Christian Urban
Gonville and Caius College
University of Cambridge Computer Laboratory

Classical Logic and Computation

A dissertation submitted to the University of Cambridge
towards the degree of Doctor of Philosophy.
October 2000

Abstract

This thesis contains a study of the proof theory of classical logic and addresses the problem of giving a computational interpretation to classical proofs. This interpretation aims to capture features of computation that go beyond what can be expressed in intuitionistic logic.

We introduce several strongly normalising cut-elimination procedures for classical logic. Our procedures are less restrictive than previous strongly normalising procedures, while at the same time retaining the strong normalisation property, which various standard cut-elimination procedures lack. In order to apply proof techniques from term rewriting, including symmetric reducibility candidates and recursive path ordering, we develop term annotations for sequent proofs of classical logic.

We then present a sequence-conclusion natural deduction calculus for classical logic and study the correspondence between cut-elimination and normalisation. In contrast to earlier work, which analysed this correspondence in various fragments of intuitionistic logic, we establish the correspondence in classical logic.

Finally, we study applications of cut-elimination. In particular, we analyse several classical proofs with respect to their behaviour under cut-elimination. Because our cut-elimination procedures impose fewer constraints than previous procedures, we are able to show how a fragment of classical logic can be seen as a typing system for the simply-typed lambda calculus extended with an erratic choice operator. As a pleasing consequence, we can give a simple computational interpretation to Lafont's example.

Preface

This dissertation is the result of my own work and except where otherwise stated includes nothing that is the outcome of work done in collaboration. Whenever possible long proofs are given in Appendix B, rather than in the main text. A reference, in each case, will point the reader to the section in the appendix where the details of the proofs can be found.

I am grateful to my examiners, Prof. A. M. Pitts and Prof. H. P. Barendregt, for their helpful comments and useful suggestions. Any remaining errors are of course all my own work.

The following people have requested (or been given ;o) a copy of this dissertation (as of March 2006):

Edmund Robinson	edmundr@dcs.qmw.ac.uk
Gianluigi Bellin	bellin@sci.univr.it
Pierre-Louis Curien	curien@dmi.ens.fr
Jose Espirito Santo	jes@math.uminho.pt
Paul Ruet	ruet@iml.univ-mrs.fr
Nicola Piccinini	pic@arena.sci.univr.it
Rene Vestergaard	vester@jaist.ac.jp
Jim Laird	jiml@cogs.susx.ac.uk
Marcelo Fiore	Marcelo.Fiore@cl.cam.ac.uk
Roy Dyckhoff	rd@dcs.st-and.ac.uk
Ralph Matthes	matthes@informatik.uni-muenchen.de
Felix Joachimski	joachski@rz.mathematik.uni-muenchen.de
Thomas Antonini	tototbol@club-internet.fr
Eduardo De la Hoz	edehozvi@hotmail.com
Jeremy Dawson	jeremy@discus.anu.edu.au
Lars Birkedal	birkedal@itu.dk
Edward Haeusler	hermann@inf.puc-rio.br
Stephane Lengrand	Stephane.Lengrand@ENS-Lyon.fr
Norman Danner	ndanner@wesleyan.edu
Dan Dougherty	dd@cs.wpi.edu
Zena Matilde Ariola	ariola@cs.uoregon.edu
Masahiko Sato	masahiko@kuis.kyoto-u.ac.jp
Ken Shan	ken@digitas.harvard.edu
James Brotherston	jjb@dcs.ed.ac.uk
Morten Heine Sørensen	mhs@it-practice.dk
Richard Nathan Linger	rlinger@cse.ogi.edu
Timothy Griffin	tim.griffin@intel.com
Valeria de Paiva	Valeria.dePaiva@parc.com
Stefan Hetzl	shetzl@chello.at
Steffen van Bakel	svb@doc.ic.ac.uk
Carsten Schürmann	carsten@cs.yale.edu
Jens Brage	brage@math.su.se
Dominic Hughes	dominic@theory.stanford.edu
Rajeev Gore	Rajeev.Gore@anu.edu.au
Alexander Leitsch	leitsch@logic.at
Agatha Ciabatone	agata@logic.at
Mattia Petrolo	mattia.petrolo@libero.it

Acknowledgements

First, and above all, I wish to thank Gavin Bierman for his immense help and constant encouragement during the time of my Ph.D. He not only has consistently given his time to improve my work, but has also been very patient with the slow speed of my writing. His expertise and the numerous discussions with him have been invaluable to this thesis. I should like to thank him for his generous support and belief in me.

