic* rule generated either by the procedure in Theorem 3.2.2 or 3.3.3. The proof is done by induction in the number of formula variables of r. In the base case, if there is no formula variable in premises, then (r) is already a complete rule. Otherwise, let A be such a formula variable, and let $\mathcal{G}_A^+$ and $\mathcal{G}_A^-$ be subsets of premises of r containing at least one occurrence of respectively A and $A^\perp$. If $\mathcal{G}_A^+ = \emptyset$, then $\mathcal{G}_A^-$ can be removed. The resulting rule implies the original one by instantiating occurrences of A with $\top$. Similarly, if $\mathcal{G}_A^- = \emptyset$, then $\mathcal{G}_A^+$ can be removed. Otherwise, if $\mathcal{G}_A^+ \neq \emptyset$ and $\mathcal{G}_A^- \neq \emptyset$, from the acyclicity property of (r), we have the following two observations:

- $A \notin \mathcal{G}_A^-$ and $A^\perp \notin \mathcal{G}_A^+$.
- if some hypersequent $\mathcal{H}_A \in \mathcal{G}_A^+$ ($\mathcal{H}_{A^\perp} \in \mathcal{G}_A^-$) contains more than one occurrence of A ($A^\perp$), then no hypersequent $\mathcal{H}_{A^\perp} \in \mathcal{G}_A^-$ ($\mathcal{H}_A \in \mathcal{G}_A^+$) contains more than one occurrence of $A^\perp$ (A).

W.l.o.g. we can assume that the elements of $\mathcal{G}_A^+$ have exactly one occurrence of A and the elements of $\mathcal{G}_A^-$ have one or more occurrences of $A^\perp$. Hence, $\mathcal{G}_A^+$ and $\mathcal{G}_A^-$ can be written as: $\mathcal{G}_A^+ = \{\mathcal{H}| \vdash \Upsilon_i, A : 1 \leq i \leq m\}$ where m is the number of premises having one occurrence of A; and $\mathcal{G}_A^- = \{\mathcal{H}| \vdash \Phi_j, A_{j_1}^\perp, \ldots, A_{j_k}^\perp : 1 \leq j \leq n \text{ and } j_k \geq 1\}$, where n is the number of premises having j_k occurrences of $A^\perp$. The set $\mathcal{G}_A^{cut} = \{\mathcal{H}| \vdash \Phi_j, \Upsilon_{i_{j1}}, \ldots, \Upsilon_{i_{jk}} : 1 \leq j \leq n \text{ and } 1 \leq i_{j1}, \ldots, i_{jk} \leq m\}$ is is the set of all hypersequents obtained by applying the *cut* rule between the elements of $\mathcal{G}_A^+$ and $\mathcal{G}_A^-$ in every possible combination, until all occurrences of the formula variable A are eliminated. Let r' be the rule obtained from r by replacing premises $\mathcal{G}_A^+ \cup \mathcal{G}_A^-$ with $\mathcal{G}_A^{cut}$. The new rule r' is equivalent to r. Indeed, $r' \Rightarrow r$ can be shown by using *cut*. The other direction, $r \Rightarrow r'$, requires the instantiation of A in the premises of r, in such a way, that each premise of r is either derivable in HMALL or is a premise of r' as well. By letting $\tilde{\mathrm{A}} = \oplus_{i=1}^m \Upsilon_i^\perp$, we have $\vdash_{\mathrm{HMALL}} \mathcal{H}| \vdash \Upsilon_i, \tilde{\mathrm{A}}$ for all $i = 1, \ldots, m$. Furthermore, for each $j = 1, \ldots, n$ the hypersequent $\mathcal{H}| \vdash \Phi_j, \tilde{\mathrm{A}}_{j_1}^\perp, \ldots, \tilde{\mathrm{A}}_{j_k}^\perp$ is derivable from $\mathcal{G}_A^{cut}$ by using the & rule. Hence, after instantiating the occurrences of A in the premises with $\tilde{\mathrm{A}}$, we can apply the rule r and obtain the conclusion of r'. This completes the proof of the equivalence $r \Leftrightarrow r'$. The acyclicity of r is preserved, and the number of formula variables occurring in its premises is reduced by 1.

□

Example 3.4.5. *Since the rule* (c') *from Example 3.2.3 is acyclic (see Example 3.4.2), we can apply the procedure in Theorem 3.4.4 to convert it into a completed rule.*

$$c' \frac{\vdash \Gamma,\ A \qquad \vdash \Delta,\ A^{\perp},\ A^{\perp}}{\vdash \Gamma, \Delta} \tag{3.16}$$

The only formula variable occurring in the premises of (c') is A. $\mathcal{G}^{+}_{A}$, $\mathcal{G}^{-}_{A}$ and $\mathcal{G}^{cut}_{A}$ are as following:

$$\begin{gathered} \mathcal{G}^{+}_{A} = \{\mathcal{H} | \vdash \Gamma,\ A\}, \qquad \mathcal{G}^{-}_{A} = \{\mathcal{H} | \vdash \Delta,\ A^{\perp},\ A^{\perp}\} \\ \mathcal{G}^{cut}_{A} = \{\mathcal{H} | \vdash \Gamma,\ \Delta,\ \Delta\} \end{gathered} \tag{3.17}$$

By replacing $\mathcal{G}^{-}_{A} \cup \mathcal{G}^{+}_{A}$ with $\mathcal{G}^{cut}_{A}$ we obtain the *contraction* rule (c):

$$c \frac{\vdash \Gamma, \Gamma, \Delta}{\vdash \Gamma, \Delta} \tag{3.18}$$

One can show that $(c) \Rightarrow (c')$ by taking the premises of (c') and applying the *cut* and (c) as in the following:

$$cut \frac{\vdash \Gamma, A \qquad cut \dfrac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}, A^{\perp}}{\vdash \Gamma, \Delta, A^{\perp}}}{c \dfrac{\vdash \Gamma, \Gamma, \Delta}{\vdash \Gamma, \Delta}} \tag{3.19}$$

The other direction, $(c') \Rightarrow (c)$ can be shown by instantiating A in (c') with $\Gamma^{\perp}$ and using the ax as in the following:

$$c' \frac{\vdash \Delta, \Gamma, \Gamma \qquad ax \dfrac{}{\vdash \Gamma, \Gamma^{\perp}}}{\vdash \Delta, \Gamma} \tag{3.20}$$

In this way, the acquired rule (c), is *completed* and is equivalent to the sequent rule (c').

Example 3.4.6. *Similarly, we can convert the acyclic rule* (com') *in Example 3.3.4, into a completed rule:*

$$com' \frac{\mathcal{H} \mid \vdash \Gamma, A \qquad \mathcal{H} \mid \vdash \Delta, B^{\perp} \qquad \mathcal{H} \mid \vdash \Sigma, B \qquad \mathcal{H} \mid \vdash \Theta, A^{\perp}}{\mathcal{H} \mid \vdash \Gamma,\ \Delta \mid \vdash \Sigma,\ \Theta} \tag{3.21}$$

In this case, the rule has two different formula variables in the premises (A and B), and therefore, the transformation will go through two steps. By picking first the formula variable A we have $\mathcal{G}^{+}_{A}$, $\mathcal{G}^{-}_{A}$ and $\mathcal{G}^{cut}_{A}$ as follows:

$$\mathcal{G}_A^+ = \{\mathcal{H} \mid\vdash \Gamma, A\}, \qquad \mathcal{G}_A^- = \{\mathcal{H} \mid\vdash \Theta, A^\perp\}$$
$$\mathcal{G}_A^{cut} = \{\mathcal{H} \mid\vdash \Gamma, \Theta\} \tag{3.22}$$

By replacing $\mathcal{G}_A^+ \cup \mathcal{G}_A^-$ with $\mathcal{G}_A^{cut}$ we obtain the following rule (com''), which has no occurrences of A in the premises:

$$com'' \frac{\mathcal{H} \mid\vdash \Gamma, \Theta \qquad \mathcal{H} \mid\vdash \Delta, B^\perp \qquad \mathcal{H} \mid\vdash \Sigma, B}{\mathcal{H} \mid\vdash \Gamma,\ \Delta \mid\vdash \Sigma,\ \Theta} \tag{3.23}$$

Using the *cut* rule, we can show that $(com'') \Rightarrow (com')$. The other direction $(com') \Rightarrow (com'')$ is shown by instantiating A with Θ and using the ax rule:

$$com' \frac{\mathcal{H} \mid\vdash \Gamma, \Theta \qquad \mathcal{H} \mid\vdash \Delta, B^\perp \qquad \mathcal{H} \mid\vdash \Sigma, B \qquad ax\ \dfrac{}{\mathcal{H} \mid\vdash \Theta, \Theta^\perp}}{\mathcal{H} \mid\vdash \Gamma,\ \Delta \mid\vdash \Sigma,\ \Theta} \tag{3.24}$$

In the same way, we can eliminate the remaining formula variable B:

$$\mathcal{G}_B^- = \{\mathcal{H} \mid\vdash \Delta, B^\perp\}, \qquad \mathcal{G}_B^+ = \{\mathcal{H} \mid\vdash \Sigma, B\}$$
$$\mathcal{G}_B^{cut} = \{\mathcal{H} \mid\vdash \Delta, \Sigma\} \tag{3.25}$$

And finally, by replacing $\mathcal{G}_B^+ \cup \mathcal{G}_B^-$ with $\mathcal{G}_B^{cut}$ we obtain the completed rule com:

$$com \frac{\mathcal{H} \mid\vdash \Gamma, \Theta \qquad \mathcal{H} \mid\vdash \Delta, \Sigma}{\mathcal{H} \mid\vdash \Gamma,\ \Delta \mid\vdash \Sigma,\ \Theta} \tag{3.26}$$

The equivalence $(com) \Leftrightarrow (com'')$ can be shown similarly as the equivalence $(com'') \Leftrightarrow (com')$. And therefore, we also have the equivalence $(com) \Leftrightarrow (com')$.

CHAPTER 4

Implementation

In this Chapter we provide a brief description of the architectural components for the existing tools in TINC and we describe the implementation details of InvAxiomCalc. This Chapter is organized as follows: Section 4.1 contains a gentle introduction to Prolog, focusing on the notions that are most relevant for our implementation. In Section 4.2 we describe the architectural components of the existing tools in TINC. Section 4.3 defines the data structure that we use during the implementation. Section 4.4 provides the implementation details for each component of InvAxiomCalc. Section 4.5 shows how InvAxiomCalc displays the computed results to the user. We close the Chapter with Section 4.6 which shows how to unit test a Prolog program, and in particular writing Unit tests for InvAxiomCalc.

4.1 A gentle introduction to Prolog

Prolog is a typical example of logic programing paradigm. In this paradigm, a program consists of sets of sentences in logical form expressing facts and rules about the problem domain. Prolog uses *terms* as building blogs for the rules and facts. There are four type of terms:

1. *atoms* - can be either:
   - strings starting with a lower case letter and followed by a sequence of alphanumeric or underscore characters, like `sings`, `plays`, `guitar`, `father`.
   - sequences of characters enclosed in single quotes, like `'John'`, `'Kurt'`, `'John sings'`, `'double base'`.
   - strings of special characters. For example `@=` , `:-`, `;`. Some of these have predefined meanings.

2. *numbers* - can be either integers (i.e. ..., -2, -1, 0, 1, 2, ...) or real numbers.

3. *variables* - are strings starting with an uppercase letter or underscore, and followed by any sequence of alphanumeric characters. For example: `X`, `Y`, `Z`, `Var`, `Var1`, `_var`, `_head_of_a_list`.

4. *compound terms* consist of an *atom* called '*functor*', and one or more terms called '*arguments*'. The arguments are enclosed in parenthesis and separated by commas. The functor has to be an atom, hence it cannot be a number or a variable. On the other hand, arguments can be any type of terms, thus, providing the possibility of having compound terms as arguments of another compound term. Some compound terms could be: `plays('Kurt', guitar)`, `sings('John')`, `knows(X, father(father(Y)))`.

   Additionally, there are two types of special compound terms: *lists* which are sets of comma separated terms enclosed in square brackets like `['Kurt', 'John', guitar]`; and *strings* which are sequences of characters enclosed in double quotes, like `"Music is great"`.

Atoms and numbers are also called *atomic terms*, while variables and compound terms are called *predicates*.

As mentioned earlier, a Prolog program consists of a finite set or rules and facts. A rule is a statement of this form:

```
head :- body.
```

where `head` is a *single* predicate and `body` is a *set* of predicates. Predicates are also referred as *goals*. The goals in the body can be separated by a comma '`,`' which stands for conjunction, or by a semicolon '`;`' which stands for disjunction of the surrounding goals. The intuitive meaning of the whole rule is: 'if `body` is true, then `head` is also true'.

A rule with an empty body is a *fact*. Facts are used to state goals that are always true. For example, we can state that `'Kurt'` plays guitar with the following fact:

```
plays('Kurt', guitar).
```

A program is also referred to as *knowledge base*. One can use *queries* for 'asking' the knowledge base for satisfiability of a certain predicate. Queries are statements of the form:

```
?- body.
```

where `body` is a sequence of goals separated by commas or semicolons, just like the body of the rules. Prolog will try to satisfy the body of the query, based on the facts and rules that are given in the knowledge base. Prolog is based on the *the closed world assumption* i.e. only the statements that can be deduced from the facts are considered to be true.

Another important feature of Prolog are the Definite Clause Grammars (DCGs). They are designed to support writing parsers that build a parse tree from a given list of tokens and for generating a flat list from a term. DCGs are generalization of context free grammars obtained by adding arguments to nonterminal symbols. The ability to call Prolog predicates increases their utility and expressive power. The syntax of DCG rules is similar to the syntax of ordinary rules described above, but they use the symbol `-->` rather then `:-` to separate the head from the body.