I was privileged in having the opportunity to work with Martin Hyland. It is a great pleasure to acknowledge the enormous amount I have had learnt from the conversations with him. His enthusiasm and experience influenced this thesis in many ways.

Roy Dyckhoff has helped me much over the years. I have learnt a great deal from his immense insight into proof theory. I am indebted to him for giving me invaluable advice for my work.

I appreciated very much the conversation with Henk Barendregt about my work in a beautiful churchyard in l'Aquila. This delightful afternoon has been a refreshing source of reassurance, which made the process of writing this thesis less arduous.

I am very grateful to Laurent Regnier and Jean-Yves Girard for giving me time to finish this thesis while working in Marseille.

The work has greatly benefited from discussions with Harold Schellinx and Jean-Baptiste Joinet concerning LK^{tq}.

I thank Claudia Faggian, who took care of me in the first months of my stay in Marseille.

Thank goes also to Barbara and Werner Daniel, who taught me that science is fun and that research is a fascinating lifelong activity.

If my laptop had not been repaired in time, my brother would have lent me his most holy possession (an Apple Powerbook G3). Thank you very much for that and all the enjoyable time with you.

I am grateful beyond any measure to my parents for their unstinting support over all the years—thank you comes not even close to adequate for how much they have supported me.

I want to express my gratitude to the German Academic Exchange Service (DAAD) for generously supporting me over three years with two scholarships. I acknowledge use of Paul Taylor's diagram package and thank the authors of dvips for making their code publicly available, without which the modification that allowed me to draw the proofs in Appendix A would have been impossible.

Contents

List of Figures

Chapter 1

Introduction

A sequent calculus without cut-elimination is like a car without engine.

—J.-Y. Girard
in Linear Logic: Its Syntax and Semantics, 1995.

In this thesis we study the proof theory of classical logic and address the problem of giving a computational interpretation to classical proofs. This interpretation aims to capture features of computation that go beyond what can be expressed in intuitionistic logic. The point of departure for this thesis will be the *Curry-Howard correspondence*, which gives a computational interpretation to intuitionistic proofs.

According to the Curry-Howard correspondence, the simply-typed lambda calculus can be viewed as a term assignment for intuitionistic proofs formalised in Gentzen's natural deduction calculus NJ. What makes this term assignment exciting is that it relates not just two calculi, but also certain concepts within them, as shown below.

Gentzen's NJ-calculus		simply-typed lambda calculus
formula	$\Leftrightarrow$	type
proof	$\Leftrightarrow$	term
normalisation	$\Leftrightarrow$	reduction

Curry and Feys [1958] hinted at these correspondences, but it took quite some time before they were systematically studied by Howard [1980].

One consequence of the Curry-Howard correspondence is that we can talk of a *computational interpretation* of NJ-proofs. The simply-typed lambda calculus is a prototypical functional programming language: the terms are programs, and reduction is a form of computation, which converts a term to its simplest form, analogous to symbolic evaluation. On the other hand, normalisation is a method for eliminating certain redundancies in proofs. Applied iteratively, it transforms a proof to one in normal form. Using the Curry-Howard correspondence we see that the two notions coincide, and thus the computational interpretation of an NJ-proof is the corresponding simply-typed lambda term representing a functional program.

Because of its connection with computation, the Curry-Howard correspondence is quite useful in programming language design: often computer scientists and logicians have arrived independently at the same idea. For example, Girard [1972] and Reynolds [1974] independently studied the second-order polymorphic lambda calculus, and versions of the typing algorithm of ML were developed independently by Hindley [1969] and Milner [1978]. Another example (described by Mitchell [1996]) is the formulation of sums in "classical" ML, which was not incorrect, but relied on exceptions to ensure type security. This meant a form of runtime type check was necessary to ensure that programs could not change their type. Clearly this runtime check slowed down the execution of programs. The proper formulation introduced later, which does not require any runtime type check, corresponds exactly to the inference rules for $\vee$ in NJ.