4.2 The System TINC

Tools for the Investigation of Nonclassical Logics (TINC) is a system developed in Prolog that implements the theoretical results in [CGT08, CLSZ13, CMS13] and is available at `http://www.logic.at/tinc`. Except for the web interface, a standard command-line interface is also provided for the users who want to run the tools on their local machine. TINC takes as input a logic specified via Hilbert systems or Kripke models and it generates an analytic calculus. Additionally, it also states certain properties about the logic, for example, it checks the sufficient condition for standard completeness. Currently, TINC consist of the following tools, which can handle large classes of substructural, paraconsistent and intermediate logics:

- *AxiomCalc* - transforms suitable axiomatic extensions of Full Lambek Calculus with exchange and weakening (Flew) into a cut-free hypersequent calculi. Furthermore, it provides an optional feature for exploiting the generated calculus by checking a sufficient condition for standard completeness of the given logic. It provides the theoretical results in [CGT08].
- *Paralyzer* - (Paraconsistent logic analyzer) transforms Hilbert axioms defining paraconsistent logics into sequent calculus rules. Furthermore, it extracts non-deterministic finite valued semantics from the obtained calculi which show the decidability of the logics and reveal whether the calculi are analytic. It provides the theoretical results in [CLSZ13].
- *Framinator* - (FRAMe condItioNs Automatically TO Rules) transforms first order frame conditions of the form $\forall\bar{x}\exists\bar{y}P$ for quantifier free P (i.e. the class Π_2 of the arithmetical hierarchy) into cut-free labeled sequent calculi. Provides the theoretical results in [CMS13].

The general structure of the implementation of each tool is depicted in Figure 4.1. The expected input is a formula of a specific form depending on the class of logics that

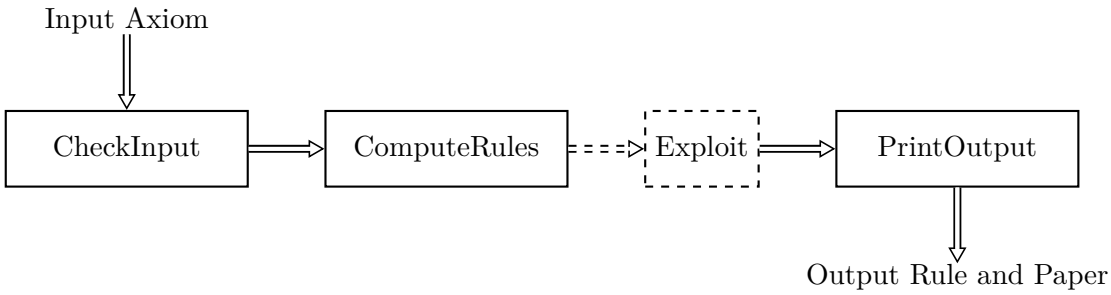

Figure 4.1: The general design of TINC

are covered by the tool. It is the responsibility of the first component, *CheckInput*, to check whether the input meets the syntactic requirements. The second component, *ComputeRules*, is the core functionality of the tool. It contains the implementation of the systematic procedure for transforming an axiom to an equivalent set of inference rules. The third component, *Exploit*, adds an optional feature to the tool. It uses the generated calculus to investigate the sufficient conditions of the properties of the formalized logic. Currently it is only implemented for the AxiomCalc and Paralyzer. The last component, *PrintOutput*, is responsible for displaying the result of the second and third components to the user. Furthermore, it generates a LaTeX document with those results.

4.3 Data structure for inference rules

The data structure chosen for representing a sequent is a list of two elements where the first one represents the antecedent and the second one represents the succedent of the sequent. For instance, a sequent of the form $\vdash a, b, c$ is represented by the following list:

```
[[], [a, b, c]]
```

Note that even though the scope of this thesis only deals with single-sided multi-conclusion sequents, the antecedent is not ignored from the chosen data structure, but it is rather represented by an empty list. In this way, it is easier to extend this implementation to deal with sequents having non-empty antecedents. On the other hand, the succedent is always represented by a flat list i.e. a list that doesn't contain another list as its element.

Since hypersequents are sets of sequents, we can represent them in Prolog as list of sequents. For instance, the hypersequent $\vdash a, b, c \mid \vdash d, e, f \mid \vdash g, h, i$ is represented in Prolog as

```
[ [[], [a, b, c]], [[], [d, e, f]], [[], [g, h, i]] ]
```

Finally, an inference rule can be represented as a list of two elements: *premises* which are sets of hypersequents and *conclusion* which is a single hypersequent. For instance, a rule of the following form

$$\frac{\vdash a, b| \vdash c, d \qquad \vdash d, e| \vdash f, g}{\vdash a, c| \vdash d, b| \vdash d, f| \vdash e, g}$$

would be represented in Prolog as:

```
[
    [ [[[],[a,b]], [[],c,d]]], [[[],[d,e]], [[],[f,g]]] ],
    [ [[],[a,c]], [[],[b,d]], [[],[d,f]], [[],[e,g]] ]
]
```

Such a data structure allows for any kind of manipulation over the premises or the conclusion of a rule.

4.4 The InvAxiomCalc tool

This section provides the implementation details of the systematic procedure for transforming an axiom into an equivalent set of analytic rules. In Subsection 4.4.1 we identify the class in the substructural hierarchy where the axiom belongs to, as defined in Section 3.1. Subsection 4.4.2 describes the transformation of an axiom into an equivalent $\mathcal{N}_2 - normal$ form, see Definition 3.1.3. Subsection 4.4.3 provides the implementation steps for transforming an axiom into a set of (hyper)sequent rules. And the implementation steps for transforming a (hyper)sequent rule into an analytic rule are given in Subsection 4.4.4.

The subject of this thesis is to extend the TINC system with a new tool called InvAxiomCalc, which implements the theoretical results in [CST09]. InvAxiomCalc follows the general design of the existing tools in TINC (Figure 4.1). It takes HMALL as a base hypersequent calculus and axioms over the language of *Multiplicative Additive Linear Logic* (i.e. classical linear logic without exponentials) to extend it. The formulas in this language are generated from the following grammar:

$$\mathcal{F} ::= \mathcal{V} \mid \mathcal{V}^{\perp} \mid \bot \mid \top \mid 1 \mid 0 \mid \mathcal{F} \parr \mathcal{F} \mid \mathcal{F} \,\&\, \mathcal{F} \mid \mathcal{F} \otimes \mathcal{F} \mid \mathcal{F} \oplus \mathcal{F} \tag{4.1}$$

where $\mathcal{V}$ and $\mathcal{V}^{\perp}$ are the set of propositional variables and their duals. We will use the convention in Table 4.1 for representing the binary connectives of the language in Prolog. We define `v` and `&` as new infix operations in Prolog. On the other hand, the operations `*` and `+` are predefined in Prolog and can be used for our purposes.

Even though the general architectural design of InvAxiomCalc is similar with the one in Figure 4.1, each component contains an implementation with specific predicates related to the tool under consideration. Table 4.2 shows the most important predicates for each component of the new tool. The following subsections will give a more detailed description of each component

Logical binary connective		Prolog representation
$\parr$	$\longrightarrow$	+
$\&$	$\longrightarrow$	&
$\otimes$	$\longrightarrow$	*
$\oplus$	$\longrightarrow$	v

Table 4.1: Convention of the language symbols in Prolog

Component	**Defined predicates**	
CheckInput	`axiom2tex`	
ComputeRules	`is_in_class/2`	finds the class of an axiom
	`to_n2_nromal/2`	converts an $\mathcal{N}_2$ axiom to normal form
	`aths_ms_r/3`	converts an axiom to set of hypersequent rules
	`htoa_ms_r/2`	converts a hypersequent rule to analytic
PrintOutput	`print_rule/2`	displays the generated rules
	`tex_out/2`	generates a LaTeX document with the new calculus

Table 4.2: The most important Prolog predicates defined in each component

4.4.1 Identifying the Axiom's Class

First, let's recall the recursive definition of the substructural hierarchy as given in [CST09]. Let $\mathcal{A}$ be the set of atomic formulas and their negations. Then the class of axioms $\mathcal{P}_n, \mathcal{N}_n$ for $n \geq 0$ is defined as follows:

$$\begin{aligned} \mathcal{P}_0 &::= \mathcal{A} \qquad \mathcal{P}_{n+1} ::= \mathcal{N}_n \mid \mathcal{P}_{n+1} \otimes \mathcal{P}_{n+1} \mid \mathcal{P}_{n+1} \oplus \mathcal{P}_{n+1} \mid 1 \mid 0 \\ \mathcal{N}_0 &::= \mathcal{A} \qquad \mathcal{N}_{n+1} ::= \mathcal{N}_n \mid \mathcal{N}_{n+1} \,\&\, \mathcal{N}_{n+1} \mid \mathcal{N}_{n+1} \parr \mathcal{N}_{n+1} \mid \top \mid \bot \end{aligned} \tag{4.2}$$

Notice that in the base case, we have $\mathcal{N}_0 = \mathcal{P}_0 = \mathcal{A}$. Furthermore, $\mathcal{P}_{n+1} \supset \mathcal{N}_n$ and $\mathcal{N}_{n+1} \supset \mathcal{P}_n$ and the constants reside on level 1, namely: $1, 0 \in \mathcal{P}_1$, $\top, \bot \in \mathcal{N}_1$.

In order to identify the class of a given axiom, we use a divide and conquer approach. First, we decompose the axiom into sub-components according to the most binding logical connective until ending up with atomic formulas or constant symbols. At this point, we can identify the class of each leaf component because atomic formulas reside in both $\mathcal{P}_0$ and $\mathcal{N}_0$, and constant symbols reside either in $\mathcal{P}_1$ or $\mathcal{N}_1$. Afterwards, having the class of two children components we can determine the class of the parent component based on the Definition 4.2.

We also recall the subclass $\mathcal{P}_3'$ of $\mathcal{P}_3$ which is defined as follows:

$$\mathcal{P}_3' ::= \mathcal{N}_2 \ \& \ 1 \mid \mathcal{P}_3' \otimes \mathcal{P}_3' \mid \mathcal{P}_3' \oplus \mathcal{P}_3' \mid 1 \mid 0 \tag{4.3}$$

A Prolog implementation of the divide and conquer procedure described above, is rather straightforward. In the Listing 4.1 is given a code snippet for checking whether an axiom falls into class $\mathcal{P}_n$ for $n > 0$. The first parameter `A` stands for the axiom to be checked, while the second parameter stands for the hierarchy level of $\mathcal{P}$ classes. As a base case, the predicate evaluates to true if and only if the class level is 1 and the axiom is an (possibly signed) atomic formula, or if the axiom is either of the constants 0 or 1 and the class level is bigger than 0. In the recursive steps, we can define the rest of possible cases for obtaining $\mathcal{P}_n$ classes by using recursion. Namely, a recursive step which allows for having class $\mathcal{P}_{n+1}$ as a superset of class $\mathcal{N}_n$ (commented as `case 5` in the code), and two recursive cases which allow for having axioms in class $\mathcal{P}_n$ as $\oplus$ and $\otimes$ concatenations of axioms within the class $\mathcal{P}_n$ (commented in the code as `case 6` and `case 7`).

Listing 4.1: Checking if the axiom belongs to the given positive class

```
is_pos_axiom(A, 1) :-
  is_atom(A).    % case 1- when the axiom is an atom

is_pos_axiom(-A, 1) :-
  is_atom(A).    % case 2 - when the axiom is a negated atom

is_pos_axiom(1, N) :-
  N > 0.     % case 3 - the axiom is the constant 1

is_pos_axiom(0, N) :-
  N > 0.     % case 4 - when the axiom is the constant 0

is_pos_axiom(A, N1) :-
  N1 > 0,     % case 5 - when the axiom is
  N is N1-1,    % in the negative class one layer below
  is_neg_axiom(A, N).

is_pos_axiom(A v B, N) :-
  N > 0,
  is_pos_axiom(A, N),   % case 6 - when the axiom is a 'v'
  is_pos_axiom(B, N).   % concatenation of two axioms in class Pn

is_pos_axiom(A * B, N) :-
  N > 0,
  is_pos_axiom(A, N),   % case 7 - when the axiom is a '*'
  is_pos_axiom(B, N).   % concatenation of two axioms in class Pn
```

Checking if an axiom falls into a class $\mathcal{N}_n$ for $n > 0$ is done by replacing the constants `1`, `0` by $\top$, $\bot$ (see case 3 and 4 in Listing 4.1), and by replacing the logical connectives `v`,

$\star$ in the recursive steps by + and &. The predicate for checking whether an axiom falls into the class $\mathcal{P}_3'$ is slightly different. Since the hierarchy level in this case is fixed, there is no need for having a second parameter for it. Atomic formulas are omitted from the base case, but there is a new case for checking whether the axiom is of the form $\mathcal{N}_2$ & 1. Listing 4.2 shows the Prolog implementation for identifying the class $\mathcal{P}_3'$.

Listing 4.2: Identifying class $\mathcal{P}_3'$

```
is_pos_3prime_axiom(1).  % 1 is in P3 prime
is_pos_3prime_axiom(0).  % 0 is in P3 prime

is_pos_3prime_axiom(A & 1) :-
  is_neg_axiom(A, 2).  % N2 & 1 is P3 prime

is_pos_3prime_axiom(A v B) :-
  is_pos_3prime_axiom(A),
  is_pos_3prime_axiom(B).  % P3_prime v P3_prime is also P3_prime

is_pos_3prime_axiom(A * B) :-
  is_pos_3prime_axiom(A),
  is_pos_3prime_axiom(B).  % P3_prime * P3_prime is also P3_prime
```

4.4.2 Preprocessing the given axiom

After having defined the class hierarchy to which the axiom belongs to, another preprocessing step is needed before applying the systematic procedure for generating the inference rules. In particular, if an axiom $\phi \in \mathcal{N}_2$ then it has to be transformed into an equivalent set of $\mathcal{N}_2 - normal$ axioms $\{\psi_1, ..., \psi_n\}$, see Proposition 3.1.2 and Definition 3.1.3. If the axiom $\phi \in \mathcal{P}_3'$, then it has to be transformed into an equivalent set $\{\phi_1, ..., \phi_n\}$ of formulas such that $\phi_i = \oplus_j(\chi_{i,j})_{\&1}$ where each $\chi_{i,j}$ is $\mathcal{N}_2 - normal$, see Proposition 3.1.4.

Listing 4.3 shows a code snippet for transforming a general $\mathcal{N}_2$ axiom into a set of equivalent $\mathcal{N}_2 - normal$ axioms. Since the shape of a general axiom $\phi \in \mathcal{N}_2$ is $\&_i(\parr_j \psi_{i,j})$ with $\psi_{i,j} \in \mathcal{P}_1$ and the shape of a general axiom in $\mathcal{N}_2 - normal$ is $\parr_j \psi_{i,j}$, then the set of formulas which ϕ is equivalent to, can be obtained by splitting ϕ on the most binding occurrences of &. The predicate `to_n2_normal/2` expects the first argument to be a $\mathcal{N}_2$ axiom. The value of the second argument, will be step-wisely constructed to represent the list of the $\mathcal{N}_2 - normal$ axioms. If & occurs as a most binding connective (i.e. the axiom has the form `Ax1 & Ax2`, see `case 1` in Listing 4.3), then first a recursive call is made to check for other & connectives in the left conjunct `Ax1`, and then `Ax2` is appended to the list of $\mathcal{N}_2$ axioms resulting from the recursive call. Otherwise, if the axiom is atomic (`case 2` in Listing 4.3) or if the most binding connection is not & (`case 3` in Listing 4.3) then the axiom is already $\mathcal{N}_2 - normal$ and a list with a single element is created as a resulting list.

Listing 4.3: Converting $\mathcal{N}_2$ axioms to $\mathcal{N}_2 - normal$

```
to_n2_normal(Ax1 & Ax2 , N2normal1) :-  % case 1
  to_n2_normal(Ax1, N2normal0),   % & concatenations can be splitted
  append([Ax2], N2normal0, N2normal1). % to list of conjuncts

to_n2_normal(Ax, N2normal):-  % case 2
  atomic(Ax),
  append([Ax], [], N2normal).

to_n2_normal(Ax, N2normal):- % case 3
  compound(Ax),
  append([Ax], [], N2normal).
```

Notice that in both cases, $\mathcal{N}_2$ and $\mathcal{P}_3'$, a set $\{\psi_1, ..., \psi_j\}$ of formulas is created which is equivalent to the original axiom. The systematic procedure for transforming an axiom to an inference rule, is going to be applied to each of such formulas ψ_i. Thus, ending up with a set of rules which are equivalent with the input axiom.

4.4.3 From Axioms to Hypersequent Rules

After the preprocessing step described above, it is time to start the first steps of the systematic procedure. If an input axiom $\phi \in \mathcal{N}_2$ is equivalent with the set $\{\psi_i \mid \psi \in \mathcal{N}_2 - normal\}$, then for each ψ_i we introduce a rule without premises:

$$\frac{}{\vdash \psi_i} \tag{4.4}$$

In particular, if we consider the Nelson axiom $nel ::= ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)) \parr A^{\perp} \parr B^{\perp}$ which is already in $\mathcal{N}_2 - normal$ form then the rule that we introduce is:

$$\frac{}{\vdash ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)) \parr A^{\perp} \parr B^{\perp}} \tag{4.5}$$

having the following Prolog representation:

$$\frac{}{\texttt{[[], [((a * a * b) v (b * b * a)) + -a + -b]]}}\ \texttt{[]} \tag{4.6}$$

If an axiom $\phi \in \mathcal{P}_3'$, is equivalent with the set $\{\psi_i \mid \psi_i = \oplus_j(\chi_{i,j})_{\&1},\ \chi_{i,j} \in \mathcal{N}_2 - normal\}$ then we introduce the hypersequent rule:

$$\frac{}{\vdash \chi_{i,1} \mid ... \mid \vdash \chi_{i,k}} \tag{4.7}$$

In particular, if we consider the linearity axiom $lin ::= (A^{\perp} \parr B)_{\&1} \oplus (B^{\perp} \parr A)_{\&1} \in \mathcal{P}_3'$ then the rule that we introduce is:

$$\top\ \frac{}{\mathcal{H}|\vdash \Gamma, \top} \qquad \bot\ \frac{\mathcal{H}|\vdash \Gamma}{\mathcal{H}|\vdash \Gamma, \bot} \qquad \parr\ \frac{\mathcal{H}|\vdash \Gamma, A, B}{\mathcal{H}|\vdash \Gamma, A \parr B} \qquad \&\ \frac{\mathcal{H}|\vdash \Gamma, A \qquad \mathcal{H}|\vdash \Gamma, B}{\mathcal{H}|\vdash \Gamma, A\&B}$$

Table 4.3: The invertible rules of the Hypersequent system HMALL

$$\frac{}{\vdash A^{\perp} \parr B \mid \vdash B^{\perp} \parr A} \tag{4.8}$$

having the Prolog representation:

$$\frac{\texttt{[]}}{\texttt{[[[],[-a + b]], [[],[-b + a]]]}} \tag{4.9}$$

Since in (4.8) the conclusion is a hypersequent, its Prolog representation (4.8) is a list of 2 sequents, unlike the rule (4.5) where the conclusion is a single sequent and the corresponding Prolog representation (4.6) is a list of a single sequent.

Applying Invertible Rules The next step is to utilize the invertible rules $\top$, $\bot$, & and $\parr$ of HMALL, Table 4.3 so that we can transform the initial rules (4.4) or (4.7) into equivalent ones, having less logical connectives. It is worth pointing out, that since ψ_i and $\chi_{i,j}$ are all $\mathcal{N}_2 - normal$, they do not have any outermost & logical connective, as they are already handled in the preprocessing step. Hence, in this step, we only need to take care of $\parr$, $\bot$ and $\top$.

The inverted $\parr$ rule can be applied to a the conclusion, just by splitting ψ_i or $\chi_{i,j}$ by the occurrence of $\parr$. For example, applying the inverted $\parr$ rule to (4.5) and (4.8) will produce the following transformation:

$$\begin{aligned} &\frac{}{\vdash ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)) \parr A^{\perp} \parr B^{\perp}} \\ \Longrightarrow\ &\frac{}{\vdash ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)), A^{\perp}, B^{\perp}} \end{aligned} \tag{4.10}$$

$$\frac{}{\vdash A^{\perp} \parr B \mid \vdash B^{\perp} \parr A} \quad \Longrightarrow \quad \frac{}{\vdash A^{\perp},\ B \mid \vdash B^{\perp},\ A} \tag{4.11}$$

Having respectively the following Prolog representations:

$$\frac{}{\texttt{[[[], [((a * a * b) v (b * b * a)), -a, -b]]]}} \tag{4.12}$$

$$\frac{}{\texttt{[[[],[-a, b]], [[],[-b, a]]]}} \tag{4.13}$$

In Prolog this is achieved by recursively collecting the $\parr$ concatenations in ψ_i or $\chi_{i,j}$ into a new list and replacing the succedent of the corresponding sequent in the conclusion with the new list.

Implementing the application of the rules $\bot$ and $\top$ is rather easy. The former is done by searching for $\bot$ elements in the succedents. If there are any, they are just removed. The latter is done by searching for $\top$ elements in the succedents. If at least one found, then the whole sequent is removed. At the end of this step, we obtain a rule where the succedents of the hypersequents in the conclusion are formulas in $\mathcal{P}_1$.

Applying Ackerman Lemma So far, only the conclusion of the rule has been manipulated and the premises are still empty. In this step, the Ackerman Lemma 3.4 (for the sequent calculus) or 3.3.1 (for the hypersequent calculus) are used to transform the rule into an equivalent one, having a new premise for each formula in the succedents of the sequents in conclusion. For example, after applying the Ackerman Lemma in (4.10) and (4.11) we would get:

$$\frac{\vdash \Gamma, ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A))^{\bot} \qquad \vdash \Delta, (A^{\bot})^{\bot} \qquad \vdash \Sigma, (B^{\bot})^{\bot}}{\vdash \Gamma \ , \ \Delta \ , \ \Sigma} \tag{4.14}$$

$$\frac{\vdash \Gamma, (A^{\bot})^{\bot} \qquad \vdash \Delta, B^{\bot} \qquad \vdash \Sigma, (B^{\bot})^{\bot} \qquad \vdash \Theta, A^{\bot}}{\vdash \Gamma \ , \ \Delta \mid \vdash \Sigma \ \Theta} \tag{4.15}$$

Listing 4.4: Implementation of the Ackerman Lemma transformation

```
apply_equivalence_lemma(_, [], [], [], 1). % counter starts from 1

apply_equivalence_lemma(P0, P3, [H|T], C3, N2) :-
apply_equivalence_lemma(_, P1, T, C1, N0), % recursive call
apply_to_mc_sequent(P0, P2, H, C2, N0, N1), % handles the current
    sequent
N2 is N0 + N1,  % update the counter offset
append([C2], C1, C3),
append(P2, P1, P3).

apply_to_mc_sequent(P0, P2, [A, S0], C1, N0, N1) :-
set_as_premise(S0, S1, P1, N0, N1), % generate new premises
C1 = [A, S1],
append(P1, P0, P2).

set_as_premise([], [], [], _, 0).

set_as_premise([H|T], C1, P1, N0, NA1) :-
N1 is N0 + 1,
set_as_premise(T, C0, P0, N1, NA0),
NA1 is NA0 + 1, % increase counter
append(['Y'+N0], C0, C1), % append to new succedent in conclusion
apply_de_morgan(-H, H2), % push the negation using de Morgan
append([[[[],['Y'+N0, H2]]]], P0, P1). %append premise
```

Original formula		Transformed formula
$(\alpha \,\&\, \beta)^{\perp}$	$\longrightarrow$	$\alpha^{\perp} \oplus \beta^{\perp}$
$(\alpha \parr \beta)^{\perp}$	$\longrightarrow$	$\alpha^{\perp} \otimes \beta^{\perp}$
$(\alpha \otimes \beta)^{\perp}$	$\longrightarrow$	$\alpha^{\perp} \parr \beta^{\perp}$
$(\alpha \oplus \beta)^{\perp}$	$\longrightarrow$	$\alpha^{\perp} \,\&\, \beta^{\perp}$
$(\alpha^{\perp})^{\perp}$	$\longrightarrow$	α

Table 4.4: De Morgan laws in MALL

where each $\Gamma, \Delta, \Sigma, \Theta$ are fresh multiset variables. The Prolog implementation defines two predicates for accessing the elements of the conclusion: `apply_equivalence_lemma/4` for recursively fetching sequents, and `apply_to_mc_sequent/6` for fetching the elements of the succedents. The latter makes use of yet another predicate `set_as_premise/5` for adding the new multiset variable in the conclusion and for creating a new premise. A counting index is added as a suffix to the multiset variable name, to ensure its uniqueness. Listing 4.4 gives a code snippet of this transformation.

Since the formulas in the conclusion are moved to the premises in a negated form, the de Morgan laws have to be invoked for pushing the negation inside the formula. Table 4.4 shows the de Morgan laws for MALL. A divide and conquer approach is used for implementing the transformation according to the de Morgan laws. The negated formula is decomposed until ending up with propositional atoms and then it is merged back by changing the logical connectives according to Table 4.4. Continuing with the transformation of (4.14) and (4.15), we get:

$$\frac{\vdash \Gamma, (A^{\perp} \parr A^{\perp} \parr B^{\perp}) \& (B^{\perp} \parr B^{\perp} \parr A^{\perp}) \qquad \vdash \Delta, A \qquad \vdash \Sigma, B}{\vdash \Gamma \ , \ \Delta \ , \ \Sigma} \tag{4.16}$$

$$\frac{\vdash \Gamma, \ A \qquad \vdash \Delta, \ -B \qquad \vdash \Sigma, \ B \qquad \vdash \Theta, -A}{\vdash \Gamma \ , \ \Delta \mid \vdash \ \Sigma \ \Theta} \tag{4.17}$$

Which are represented in Prolog respectively as:

```
[[],[Y1,(-a + -a + -b)&(-b + -b + -a)]]     [[],[Y2,a]]     [[],[Y3,b]]
                          [[[],[Y1 , Y2, Y3]]]
```

(4.18)

```
[[],[Y1,a]]     [[],[Y2,-b]]     [[],[Y3,b]]     [[],[Y4,-a]]
                [[[], [Y1,Y2]], [[],[Y3,Y4]]]
```

(4.19)

At the end of this step, we obtain a rule whose conclusion consist of multiset variables and premises consisting of a multiset variable and the corresponding formula variable $\in \mathcal{P}_1$ in a negated form.

Applying Invertible Rules to Premises According to Proposition 3.1.1, if an axiom $\alpha \in \mathcal{P}_1$, then the axiom $\alpha^{\perp} \in \mathcal{N}_1$ because after applying the de Morgan laws, $\oplus$ is converted to & and $\otimes$ is converted to $\parr$. Hence, the formulas in the new premises obtained from the step above have only & and $\parr$ as logical connectives. The inference rules corresponding to & and $\parr$ connectives are both invertible, Table 4.3. Therefore, by utilizing the invertible rules once again (this time for the premises) we can eliminate all of the logical connectives from the premises.

The Prolog implementation defines the predicate `apply_to_premise_sequent/2` which takes the list of existing premises as the first argument, and outputs the resulting premises in the second argument. It is slightly different from the implementation of applying the invertible rules to the conclusion, because in that case the & rule was excluded since the preprocessing step eliminates the & connectives. On the other hand, if a premise contains a formula having & as the most binding connective, it will be separated into two premises according to the & rule. An example is the rule obtained in (4.16) where the result of applying de Morgan laws, leads to an occurrence of & as the most binding connective: $((A^{\perp} \parr A^{\perp} \parr B^{\perp}) \& (B^{\perp} \parr B^{\perp} \parr A^{\perp}))$. After applying the inverted & rule over the premises of (4.16) we get:

$$\frac{\vdash \Gamma, A^{\perp} \parr A^{\perp} \parr B^{\perp} \qquad \vdash \Gamma, B^{\perp} \parr B^{\perp} \parr A^{\perp} \qquad \vdash \Delta, A \qquad \vdash \Sigma, B}{\vdash \Gamma \ , \ \Delta \ , \ \Sigma} \tag{4.20}$$

Listing 4.5 shows the part of the definition of `apply_to_premise_sequent/2` which separates a premise P of the form $\mathcal{H} | \vdash \Gamma, A \& B$ into two premises $\mathcal{H} | \vdash \Gamma, A$ and $\mathcal{H} | \vdash \Gamma, B$ (i.e. applies the inverse of the & rule).

Listing 4.5: Implementation of applying the inverted & rule a sequent of a premise

```
apply_to_premise_sequent([[Antecedent, Succedent]|T], P4):-
  member(Ax1 & Ax2, Succedent),
  apply_to_premise_sequent(T, P1),
  remove(Ax1 & Ax2, Succedent, S1), % remove the exisitn premise
  append([Ax1], S1, S2),  % adds a premise with the left conjunct
  append([Ax2], S1, S3),   % adds a premise with the right conjunct
  append([[[Antecedent, S2]]], P1, P3),
  append([[[Antecedent, S3]]], P3, P4).
```

The implementation for the remaining invertible rules $\parr$, $\perp$ and $\top$ is rather easy because they do not introduce new premises. In particular: as a result of applying the inverted $\perp$ rule, the $\perp$ elements are removed from the premises, for example the premise $P_1 : \ \mathcal{H} \mid \Gamma, \perp$ is converted to $P_1' : \ \mathcal{H} \mid \Gamma$; premises having a $\top$ element are completely removed; and if $\parr$ occurs as a most binding connective over a formula ϕ in some premise, then the ϕ will be replaced with the set of formulas obtained from splitting ϕ by the occurrences of $\parr$. For example, after applying the inverted $\parr$ rule in (4.20), we get:

$$\frac{\vdash \Gamma, A^{\perp}, A^{\perp}, B^{\perp} \qquad \vdash \Gamma, B^{\perp}, B^{\perp}, A^{\perp} \qquad \vdash \Delta, A \qquad \vdash \Sigma, B}{\vdash \Gamma\ ,\ \Delta\ ,\ \Sigma} \tag{4.21}$$

having the following Prolog representation:

```
[[],[Y1, -a, -a, -b]]    [[],[Y1, -b, -b, -a]]    [[],[Y2,a]]    [[],[Y3,b]]
                          [[],[Y1 , Y2, Y3]]
```
(4.22)

As a result of this step, the formulas occurring in the premises of the rules are free from logical connectives. Furthermore, the conclusion consist only of multiset variables which occur with the same polarity also in the premises. This result is claimed by Theorem 3.2.2 and 3.3.3, too.

4.4.4 From Hypersequent to Analytic Rules

This step concludes the generation of the inference rules by transforming the hypersequent rule obtained in the previous step to a *completed* rule (see Definition 3.4.3), i.e. a rule which satisfies the No Forumula Variable (NFV) property, the Linear Conclusion (LC) property and the Subformula Property (SP). The LC property is already satisfied by the rules obtained from the previous steps, because during the application of the Ackerman Lemma, each formula in the succedents of the conclusion is replaced with a fresh new multiset variable, (see Theorem 3.2.2 and 3.3.3). Furthermore, the multiset variables that are introduced in the conclusion by the Ackerman Lemma, are also introduced in the premises with the same polarity. Hence, the NFV and SP properties are being violated only by the *formula variables* occurring on the premises. For example, the formula variables A, B and their duals in (4.21) are only occurring in the premises, while the multiset variables are Γ, Δ and Σ are occurring both in the premises and in the conclusion.

This section describes the implementation of the procedure given in the proof of Theorem 3.4.4 which cuts all of the formula variables from the premises of the generated rule. Hence, transforming it to a *complete* rule. The formula variables are represented in Prolog as atoms starting with lower case letter. On the other hand, the multiset variables are represented as sequence of characters enclosed in quotes, starting with the uppercase letter 'Y' and followed by a counting index. In this way, we can distinguish between the formula variables and the multiset variables. The presence of the formula variables in the premises consists also the stopping criteria of the cut procedure.

Checking Acyclicity Criteria In absence of weakening rule, the cut procedure can only be applied if the rule resulting from the previous steps is *acyclic*, which means that no sequent of the form $\vdash \Gamma, A, A^{\perp}$ appears in the cut closure of the premises (see Definition 3.4.1). As a result of acyclicity, if a premise has multiple occurrences of a formula variable A (or $A^{\perp}$) then, no premise has multiple occurrences of $A^{\perp}$ (or A).

Additionally, no premise should contain a sequent having both $A^{\perp}$ and A at the same time.

We define the Prolog predicate `is_cyclic_on_mv/2` which accepts a list of predicates and a metavariable, and it is satisfied only if the list of premises is acyclic with respect to the given metavariable. The definition of `is_cyclic_on_mv/2` contains two clauses. The first clause, uses two helper predicates for counting the premises having multiple occurrences of the provided metavarialbe with positive and negative polarity. If both counters are bigger than one, then the list of premises is cyclic with. The second clause of the definition of `is_cyclic_on_mv/2` uses a third helper predicate `count_having_mixed_occurrence/3` for counting premises having occurrences of the metavariable with both polarities simultaneously. The following Listing gives the definition of `is_cyclic_on_mv/2` and the helper predicates:

```
is_cyclic_on_mv(Premises, Mv) :-
  count_having_multi_occurrence(Premises, Mv, R1),
  count_having_multi_occurrence(Premises, -Mv, R2),
  R1 > 0,
  R2 > 0.

is_cyclic_on_mv(Premises, Mv) :-
  count_having_mixed_occurrence(Premises, Mv, R1),
  R1 > 0.

count_having_multi_occurrence([], _, 0).
count_having_multi_occurrence([[[_, S]]|T], Mv, NumPrMultiMv) :-
  count_having_multi_occurrence(T, Mv, Num0),
  count_repetitions(S, Mv, NumRepetitions),
  ( (NumRepetitions > 1) ->
    NumPrMultiMv is Num0 + 1
    ;NumPrMultiMv is Num0
  ).

count_having_mixed_occurrence([], _, 0).
count_having_mixed_occurrence([[[_, S]]|T], Mv, NumPrMixedMv) :-
  count_having_mixed_occurrence(T, Mv, Num0),
  has_mixed_polarities_mv(S, Mv, R),
  NumPrMixedMv is Num0 + R.
```

Considering the rule in (4.21), we have the following premises containing the metavariable A:

$$\mathcal{G}_A = \{\vdash \Gamma,\ A^{\perp},\ A^{\perp},\ B^{\perp};\ \vdash \Gamma,\ B^{\perp},\ B^{\perp},\ A^{\perp};\ \vdash \Delta,\ A\} \tag{4.23}$$

Which are represented in Prolog as:

```
P2 ::= [ [[],[Y1, -a, -a, -b]], [[],[Y1, -b, -b, -a]], [[],[Y2, a]] ]   (4.24)
```

none of the clauses of the definition of `is_cyclic_on_mv/2` is satisfied, and therefore, the rule in 4.21 is indeed acyclic.

If the rule is cyclic, then no further transformation is possible, unless the weakening rule is present in the base calculus. A predicate `convert_to_cyclic/3` is defined to use weakening rule (w) for eliminating the occurrences of the metavariables that are making the rule cyclic. `convert_to_cyclic/3` takes a list of cyclic premises with respect to a metavariable, and produces a list of acyclic premises. The implementation is shown in the following Listing:

```
convert_to_acyclic(Premises, Mv, AcPremises) :-
  is_cyclic_on_mv(Premises, Mv),
  count_having_multi_occurrence(Premises, Mv, R1),
  count_having_multi_occurrence(Premises, -Mv, R2),
  count_having_mixed_occurrence(Premises, Mv, R3),
  neg_free_mv(Mv, NegFreeMv),
  convert_to_acyclic_on_mv(Premises, AcPremises, NegFreeMv, R1, R2,
     R3).

convert_to_acyclic(Premises, Mv, Premises) :-
\+ is_cyclic_on_mv(Premises, Mv).

convert_to_acyclic_on_mv(CPremises, AcPremises, Mv, R1, R2, _):-
  R1 > 0,
  R2 > 0,
  apply_weakening_on_mv(CPremises, -Mv, AcPremises).
```

At the end of this step, we obtain an acylic (hyper)sequent rule. A separate module `mall_acyclicity` contains all the Prolog predicates for checking the acyclicity conditions and invoking the weacening rule to create acyclic rules. A flag for indicating whether weakening is required is also generated. This flag will be used later on by the modules which print the generated results for the user.

Applying the Combinatorial Cut In this step, the set of premises $\mathcal{G}_A^+ \cup \mathcal{G}_A^+$ is replaced with the set $\mathcal{G}_A^{cut}$ which is obtained by repeatedly applying the cut rule over premises $P_i^+ \in \mathcal{G}_A^+$ and $P_j^- \in \mathcal{G}_A^-$ until all occurrences of A are eliminated, see the proof of Theorem 3.4.4. If either $\mathcal{G}_A^+ = \varnothing$ or $\mathcal{G}_A^- = \varnothing$ then also $\mathcal{G}_A^{cut} = \varnothing$. Otherwise, $\mathcal{G}_A^{cut}$ will consist of premises obtained by applying each possible combination of cut between $P_i^+ \in \mathcal{G}_A^+$ and $P_j^- \in \mathcal{G}_A^-$. W.l.g. lets assume that every premise $P_i^+ \in \mathcal{G}_A^+$ has exactly one occurrence of A and therefore, from the acyclicity, premises $P_j^- \in \mathcal{G}_A^-$ have one or more occurrences of $A^\perp$. The number of the new premises introduced by cutting the occurrences of $A^\perp$ in P_j^- with the premises in $\mathcal{G}_A^+$ is given by the following formula:

$$NumNewPr = \frac{(|\mathcal{G}_A^+| + R_j - 1)!}{(|\mathcal{G}_A^+| - 1)! \cdot R_j!} \tag{4.25}$$

where R_j is the number of the occurrences of $A^\perp$ in P_j^-. The resulting premises in $\mathcal{G}_A^{cut}$ is the union of all premises obtained by eliminating $A^\perp$ from every premise $P_j^- \in \mathcal{G}_A^-$.

The Prolog implementation defines a predicate `combinatorial_merge_neg/5` for generating the list of premises resulting from cutting the occurrence(s) of a formula variable $A^{\perp}$ in $P_j^- \in \mathcal{G}_{\mathcal{A}}^-$ with each $P_i^+ \in \mathcal{G}_A^+$. It takes as parameters the list of premises $\mathcal{G}_A^-$ having at least one occurrence of $A^{\perp}$, the list of premises $\mathcal{G}_A^+$ having exactly one occurrence of A, the formula variable to be cut A, and the size of $\mathcal{G}_A^+$. One of the parameters is used for outputting the result of the cut. An auxiliary predicate `generate_combination/6` is used for creating the combinations of the formulas from premises in $\mathcal{G}_A^+$ which will be used for replacing $A^{\perp}$ in P_j^-. `generate_combination/6` takes as parameter the number of repetitions of $A^{\perp}$ in P_j^-, the set $\mathcal{G}_A^+$ and the total number of combinations to be generated (calculated with the formula (4.25)). One of the parameters is used again for outputting the result, which in this case is the set of all combinations R of premises $P_i^+ \in \mathcal{G}_A^+$ obtained by combining k premises P_i^+ and removing the occurrences of A, where k is the number of occurrences of $A^{\perp}$ in P_j^-. Yet another auxiliary predicate `create_new_premises/4` is used for creating a list of premises out of the set R and the remaining part of P_j^-.

Similarly, a predicate `combinatorial_merge_pos/5` is defined for handling the cases where premises $P_j^- \in \mathcal{G}_A^-$ have exactly one occurrence of $A^{\perp}$ and therefore, premises in $P_i^+ \in \mathcal{G}_A^+$ contain one or more occurrences of A. The behavior of the predicate is the same as `combinatorial_merge_neg/5` but the sign of the formula variable A is inverted.

Considering $\mathcal{G}_A^+$ and $\mathcal{G}_A^-$ given in 4.23, the set $\mathcal{G}_A^{cut}$ can be obtained by cutting A as in the following (note that since $|\mathcal{G}_A^+| = 1$ there is only one possibility for choosing the cut formula in each step):

$$\dfrac{\dfrac{\vdash\ \Delta, A \qquad \dfrac{\vdash\ \Gamma, A^{\perp}, A^{\perp}, B^{\perp}}{\vdash\ \Gamma, \Delta, A^{\perp}, B^{\perp}}\,(\text{cut})}{\vdash\ \Gamma, \Delta, \Delta, B^{\perp}}\,(\text{cut}) \qquad \dfrac{\vdash\ \Delta, A \qquad \vdash\ \Gamma, B^{\perp}, B^{\perp}, A^{\perp}}{\vdash \Gamma, B^{\perp}, B^{\perp}, \Delta}\,(\text{cut})}{} \tag{4.26}$$

$$\mathcal{G}_A^{cut} = \{\vdash \Gamma, \Delta,\ \Delta,\ B^{\perp};\ \vdash \Gamma,\ B^{\perp},\ B^{\perp},\ \Delta\} \tag{4.27}$$

At this point, $\mathcal{G}_A^+ \cup \mathcal{G}_A^-$ will be replaced in 4.21 by $\mathcal{G}_A^{cut}$ in 4.27 to obtain:

$$\dfrac{\vdash \Gamma,\ \Delta,\ \Delta,\ B^{\perp} \qquad \vdash \Gamma,\ B^{\perp},\ B^{\perp},\ \Delta \qquad \vdash \Sigma,\ B}{\vdash\ \Gamma\ ,\ \Delta\ ,\ \Sigma} \tag{4.28}$$

where all occurrences of A are eliminated form the premises. In a similar way we eliminate the occurrences of B, which in this case, is the only formula variable left in the premises. This concludes the transformation of the Nelson axiom into an analytic rule *nel*:

$$\dfrac{\vdash \Gamma,\ \Delta,\ \Delta,\ \Sigma \qquad \vdash \Gamma,\ \Sigma,\ \Sigma,\ \Delta}{\vdash\ \Gamma\ ,\ \Delta\ ,\ \Sigma} \tag{4.29}$$

A summary of the transformation steps of the Nelson axiom $((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)) \parr A^\perp \parr B^\perp$ into the inference rule *nel*, is given in the following:

$$
\begin{array}{l}
\dfrac{}{\vdash ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)) \parr A^\perp \parr B^\perp} \\[2ex]
\overset{\textit{invertible to conclusion}}{\Longrightarrow} \quad \dfrac{}{\vdash ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)), A^\perp, B^\perp} \\[2ex]
\overset{\textit{eq. lemma}}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A))^\perp \qquad \vdash \Delta, (A^\perp)^\perp \qquad \vdash \Sigma, (B^\perp)^\perp}{\vdash \ \Gamma \ , \ \Delta \ , \ \Sigma} \\[2ex]
\overset{\textit{de Morgan}}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, (A^\perp \parr A^\perp \parr B^\perp) \& (B^\perp \parr B^\perp \parr A^\perp) \qquad \vdash \Delta, A \qquad \vdash \Sigma, B}{\vdash \ \Gamma \ , \ \Delta \ , \ \Sigma} \\[2ex]
\overset{\textit{invertible to premises}}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, A^\perp, \ A^\perp, \ B^\perp \qquad \vdash \Gamma, B^\perp, B^\perp, \ A^\perp \qquad \vdash \Delta, A \qquad \vdash \Sigma, B}{\vdash \ \Gamma \ , \ \Delta \ , \ \Sigma} \\[2ex]
\overset{\textit{cutting } A}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, \Delta, \ \Delta, \ B^\perp \qquad \vdash \Gamma, B^\perp, B^\perp, \ \Delta \qquad \vdash \Sigma, B}{\vdash \ \Gamma \ , \ \Delta \ , \ \Sigma} \\[2ex]
\overset{\textit{cutting } B}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, \Delta, \ \Delta, \ \Sigma \qquad \vdash \Gamma, \Sigma, \Sigma, \ \Delta}{\vdash \ \Gamma \ , \ \Delta \ , \ \Sigma} \ (\text{nel})
\end{array}
\tag{4.30}
$$

Similarly, the transformation steps for the linearity axiom are:

$$
\begin{array}{l}
\dfrac{}{\vdash A^\perp \parr B \mid \vdash B^\perp \parr A} \\[2ex]
\overset{\textit{invertible to conclusion}}{\Longrightarrow} \quad \dfrac{}{\vdash A^\perp, \ B \mid \vdash B^\perp, \ A} \\[2ex]
\overset{\textit{eq. lemma}}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, (A^\perp)^\perp \qquad \vdash \Delta, B^\perp \qquad \vdash \Sigma, (B^\perp)^\perp \qquad \vdash \Theta, A^\perp}{\vdash \ \Gamma \ , \ \Delta \mid \vdash \ \Sigma \ \Theta} \\[2ex]
\overset{\textit{de Morgan}}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, \ A \qquad \vdash \Delta, \ B^\perp \qquad \vdash \Sigma, \ B \qquad \vdash \Theta, A^\perp}{\vdash \ \Gamma \ , \ \Delta \mid \vdash \ \Sigma \ \Theta} \\[2ex]
\overset{\textit{cutting } A}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, \ \Theta \qquad \vdash \Delta, \ B^\perp \qquad \vdash \Sigma, \ B}{\vdash \ \Gamma \ , \ \Delta \mid \vdash \ \Sigma \ \Theta} \\[2ex]
\overset{\textit{cutting } B}{\Longrightarrow} \quad \dfrac{\vdash \Gamma, \ \Theta \qquad \vdash \Delta, \ \Sigma}{\vdash \ \Gamma \ , \ \Delta \mid \vdash \ \Sigma \ \Theta} \ (\text{com})
\end{array}
\tag{4.31}
$$

Note that in (4.31), after applying the de Morgan law, all of the logical connectives are eliminated, and there is no need to apply the invertible rules to the premises.

At the end of this step, every formula variable is eliminated from the premises. Thus, ending up with a *completed rule* consisting only of multiset variables and which is equivalent with the acyclic rule given as the input to this step.

4.5 Automated Paper Generation

InvAxiomCalc represents the result of the systematic procedure in two ways. If you are running InvAxiomCalc locally, the generated rules are displayed in the Prolog terminal. Additionally, a Latex paper containing the transformation steps and the the set of the generated rules is created automatically. See the appendix for some examples of the generated papers. The preliminaries of HMALL and a description of the systematic procedure in [CST09] are hard coded in a predefined Latex template. InvAxiomCalc generates a file called `InvAxiomCalc.sty` that defines new Latex commands for the following:

- The name of the base calculus which is either MALL or HMALL
- The Latex representation of the input axiom.
- The Latex representation of the rules obtained after applying Ackerman Lemma. See Sections 3.2, 3.3 and the implementation in Section 4.4.3.
- A flag indicating whether the rules obtained after applying the Ackerman Lemma are acyclic.
- A Latex representation of the analytic rules.

The generated file with the defined commands is imported as a package in the main template and thus allowing for generating an article with the results of the systematic procedure initiated with the axiom provided by the user. The result of InvAxiomCalc for the Nelson axiom in $\mathcal{N}_2$ and Linearity axiom in $\mathcal{P}_3^{'}$ (see Table 2.1) are shown respectively in Appendix A and Appendix B. Whereas the result for the *cancel* axiom $(A \parr A^{\perp})^{\perp}$ (see Example 3.4.2) which leads to cyclic structural rule After applying the Ackerman Lemma is shown in Appendix C.

Definite Clause Grammar (DCG) are used to implement the translation from Prolog to Latex. For example, the transformation of an axiom to Latex is done through `axiom2tex` rule:

Listing 4.6: Translation of an axiom from Prolog to Latex

```
bs     --> [92]. % ASCII code for the backslash \

axiom2tex(-X)  --> " ", axiom2texP1(X), "^", axiom2tex(bot).
axiom2tex(X & Y) --> axiom2texP2(X), " ", bs, "& ", axiom2texP2(Y).
axiom2tex(X v Y) --> axiom2texP3(X), " ", bs, "oplus ",
   axiom2texP3(Y).
axiom2tex(X * Y) --> axiom2texP4(X), " ", bs, "otimes ",
   axiom2texP4(Y).
axiom2tex(X + Y) --> axiom2texP5(X), " ", bs, "nand ",
   axiom2texP5(Y).
axiom2tex(1)  --> "1".
axiom2tex(bot) --> bs, "bot".
axiom2tex(top)  --> bs, "top".
axiom2tex(0)  --> "0".
axiom2tex(a)  --> "A".
...
axiom2tex(z)  --> "Z".
axiom2texP2(-X) --> " ", axiom2texP1(X), "^", axiom2tex(bot).
axiom2texP2(X & Y) --> axiom2texP2(X), " ", bs, "& ",
   axiom2texP2(Y).
axiom2texP2(X v Y) --> "(", axiom2texP3(X), " ", bs, "oplus ",
   axiom2texP3(Y), ")".
axiom2texP2(X * Y) --> "(", axiom2texP4(X), " ", bs, "otimes ",
   axiom2texP4(Y), ")".
axiom2texP2(X + Y) --> "(", axiom2texP5(X), " ", bs, "nand ",
   axiom2texP5(Y), ")".
axiom2texP2(X) --> {atomic(X);member(X, [0, 1, bot, top])},
   axiom2tex(X).
```

DCGs consist of a set of grammar rules. The body of a grammar rule may contain three type of terms: *callable terms* which are interpreted as references to grammar rules; code between curly braces `{...}` that is interpreted as plain Prolog code; and *lists* enclosed in square brackets `[...]` that represent a sequence of literals. We introduce one DCG rule for every logical connective of the language. In this way, it is possible to recursively transform every axiom into its corresponding Latex representation. For example, the Nelson axiom $nel ::= ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A)) \parr A^{\perp} \parr B^{\perp}$ which we represent in Prolog as `((a * a * b) v (b * b * a)) + -a + -b` is converted into Latex by the `axiom2tex` rule as:

```
((A \otimes A \otimes B) \oplus (B \otimes B \otimes A))
  \nand A^\bot \nand B^\bot
```

Transforming a (hyper)sequent rule into Latex is performed similarly. The Prolog representation of the sequents is transformed into Latex by the DCG rule `rule2tex`.

Afterwards, the Latex representation of the sequents are combined to form the Latex representation of the hypersequent rule. Listing 4.7 gives the implementation of the DCG rule `rule2tex`, and Listing 4.8 gives the recursive definition of the Prolog predicate `create_hseq_rule` which combines the output of `rule2tex` to create the full Latex representation of a hypersequent rule in HMALL

Listing 4.7: Translation of a rule from Prolog to Latex

```
rule2tex('Y'+1)  --> bs, "Gamma".
rule2tex('Y'+2)  --> bs, "Delta".
rule2tex('Y'+3)  --> bs, "Sigma".
rule2tex('Y'+4)  --> bs, "Theta".
rule2tex('Y'+N)  --> bs, "Lambda_{",nr2tex(N),"}".
rule2tex(a)   --> " ", "A".
rule2tex(b)   --> " ", "B".
...
rule2tex(Z)   --> " ", "Z".
rule2tex(-X)   --> rule2tex(X), "^{", axiom2tex(bot), "}".
rule2tex([])   --> "".
rule2tex([['HContex'] | T])  --> bs, "mathcal{H}", bs, ", | ",
    rule2tex(T).
rule2tex([[[], [H|T]] | []]) --> rule2tex([[],[H|T]]).
rule2tex([[[], [H|T]] | T2]) --> rule2tex([[],[H|T]]) , "|",
    rule2tex(T2).
rule2tex([[], [H|T]])  --> bs, "vdash", rule2tex([H|T]).
rule2tex([H | []])   --> rule2tex(H).
rule2tex([H | T])   --> rule2tex(H), ",", rule2tex(T).
```

Listing 4.8: Creating the Latex representation of a hypersequent rule

```
create_hseq_rule(Prem, Con) :-
  length(Prem,N),
  member(N, [2,4]),
  create_hseq_premises(Prem, TPrem),
  nl,
  N1 is N/2+1,
  printBinPremises(1, TPrem, N1, 1),
  create_hseq_conclusion(Con, TCon1),
  append([],TCon1,TCon),
  print_phrase(texNewBinCon(TCon)).

create_hseq_premises([], []).
create_hseq_premises([H|T], TPrem) :-
  append([['HContex']], H, H1),
  rule2tex(H1, TH, []),
  create_hseq_premises(T, TPrem1),
  append([TH],TPrem1,TPrem).
```

4.6 Prolog Unit Tests

One of the most important Quality Assurance measurements during software development is automatic software testing. Tests allow for validating the final system. Additionally, they offer important advantages like:

- documentation on how the code should be used.
- validating the claims made on the Prolog implementation.
- identifying accidental changes or side effects as a result of modifying different parts of the code.

PlUnit is a unit-tests framework which was developed for SWI-Prolog and allows for writing unit tests in Prolog. Tests are enclosed by directives `begin_tests/1,2` and `end_tests/1`. They can either be embedded within the main Prolog module, or they can be written in separate test files which import the files to be tested. The entry points are defined by rules using the head `test(Name)` or `test(Name, Options)`, where `Name` is a ground term and `Options` is a list of additional properties of the tests. A simple unit tests for checking if the Nelson axiom belongs to class $\mathcal{N}_2$ of the substructural hierarchy is:

Listing 4.9: Unit testing class hierarchy of Nelson Axiom

```
:- begin_tests(mall_hierarchy).
  % expecting N_2 as the least negative class for Nelsion axiom
  test(is_neg_class) :-
    is_neg_class( (((a * a * b) v (b * b * a)) + -a + -b), 0, 2).

:- end_tests(mall_hierarchy).
```

Tests can be executed by loading the test file and running `run_tests/0` or `run_tests/1`. The former runs all tests defined in the loaded file, whereas the latter accepts a list of test to execute.

Listing 4.10: Unit tests taken from different modules of InvAxiomCalc

```
% applying de Morgan laws
test(apply_de_morgan) :-
  apply_de_morgan(-( a * b ), -a + -b).

% applying invertible rules to conclusion
test(apply_to_conclusion) :-
  apply_to_conclusion([[[],[ -a + (a * a) ]]], [[[],[ -a, (a * a)
      ]]]).

% cutting the occurrences of metavariable b
test(cut) :-
  cut(
    [[[[], [-b, -b, 'Y'+2, 'Y'+1]]], [[[], ['Y'+2, 'Y'+2, -b,
       'Y'+1]]], [[[], ['Y'+3, b]]]],
    [[[[], ['Y'+3, 'Y'+3, 'Y'+2, 'Y'+1]]], [[[], ['Y'+2, 'Y'+2,
       'Y'+3, 'Y'+1]]]],
    0
  ).

% negative test case for acyclicity condition
test(is_cyclic_on_mv, fail) :-
  is_cyclic_on_mv([[[[], [a, 'Y'+4]]], [[[], [-a, 'Y'+4]]], [[[],[-a,
      'Y'+1]]], [[[],[a, 'Y'+1]]] ], a).
```

We have created one test file per module of InvAxiomCalc for testing the predicates that they define. The following Listing gives some unit tests taken from different modules. Note that we can also write negative unit tests by providing the `fail` option as a second parameter on the `test` predicate. For more information about PlUnit, please consult the Prolog Unit Tests official documentation [Wie].

4.7 Related tools for Logic Engineering

The idea of using computer supported tools (similar to TINC) for investigating logics has been well explored during the last decades. For instance, MUltlog [BFSZ96] is a system that generates a sequent calculus, a natural deduction system and clause formulation rules for a given finitely-valued first order logic. MetTel2 [TSK12] is a tool written in Java, for generating tableau provers from a given syntax and tableau calculus of a logical theory. SELLF [NPR16] is a tool that takes the specification of a proof system and checks whether it admits cut-elimination and if it is complete. The main goal of all such systems, is to make theoretical results more accessible to the researchers and practitioners.

CHAPTER 5

Summary

This Chapter contains a summary of the contents of the thesis. Section 5.1 gives the language and the hypersequent system of MALL. Section 5.2 contains an overview of the systematic procedure for transforming axioms in the language of MALL into analytic (hyper)structural rules. Section 5.3 summarizes InvAxiomCalc and the implementation details.

5.1 The language and hypersequent system of MALL

We started the thesis by giving an overview on the syntax and semantics of classical Linear Logic with focus on the Multiplicative Additive Linear Logic (MALL), i.e. the fragment of classical linear logic without exponentials. The formulas on this language are generated by the grammar:

$$\mathcal{F} ::= \mathcal{V} \mid \mathcal{V}^{\perp} \mid \perp \mid \top \mid 1 \mid 0 \mid \mathcal{F} \parr \mathcal{F} \mid \mathcal{F} \,\&\, \mathcal{F} \mid \mathcal{F} \otimes \mathcal{F} \mid \mathcal{F} \oplus \mathcal{F} \mid \mathcal{F} \multimap \mathcal{F} \tag{5.1}$$

where $\{\&(\text{with}), \parr(\text{par}), \otimes(\text{times}), \oplus(\text{plus}), \multimap \text{ (linear implication)}\}$ are the logical connectives of the language, $\{\perp, \top, 1, 0\}$ are language constants, and $\{\mathcal{V}, \mathcal{V}^{\perp}\}$ are propositional variables and their duals. The linear implication $A \multimap B$ can be expressed as $A^{\perp} \parr B$. An important property in MALL is that the linear negation $^{\perp}$ (nil) is involutive i.e. $A^{\perp\perp} \Leftrightarrow A$. This allows for applying De Morgan-like laws over formulas in MALL as follows:

$$\begin{array}{lcl} (A \,\&\, B)^{\perp} & \Leftrightarrow & A^{\perp} \oplus B^{\perp} \\ (A \parr B)^{\perp} & \Leftrightarrow & A^{\perp} \otimes B^{\perp} \end{array}$$

Additionally, we described the basic notions of Gentzen style proof theory for sequent and hypersequent calculus. As MALL is the subject of this thesis, we stopped at Hypersequent calculus for Multiplicative Additive Linear Logic (HMALL), Table 5.1.

$$ax\ \frac{}{\mathcal{H}|\vdash A, A^{\perp}} \qquad cut\ \frac{\mathcal{H}|\vdash \Gamma, A \qquad \mathcal{H}|\vdash A^{\perp}, \Delta}{\mathcal{H}|\vdash \Gamma, \Delta} \qquad ew\ \frac{\mathcal{H}}{\mathcal{H}|\vdash \Gamma}$$

$$ec\ \frac{\mathcal{H}|\vdash \Gamma|\vdash \Gamma}{\mathcal{H}|\vdash \Gamma} \qquad \otimes\ \frac{\mathcal{H}|\vdash \Gamma, A \qquad \mathcal{H}|\vdash B, \Delta}{\mathcal{H}|\vdash \Gamma, A \otimes B, \Delta} \qquad 1\ \frac{}{\mathcal{H}|\vdash 1}$$

$$\oplus_1\ \frac{\mathcal{H}|\vdash \Gamma, A}{\mathcal{H}|\vdash \Gamma, A \oplus B} \qquad \oplus_2\ \frac{\mathcal{H}|\vdash \Gamma, B}{\mathcal{H}|\vdash \Gamma, A \oplus B} \qquad \top\ \frac{}{\mathcal{H}|\vdash \Gamma, \top}$$

$$\parr\ \frac{\mathcal{H}|\vdash \Gamma, A, B}{\mathcal{H}|\vdash \Gamma, A\ \parr\ B} \qquad \&\ \frac{\mathcal{H}|\vdash \Gamma, A \qquad \mathcal{H}|\vdash \Gamma, B}{\mathcal{H}|\vdash \Gamma, A\ \&\ B} \qquad \perp\ \frac{\mathcal{H}|\vdash \Gamma}{\mathcal{H}|\vdash \Gamma, \perp}$$

Table 5.1: Hypersequent system HMALL

5.2 The systematic procedure

The goal of the thesis was to extend the system Tools for the Investigation of Nonclassical Logics (TINC) with a new tool InvAxiomCalc that implements the systematic procedure introduced in [CST09] for automatic generation of axiomatic extensions of HMALL. The procedure is based on a substructural hierarchy which is a classification of Hilbert axioms based on the properties of the logical. The hierarchy consists of $\mathcal{P}_n$ and $\mathcal{N}_n$ classes where $n \geq 0$. For the axioms in the language of MALL, the substructural hierarchy is defined as below:

$$\begin{aligned} \mathcal{P}_0 &::= \mathcal{A} \qquad \mathcal{P}_{n+1} ::= \mathcal{N}_n \mid \mathcal{P}_{n+1} \otimes \mathcal{P}_{n+1} \mid \mathcal{P}_{n+1} \oplus \mathcal{P}_{n+1} \mid 1 \mid 0 \\ \mathcal{N}_0 &::= \mathcal{A} \qquad \mathcal{N}_{n+1} ::= \mathcal{P}_n \mid \mathcal{N}_{n+1}\ \&\ \mathcal{N}_{n+1} \mid \mathcal{N}_{n+1}\ \parr\ \mathcal{N}_{n+1} \mid \top \mid \perp \end{aligned} \tag{5.2}$$

where $\mathcal{A}$ denotes the set of single propositional variables. Note that $\mathcal{P}_0 = \mathcal{N}_0 = \mathcal{A}$, $\mathcal{P}_n \subset \mathcal{N}_{n+1}$, and $\mathcal{N}_n \subset \mathcal{P}_{n+1}$. Of particular interest is also the class $\mathcal{P}_3' \subset \mathcal{P}_3$ defined as follows:

$$\mathcal{P}_3' ::= \mathcal{N}_2\ \&\ 1 \mid \mathcal{P}_3' \otimes \mathcal{P}_3' \mid \mathcal{P}_3' \oplus \mathcal{P}_3' \mid 1 \mid 0 \tag{5.3}$$

The systematic procedure accepts axioms up to the classes $\mathcal{N}_2$, $\mathcal{P}_2$, $\mathcal{P}_3'$ and generates an equivalent set of analytic inference rules i.e. rules that enjoy the cut-elimination property. Altogether, the systematic procedure consists of the following steps:

1. Finds the class in the substructural hierarchy where the given axiom belongs to. If the class is beyond $\mathcal{N}_2$, $\mathcal{P}_2$ and $\mathcal{P}_3'$ the procedure terminates immediately.

2. Either of the following:

Logical connective		Prolog representation
$\parr$	$\longrightarrow$	+
&	$\longrightarrow$	&
$\otimes$	$\longrightarrow$	*
$\oplus$	$\longrightarrow$	v

Table 5.2: The representation of logical connectives of MALL in Prolog.

- If the class falls within $\mathcal{N}_2$, transforms the axiom into a set of structural rules.
- If the class falls within $\mathcal{P}_2$ or $\mathcal{P}_3'$, transforms the axiom into a set of hyper-structural rules.

The transformation to (hyper)structural rules is performed based on the Ackerman Lemma 3.2.1, Lemma 3.3.1, and applying the reverse of invertible rules $\parr$ and &, see Table 5.1.

3. Verifies the acyclicity criteria of the generated (hyper)structural rules according to Definition 3.4.1.

4. Transforms the acyclic (hyper)structural rules into analytic rules. According to Theorem 3.4.4 every acyclic rule generated from step 2 can be converted into a Completed Rule (see Definition 3.4.3). The proof of Theorem 3.4.4 serves also as a general procedure of the transformation.

5.3 The InvAxiomCalc tool

InvAxiomCalc is build based on the steps in Section 5.2. Similar to the rest of the tools in TINC, InvAxiomCalc is implemented in Prolog. We defined new Prolog operators to represent the logical connectives of MALL as in the Table 5.2.

We make use of List data structures in Prolog for representing sequents, hypersequents, and rules. A sequent in Prolog is represented as a list of two elements where the first is a flat List representing the antecedent and the second is also a flat List representing the succedent of the sequent. Naturally, a hypersequent is represented as a List of sequents. A hyperstructural rule in Prolog is represented as a list of two elements: the first is a List of hypersequents representing the premises of the rule, and the second is a single hypersequent representing the rule conclusion. See Section 4.3 for a detailed description of the data structure we use for representing sequents, hypersequents and rules.

InvAxiomCalc consists of the following main modules:

- `invac_hierarchy` - this module is responsible for deriving the class in the substructural hierarchy where the given axiom belongs to. The input axiom is

decomposed into subformulas until ending up with atomic propositional variables. Afterwards, the class where the axiom belongs to is derived based on the recursive Definitions 5.2 and 5.3.

- `invac_hyperstructural` - generates a set of (hyper)structural rules that are equivalent with the given axiom. A new rule without premises and the input axiom in the succedent of the conclusion is introduced. The transformation consist of three steps: 1) applies the reversed ⅋ rule (see Table 5.1) in the conclusion of the introduced rule until all most binding ⅋ connectives are eliminated. 2) applies the Lemma 3.2.1 or Lemma 3.3.1 to introduce a new premise for each element of the conclusion. 3) applies the reversed ⅋ and & rules to the introduced premises until all logical connectives are eliminated.

- `invac_acyclicity` - computes the *cut-closure* (see Definition 3.4.1) of premises in the generated rule and verifies if it is acyclic w.r.t. all propositional variables.

- `invac_analytic` - converts the acyclic (hyper)structural rules into Completed Rules (see Definition 3.4.3) by applying all combinations of the cut rule.

- `invac_grammar` - provides Prolog DCG grammars for verifying the syntax of the input axiom and converting Prolog representations of axioms, sequents, hypersequents, and rules into LaTeX.

- `invac_axiom2rule` - is the parent module which puts together the functionalities of all modules.

A web interface for accessing InvAxiomCalc is available at logic.at/tinc/webinvaxiomcalc. The results are presented to the user through an automatically generated paper. Some examples of generated papers are shown in Appendix A, Appendix B and Appendix C. InvAxiomCalc is also available as a command line tool, in which case the results are not only printed into the terminal but the LaTeX sources to create a paper are also generated.

APPENDIX A

Output of InvAxiomCalc on Nelson Axiom

An analytic calculus for **MALL** extended with the axiom $((A \otimes A \otimes B) \oplus (B \otimes B \otimes A) \parr A^{\perp}) \parr B^{\perp}$

InvAxiomCalc*

May 9, 2019

Abstract

This paper defines an analytic sequent calculus for **MALL** extended with the (Hilbert) axiom $((A \otimes A \otimes B) \oplus (B \otimes B \otimes A) \parr A^{\perp}) \parr B^{\perp}$. The calculus is generated by the PROLOG-program *InvAxiomCalc*, which implements the procedure in [2].

1 Introduction

Non-classical logics are often defined by adding Hilbert axioms to known systems. The usefulness of these logics, however, heavily depends on the availability of calculi which admit cut-elimination and the subformula property (i.e., analytic calculi).

These calculi, in which the proof search proceeds by stepwise decomposition of the formulas to be proved, are indeed a prerequisite for the development of automated reasoning methods, and also key to establish essential properties of the formalized logics.

We introduce an analytic calculus for the logic obtained by extending Multiplicative Additive Linear Logic (**MALL**) with the axiom $((A \otimes A \otimes B) \oplus (B \otimes B \otimes A) \parr A^{\perp}) \parr B^{\perp}$. The calculus is generated via a PROLOG-implementation of the procedure in [2].

2 Preliminaries

The formulas of **MALL** are built over the language of classical linear logic without exponentials [3]. The language consists of propositional variables $\mathcal{V} = \{a, b, c, \ldots\}$, their duals $\mathcal{V}^{\perp} = \{a^{\perp}, b^{\perp}, c^{\perp}, \ldots\}$, the constants $\{\bot, \top, 1, 0\}$ and the logical connectives $\{\&, \parr, \otimes, \oplus\}$.

The base calculus we deal with is the sequent system for **MALL** (see Table 1), which we will simply call **MALL**. Metavariables $A, B, C, \ldots$ denote formulas, and $\Gamma, \Delta, \ldots$ denote finite multisets of formulas. We only consider single-sided multi-conclusion sequents i.e., sequents whose left-hand side (LHS)

*http://www.logic.at/tinc

$$ax\ \frac{}{\vdash A, A^\perp} \qquad cut\ \frac{\vdash \Gamma, A \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta} \qquad 1\ \frac{}{\vdash 1}$$

$$\oplus_1\ \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B} \qquad \oplus_2\ \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B} \qquad \bot\ \frac{\vdash \Gamma}{\vdash \Gamma, \bot}$$

$$\top\ \frac{}{\vdash \Gamma, \top} \qquad \otimes\ \frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}$$

$$\parr\ \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A\ \parr B} \qquad \&\ \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A\ \&\ B}$$

Table 1: The sequent calculus **MALL**

is empty and right-hand side (RHS) consists of a (possibly empty) set of formulas. In inference rules we will refer to $\Gamma, \Delta, \Sigma, \ldots$ as *multiset variables* and $A, B, C, \ldots$ as *formula variables*.

The notion of a proof in **MALL** is defined as usual. Let R be a set of rules. If there is a proof in **MALL** extended with R (**MALL** $+R$, for short) of a sequent S_0 from a set of sequents $\mathcal{S}$, we say that S_0 is derivable from $\mathcal{S}$ in **MALL**$+R$ and write $\mathcal{S} \vdash_{\mathbf{MALL}+R} S_0$. We write $\vdash_{\mathbf{MALL}+R} \alpha$ if $\emptyset \vdash_{\mathbf{MALL}+R} \alpha$.

Definition 1 (Equivalent Rules) *Given two sets of inference rules R_1 and R_2, we say that R_1 and R_2 are equivalent in MALL iff MALL$+R_1$ and MALL$+R_2$ prove the same set of sequents.*

If $R = \{r\}$ is a singleton, we write MALL$+r$ instead of MALL$+R$. An axiom φ is a rule without premises. Thus, the above definition applies also to (sets of) axioms.

Definition 2 (Structural Rules) *A sequent structural rule r is a rule of the form:*

$$\frac{\vdash \Psi_1 \qquad \ldots \qquad \vdash \Psi_n}{\vdash \Phi} \tag{1}$$

where each Ψ_i and Φ contains only multiset variables and formula variables.

Definition 3 (Acyclic Rules) *The cut-closure $CUT(r)$ of a structural rule r is the minimal set which contains the premises of r and is closed under the application of the cut rule. A rule r is said to be cyclic if for some formula variable A, we have $\vdash \Gamma, A, A^\perp \in CUT(r)$. Otherwise, r is acyclic.*

2.1 Substructural Hierarchy

The substructural hierarchy is a syntactic classification of Hilbert axioms. It has been introduced in [1] for formulas of Full Lambek calculus with exchange (**FLe**). In [2], the classification was adapted for formulas of **MALL** as follows:

Definition 4 (Substructural Hierarchy) [2] *Let $\mathcal{A}$ be a set of atomic formulas. For $n \geq 0$, the sets $\mathcal{P}_n$,$\mathcal{N}_n$ of formulas are defined via the following grammar:*

$$\begin{aligned} \mathcal{P}_0 &::= \mathcal{A} & \mathcal{P}_{n+1} &::= \mathcal{N}_n \mid \mathcal{P}_{n+1} \otimes \mathcal{P}_{n+1} \mid \mathcal{P}_{n+1} \oplus \mathcal{P}_{n+1} \mid 1 \mid 0 \\ \mathcal{N}_0 &::= \mathcal{A} & \mathcal{N}_{n+1} &::= \mathcal{P}_n \mid \mathcal{N}_{n+1}\ \&\ \mathcal{N}_{n+1} \mid \mathcal{N}_{n+1}\ \parr\ \mathcal{N}_{n+1} \mid \top \mid \bot \end{aligned} \tag{2}$$

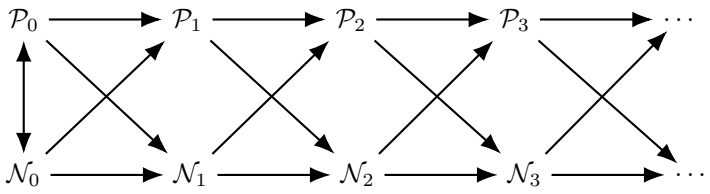

Figure 1: The substructural hierarchy [2]

A graphical representation of the substructural hierarchy is depicted in Figure 1. Note that the arrows $\rightarrow$ stand for inclusions $\subseteq$ of the classes.

3 From axioms to analytic rules

The axiom $\phi = ((A \otimes A \otimes B) \oplus (B \otimes B \otimes A) \parr A^{\perp}) \parr B^{\perp}$ is within the class $\mathcal{N}_2$ of the substructural hierarchy [2]. The transformation of ϕ into equivalent sequent rule is performed in two steps:

1) The algorithm in [2, Section 4] is used to transform ϕ into the following equivalent structural rule:

$$\frac{\vdash B^{\perp}, B^{\perp}, A^{\perp}, \Gamma \qquad \vdash \Delta, A \\ \vdash A^{\perp}, A^{\perp}, B^{\perp}, \Gamma \qquad \vdash \Sigma, B}{\vdash \Gamma, \Delta, \Sigma} \tag{3}$$

Note that the acyclicity condition is satisfied by the structural rule (3), which is the prerequisite for the second step.

2) The Rule Completion algorithm from [2, Section 6] is used to transform (3) into the following analytic sequent rule:

$$\frac{\vdash \Sigma, \Sigma, \Delta, \Gamma \qquad \vdash \Delta, \Delta, \Sigma, \Gamma}{\vdash \Gamma, \Delta, \Sigma} \tag{4}$$

Theorem 5 (Soundness and Completeness.) *The axiom* $((A \otimes A \otimes B) \oplus (B \otimes B \otimes A) \parr A^{\perp}) \parr B^{\perp}$ *is equivalent to the newly generated rule (4).*

Proof. See [2].

Theorem 6 (Cut-Elimination.) *MALL extended with the newly generated rule (4) admits cut elimination.*

Proof. See [2], which contains a general, syntactic cut-elimination procedure.

Corollary 7 *MALL extended with the rule (4) admits the subformula property.*

References

[1] A. Ciabattoni, N. Galatos, and K. Terui. From axioms to analytic rules in nonclassical logics. In *IEEE Symposium on Logic in Computer Science (LICS 08)*, pages 229–240, 2008.

[2] A. Ciabattoni, L. Straßburger, and K. Terui. Expanding the realm of systematic proof theory. In *Proceedings of Computer Science Logic (CSL 09), LNCS*, pages 163–178, 2009.

[3] J. -Y. Girard. Linear Logic In *Theoretical Computer Science*, Vol. 50, pages 1–101, 1987.

APPENDIX B

Output of InvAxiomCalc on Linearity Axiom

An analytic calculus for **HMALL** extended with the axiom $((A^{\perp} \parr B)\&1) \oplus ((B^{\perp} \parr A)\&1)$

InvAxiomCalc*

May 9, 2019

Abstract

This paper defines an analytic hypersequent calculus for **HMALL** extended with the (Hilbert) axiom $((A^{\perp} \parr B)\&1) \oplus ((B^{\perp} \parr A)\&1)$. The calculus is generated by the PROLOG-program *InvAxiomCalc*, which implements the procedure in [2].

1 Introduction

Non-classical logics are often defined by adding Hilbert axioms to known systems. The usefulness of these logics, however, heavily depends on the availability of calculi which admit cut-elimination and the subformula property (i.e., analytic calculi).

These calculi, in which the proof search proceeds by stepwise decomposition of the formulas to be proved, are indeed a prerequisite for the development of automated reasoning methods, and also key to establish essential properties of the formalized logics.

We introduce an analytic calculus for the logic obtained by extending Multiplicative Additive Linear Logic (**HMALL**) with the axiom $((A^{\perp} \parr B)\&1) \oplus ((B^{\perp} \parr A)\&1)$. The calculus is generated via a PROLOG-implementation of the procedure in [2].

2 Preliminaries

The formulas of **HMALL** are built over the language of classical linear logic without exponentials [3]. The language consists of propositional variables $\mathcal{V} = \{a, b, c, \ldots\}$, their duals $\mathcal{V}^{\perp} = \{a^{\perp}, b^{\perp}, c^{\perp}, \ldots\}$, the constants $\{\perp, \top, 1, 0\}$ and the logical connectives $\{\&, \parr, \otimes, \oplus\}$.

The base calculus we deal with is the hypersequent system for **HMALL** (see Table 1), which we will simply call **HMALL**. Metavariables $A, B, C, \ldots$ denote formulas, and $\Gamma, \Delta, \ldots$ denote finite multisets of formulas. We only consider single-sided multi-conclusion sequents i.e., sequents whose left-hand side (LHS) is empty and right-hand side (RHS) consists of a (possibly empty) set of formulas. In inference rules we will refer to $\Gamma, \Delta, \Sigma, \ldots$ as *multiset variables* and $A, B, C, \ldots$ as *formula variables*.

*http://www.logic.at/tinc

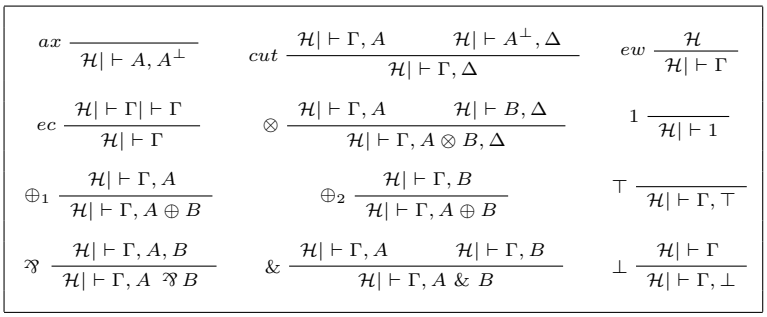

Table 1: The hypersequent calculus **HMALL**

Definition 1 (Hypersequents) *A* hypersequent $\mathcal{H}$ *is a multiset* $S_1 \mid \cdots \mid S_n$ *where each* S_i *for* $i = 1, \ldots, n$ *is a sequent, called a* component *of the hypersequent.*

The symbol " | " is intended to denote disjunction at the meta-level. The notion of proof in **HMALL** is defined as usual. Let R be a set of rules. If there is a proof in **HMALL** extended with R (**HMALL**$+R$, for short) of a sequent S_0 from a set of sequents $\mathcal{S}$, we say that S_0 is derivable from $\mathcal{S}$ in **HMALL**$+R$ and write $\mathcal{S} \vdash_{\mathbf{HMALL}+R} S_0$. We write $\vdash_{\mathbf{HMALL}+R} \alpha$ if $\emptyset \vdash_{\mathbf{HMALL}+R} \Rightarrow \alpha$.

Definition 2 (Equivalent Rules) *Given two sets of inference rules* R_1 *and* R_2*, we say that* R_1 *and* R_2 *are equivalent in HMALL iff HMALL*$+R_1$ *and HMALL*$+R_2$ *prove the same set of sequents.*

If $R = \{r\}$ is a singleton, we write HMALL$+r$ instead of HMALL$+R$. An axiom φ is a rule without premises. Thus, the above definition applies also to (sets of) axioms.

Definition 3 (Hyperstructural Rules) *A hyperstructural rule* r *is a rule of the form:*

$$\frac{\mathcal{H}| \vdash \Psi_1 \qquad \ldots \qquad \mathcal{H}| \vdash \Psi_n}{\mathcal{H}| \vdash \Phi_1 | \ldots | \vdash \Phi_m} \tag{1}$$

where each Φ_i *and* Ψ_j *contains only multiset variables and formula variables.*

Definition 4 (Acyclic Rules) *The cut-closure* $CUT(r)$ *of a hyperstructural rule* r *is the minimal set which contains the premises of* r *and is closed under the application of the cut rule. A rule* r *is said to be cyclic if for some formula variable* A*, we have* $\mathcal{H}| \vdash \Gamma, A, A^{\perp} \in CUT(r)$*. Otherwise,* r *is acyclic.*

2.1 Substructural Hierarchy

The substructural hierarchy is a syntactic classification of Hilbert axioms. It has been introduced in [1] for formulas of Full Lambek calculus with exchange (**FLe**). In [2], the classification was adapted for formulas of **HMALL** as follows:

Definition 5 (Substructural Hierarchy) [2] *Let* $\mathcal{A}$ *be a set of atomic formulas. For* $n \geq 0$*, the sets* $\mathcal{P}_n, \mathcal{N}_n$ *of formulas are defined via the following grammar:*

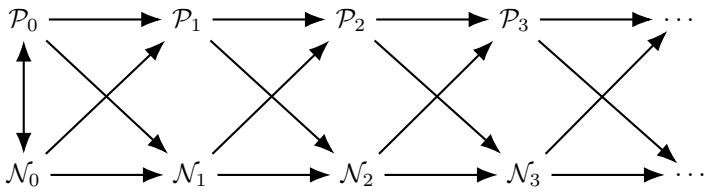

Figure 1: The substructural hierarchy [2]

$$\begin{array}{ll} \mathcal{P}_0 ::= \mathcal{A} & \mathcal{P}_{n+1} ::= \mathcal{N}_n \mid \mathcal{P}_{n+1} \otimes \mathcal{P}_{n+1} \mid \mathcal{P}_{n+1} \oplus \mathcal{P}_{n+1} \mid 1 \mid 0 \\ \mathcal{N}_0 ::= \mathcal{A} & \mathcal{N}_{n+1} ::= \mathcal{P}_n \mid \mathcal{N}_{n+1} \,\&\, \mathcal{N}_{n+1} \mid \mathcal{N}_{n+1} \parr \mathcal{N}_{n+1} \mid \top \mid \bot \end{array} \tag{2}$$

A graphical representation of the substructural hierarchy is depicted in Figure 1. Note that the arrows $\rightarrow$ stand for inclusions $\subseteq$ of the classes.

3 From axioms to analytic rules

The axiom $\phi = ((A^\perp \parr B)\&1) \oplus ((B^\perp \parr A)\&1)$ is within the class $\mathcal{P}_3'$ of the substructural hierarchy [2]. The transformation of ϕ into equivalent hypersequent rule is performed in two steps:

1) The algorithm in [2, Section 5] is used to transform ϕ into the following equivalent hyperstructural rule:

$$\frac{\mathcal{H} \mid \vdash \Sigma, A \qquad \mathcal{H} \mid \vdash \Gamma, B \qquad \mathcal{H} \mid \vdash \Theta, B^\perp \qquad \mathcal{H} \mid \vdash \Delta, A^\perp}{\mathcal{H} \mid \vdash \Sigma, \Theta \mid \vdash \Gamma, \Delta} \tag{3}$$

Note that the acyclicity condition is satisfied by the hyperstructural rule (3), which is the prerequisite for the second step.

2) The Rule Completion algorithm from [2, Section 6] is used to transform (3) into the following analytic hypersequent rule:

$$\frac{\mathcal{H} \mid \vdash \Delta, \Sigma \qquad \mathcal{H} \mid \vdash \Theta, \Gamma}{\mathcal{H} \mid \vdash \Sigma, \Theta \mid \vdash \Gamma, \Delta} \tag{4}$$

Theorem 6 (Soundness and Completeness.) *The axiom* $((A^\perp \parr B)\&1) \oplus ((B^\perp \parr A)\&1)$ *is equivalent to the newly generated rule (4).*

Proof. See [2].

Theorem 7 (Cut-Elimination.) *HMALL extended with the newly generated rule (4) admits cut elimination.*

Proof. See [2], which contains a general, syntactic cut-elimination procedure.

Corollary 8 *HMALL extended with the rule (4) admits the subformula property.*

References

[1] A. Ciabattoni, N. Galatos, and K. Terui. From axioms to analytic rules in nonclassical logics. In *IEEE Symposium on Logic in Computer Science (LICS 08)*, pages 229–240, 2008.

[2] A. Ciabattoni, L. Straßburger, and K. Terui. Expanding the realm of systematic proof theory. In *Proceedings of Computer Science Logic (CSL 09), LNCS*, pages 163–178, 2009.

[3] J. -Y. Girard. Linear Logic In *Theoretical Computer Science*, Vol. 50, pages 1–101, 1987.

APPENDIX C

Output of InvAxiomCalc on Cancel Axiom

An analytic calculus for **MALL** extended with the axiom $(A \parr A^{\perp})^{\perp}$

InvAxiomCalc*

May 9, 2019

Abstract

This paper defines a sequent calculus for **MALL** extended with the (Hilbert) axiom $(A \parr A^{\perp})^{\perp}$. Furthermore, in presence of weakening w, an *analytic* calculus is defined for **MALL**+w extended with $(A \parr A^{\perp})^{\perp}$. The calculus is generated by the PROLOG-program *InvAxiomCalc*, which implements the procedure in [2].

1 Introduction

Non-classical logics are often defined by adding Hilbert axioms to known systems. The usefulness of these logics, however, heavily depends on the availability of calculi which admit cut-elimination and the subformula property (i.e., analytic calculi).

These calculi, in which the proof search proceeds by stepwise decomposition of the formulas to be proved, are indeed a prerequisite for the development of automated reasoning methods, and also key to establish essential properties of the formalized logics.

We introduce a calculus for the logic obtained by extending Multiplicative Additive Linear Logic (**MALL**) with the axiom $(A \parr A^{\perp})^{\perp}$. The calculus is obtained by extending the sequent system for **MALL** with a structural rule equivalent to the axiom. In presence of weakening w we also define an *analytic* calculus for MALL+w extended with $(A \parr A^{\perp})^{\perp}$. The calculus is generated via a PROLOG-implementation of the procedure in [2].

2 Preliminaries

The formulas of **MALL** are built over the language of classical linear logic without exponentials [3]. The language consists of propositional variables $\mathcal{V} = \{a, b, c, \ldots\}$, their duals $\mathcal{V}^{\perp} = \{a^{\perp}, b^{\perp}, c^{\perp}, \ldots\}$, the constants $\{\perp, \top, 1, 0\}$ and the logical connectives $\{\&, \parr, \otimes, \oplus\}$.

The base calculus we deal with is the sequent system for **MALL** (see Table 1), which we will simply call **MALL**. Metavariables $A, B, C, \ldots$ denote formulas, and $\Gamma, \Delta, \ldots$ denote finite multisets of formulas. We only consider

*http://www.logic.at/tinc

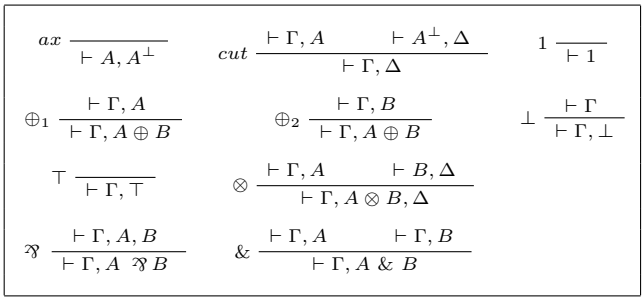

Table 1: The sequent calculus **MALL**

single-sided multi-conclusion sequents i.e., sequents whose left-hand side (LHS) is empty and right-hand side (RHS) consists of a (possibly empty) set of formulas. In inference rules we will refer to $\Gamma, \Delta, \Sigma, \ldots$ as *multiset variables* and $A, B, C, \ldots$ as *formula variables*.

The notion of a proof in **MALL** is defined as usual. Let R be a set of rules. If there is a proof in **MALL** extended with R (**MALL** $+R$, for short) of a sequent S_0 from a set of sequents $\mathcal{S}$, we say that S_0 is derivable from $\mathcal{S}$ in **MALL**$+R$ and write $\mathcal{S} \vdash_{\mathbf{MALL}+R} S_0$. We write $\vdash_{\mathbf{MALL}+R} \alpha$ if $\emptyset \vdash_{\mathbf{MALL}+R} \alpha$.

Definition 1 (Equivalent Rules) *Given two sets of inference rules R_1 and R_2, we say that R_1 and R_2 are equivalent in MALL iff MALL$+R_1$ and MALL$+R_2$ prove the same set of sequents.*

If $R = \{r\}$ is a singleton, we write MALL$+r$ instead of MALL$+R$. An axiom φ is a rule without premises. Thus, the above definition applies also to (sets of) axioms.

Definition 2 (Structural Rules) *A sequent structural rule r is a rule of the form:*

$$\frac{\vdash \Psi_1 \qquad \ldots \qquad \vdash \Psi_n}{\vdash \Phi} \tag{1}$$

where each Ψ_i and Φ contains only multiset variables and formula variables.

Definition 3 (Acyclic Rules) *The cut-closure $CUT(r)$ of a structural rule r is the minimal set which contains the premises of r and is closed under the application of the cut rule. A rule r is said to be cyclic if for some formula variable A, we have $\vdash \Gamma, A, A^{\perp} \in CUT(r)$. Otherwise, r is acyclic.*

2.1 Substructural Hierarchy

The substructural hierarchy is a syntactic classification of Hilbert axioms. It has been introduced in [1] for formulas of Full Lambek calculus with exchange (**FLe**). In [2], the classification was adapted for formulas of **MALL** as follows:

Definition 4 (Substructural Hierarchy) [2] *Let $\mathcal{A}$ be a set of atomic formulas. For $n \geq 0$, the sets $\mathcal{P}_n$,$\mathcal{N}_n$ of formulas are defined via the following grammar:*

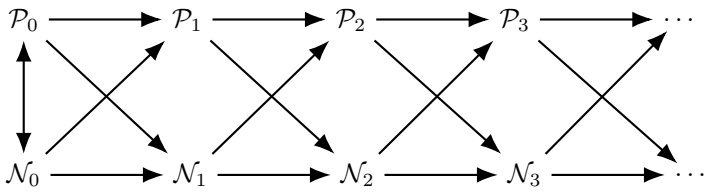

Figure 1: The substructural hierarchy [2]

$$\begin{array}{ll} \mathcal{P}_0 ::= \mathcal{A} & \mathcal{P}_{n+1} ::= \mathcal{N}_n \mid \mathcal{P}_{n+1} \otimes \mathcal{P}_{n+1} \mid \mathcal{P}_{n+1} \oplus \mathcal{P}_{n+1} \mid 1 \mid 0 \\ \mathcal{N}_0 ::= \mathcal{A} & \mathcal{N}_{n+1} ::= \mathcal{P}_n \mid \mathcal{N}_{n+1} \,\&\, \mathcal{N}_{n+1} \mid \mathcal{N}_{n+1} \parr \mathcal{N}_{n+1} \mid \top \mid \bot \end{array} \tag{2}$$

A graphical representation of the substructural hierarchy is depicted in Figure 1. Note that the arrows $\rightarrow$ stand for inclusions $\subseteq$ of the classes.

3 From axioms to analytic rules

The axiom $\phi = (A \parr A^\perp)^\perp$ is within the class $\mathcal{N}_2$ of the substructural hierarchy [2]. The transformation of ϕ into equivalent sequent rule is performed in two steps:

1) The algorithm in [2, Section 4] is used to transform ϕ into the following equivalent structural rule:

$$\frac{\vdash A, A^\perp, \Gamma}{\vdash \Gamma} \tag{3}$$

Note that the acyclicity condition is *not* satisfied by the structural rule (3). Therefore, the second step cannot be performed without the presence of the weakening rule w:

$$w\,\frac{\vdash \Gamma}{\vdash \Gamma, \Delta} \tag{4}$$

2) The Rule Completion algorithm from [2, Section 6] is used to transform (3) into the following analytic sequent rule:

$$\frac{}{\vdash \Gamma} \tag{5}$$

In presence of weakening this leads to an analytic rule equivalent to the axiom. Hence we have:

Theorem 5 (Soundness and Completeness.) *The axiom* $(A \parr A^\perp)^\perp$*:*

- *is equivalent in MALL to the rule (3).*
- *is equivalent in MALL+w to the rule (5).*

Proof. See [2].

Theorem 6 (Cut-Elimination.)

- *MALL extended with the newly generated rule (3) admits cut elimination.*
- *MALL+w extended with the newly generated rule (5) admits cut elimination and the subformula property.*

Proof. See [2], which contains a general, syntactic cut-elimination procedure.

References

[1] A. Ciabattoni, N. Galatos, and K. Terui. From axioms to analytic rules in nonclassical logics. In *IEEE Symposium on Logic in Computer Science (LICS 08)*, pages 229–240, 2008.

[2] A. Ciabattoni, L. Straßburger, and K. Terui. Expanding the realm of systematic proof theory. In *Proceedings of Computer Science Logic (CSL 09), LNCS*, pages 163–178, 2009.

[3] J. -Y. Girard. Linear Logic In *Theoretical Computer Science*, Vol. 50, pages 1–101, 1987.

List of Figures

List of Tables

List of Algorithms

Glossary

active formula the formulas in the premises of a rule from where the principal formula is derived. 9

end sequent the last sequent occuring in a derivation. 8, 13

formula variable a variable representing a single formula or an axiom. 11

InvAxiomCalc New tool in TINC which allows for automatic generation of analytic calculi for axiomatic extensions of classical linear logic without exponentials.. xi–xiii, 2, 3, 29, 33, 47, 51, 53–56

invertible rule a rule where a proof of the conclusion, also implies a proof for each its premises. 9, 11, 55

MetTel2 a tool for generating tableau provers from specifications of a syntax and a tableau calculus for a logical theory [TSK12].. 51

multiset variable a variable representing a multi set of formulas. 11

MUltlog a system which takes as input the specification of a finitely-valued first-order logic and produces a sequent calculus, a natural deduction system, and clause formation rules for this logic [BFSZ96].. 51

PlUnit A Prolog unit-test framework which was initially developed for SWI-Prolog. 50, 51

principal formula the formula introduced by a logical rule. 9

SELLF a tool that takes the specification of a proof system and checks whether it admits cut-elimination and if it is complete [NPR16].. 51

stable truth truth values which are not affected by the application of the logical connectives. For example, if we know A, and $A \supset B$ we can derive B without affecting the truth value of A.. 6

subformula property a property of a rule indicating that every formula in the premises is a subformula in the conclusion. 1, 13

Acronyms

DCG Definite Clause Grammar. 31, 47–49, 56

Flew Full Lambek Calculus with exchange and weakening. 31

HMALL Hypersequent calculus for Multiplicative Additive Linear Logic. 10–13, 15, 23, 25, 33, 38, 47, 49, 53, 54, 77

IL Intuitionistic Logic. 5

IMALL Intuitionistic Multiplicative Additive Linear Logic. 16

LC Linear Conclusion. 24, 25, 42

MALL Multiplicative Additive Linear Logic. xi–xiii, 2, 5, 7, 10, 40, 47, 53–55, 77

NFV No Forumula Variable. 24, 25, 42

SP Subformula Property. 24, 25, 42

TINC Tools for the Investigation of Nonclassical Logics. xi, xiii, 1–3, 29, 31–33, 51, 54, 55, 75, 81

W.l.g. Without loss of generality. 44

Bibliography

[Avr87] Arnon Avron. A constructive analysis of RM. *J. Symb. Log.*, 52(4):939–951, 1987.

[BFSZ96] Matthias Baaz, Christian G. Fermüller, Gernot Salzer, and Richard Zach. Multlog 1.0: Towards an expert system for many-valued logics. In *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction, New Brunswick, NJ, USA, July 30 - August 3, 1996, Proceedings*, pages 226–230, 1996.

[Bus98] Samuel R. Buss. Chapter i - an introduction to proof theory. In Samuel R. Buss, editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 1 – 78. Elsevier, 1998.

[CGT08] Agata Ciabattoni, Nikolaos Galatos, and Kazushige Terui. From axioms to analytic rules in nonclassical logics. In *LICS*, pages 229–240. IEEE Computer Society, 2008.

[CGT12] Agata Ciabattoni, Nikolaos Galatos, and Kazushige Terui. Algebraic proof theory for substructural logics: Cut-elimination and completions. *Ann. Pure Appl. Logic*, 163(3):266–290, 2012.

[CLSZ13] Agata Ciabattoni, Ori Lahav, Lara Spendier, and Anna Zamansky. Automated support for the investigation of paraconsistent and other logics. In *LFCS*, volume 7734 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2013.

[CMS13] Agata Ciabattoni, Paolo Maffezioli, and Lara Spendier. Hypersequent and labelled calculi for intermediate logics. In *TABLEAUX*, volume 8123 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2013.

[CS14] Agata Ciabattoni and Lara Spendier. Tools for the investigation of substructural and paraconsistent logics. In *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*, pages 18–32, 2014.

[CST09] Agata Ciabattoni, Lutz Straßburger, and Kazushige Terui. Expanding the realm of systematic proof theory. In *CSL*, volume 5771 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2009.

[CZ97] Alexander V. Chagrov and Michael Zakharyaschev. *Modal Logic*, volume 35 of *Oxford logic guides*. Oxford University Press, 1997.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische schließen i und ii. *Mathematische Zeitschrift*, 39:176–210, 1935.

[Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[NPR16] Vivek Nigam, Elaine Pimentel, and Giselle Reis. An extended framework for specifying and reasoning about proof systems. *J. Log. Comput.*, 26(2):539–576, 2016.

[TSK12] Dmitry Tishkovsky, Renate A. Schmidt, and Mohammad Khodadadi. The tableau prover generator mettel2. In *JELIA*, volume 7519 of *Lecture Notes in Computer Science*, pages 492–495. Springer, 2012.

[Wie] Jan Wielemaker. Prolog unit tests.