In practice, the Curry-Howard correspondence is often employed in proof assistants with which many mathematical proofs can be fully formalised and checked. Formalised proofs however tend to be very large, and therefore one needs good bookkeeping devices to keep track of them. NuPrl and Coq are examples of contemporary proof assistants that make use of the Curry-Howard correspondence by representing proofs as terms.

Whilst the Curry-Howard correspondence was originally established for the simply-typed lambda calculus and Gentzen's NJ-calculus only, later it has been often studied in the setting of more expressive typed lambda calculi and more expressive logics. This was motivated by the fact that, although the simply-typed lambda calculus can be seen as a core of various mainstream functional programming languages, only a limited number of features of those languages (e.g. pattern matching and abstract datatypes) can be easily encoded into this core. Other features cannot. Examples are polymorphism and state. In more expressive typed lambda calculi these features are directly accessible. For instance, System F provides facilities for polymorphism [Girard et al., 1989], and linear lambda calculi offer mechanisms to manipulate state [Wadler, 1990]. Using the Curry-Howard correspondence one finds the more expressive lambda calculi give a computational interpretation to proofs in more expressive NJ-calculi, three of which are listed below.

NJ-calculi		typed lambda calculi	
first-order NJ-calculus	$\Leftrightarrow$	λP	[de Bruijn, 1980]
second-order NJ-calculus	$\Leftrightarrow$	System F	[Girard, 1972]
linear NJ-calculus	$\Leftrightarrow$	linear lambda calculus	[Benton et al., 1993]

In this thesis we shall study the Curry-Howard correspondence in the setting of classical logic. This logic is more expressive than intuitionistic logic, in the sense that every intuitionistic proof is a classical proof but not *vice versa*. Griffin [1990] was the first to note a connection between classical logic and functional programming languages with control operators. We take this observation as evidence that a computational interpretation for classical proofs can capture features of computation not directly expressible in a typed lambda calculus which corresponds to intuitionistic logic.

Usually, the Curry-Howard correspondence is formulated for natural deduction calculi, because in intuitionistic logic they have compelling advantages as deduction systems. Unfortunately, there are various technical reasons[1] which make their classical counterparts less suitable as deduction systems for classical logic. Sequent calculi seem much better suited. Therefore in this thesis we shall mainly focus on sequent calculus formulations of classical logic. The best-known sequent calculus for classical logic is the calculus LK invented by Gentzen [1935]. To illustrate some problems concerning a computational interpretation of classical logic, we shall describe this calculus briefly below.

In LK proofs consist of trees labelled by *sequents* of the form $\Gamma \vdash \Delta$, where Γ and Δ are collections of formulae. The axioms of LK are of the form $\overline{B \vdash B}$. The general form of an *inference rule* is either

$$\frac{P}{C} \qquad \text{or} \qquad \frac{P_1 \quad P_2}{C}$$

where the P's stand for sequents, called *premises*, and C stands for a sequent, called the *conclusion*, which is deduced from the premises. There are three sorts of inference rules in LK. Logical inference rules introduce a formula in the conclusion and come in two flavours—left rules and right rules. For example, the left and right rule introducing an implicational formula have the form:

$$\frac{\Gamma_1 \vdash \Delta_1, B \quad C, \Gamma_2 \vdash \Delta_2}{B \supset C, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \supset_L \qquad \frac{B, \Gamma \vdash \Delta, C}{\Gamma \vdash \Delta, B \supset C} \supset_R$$

Structural inference rules manipulate sequents, in the sense that contraction identifies occurrences of the same formula and weakening introduces surplus formulae. For example, the contraction-left and weakening-right rule have the form:

$$\frac{B, B, \Gamma \vdash \Delta}{B, \Gamma \vdash \Delta} \,\text{Contr}_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, B} \,\text{Weak}_R$$

The third sort of inference rule is the cut-rule

$$\frac{\Gamma_1 \vdash \Delta_1, B \quad B, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \,\text{Cut}$$

which enables us to join two LK-proofs together. We shall often simply refer to an application of the cut-rule as a cut.

One consequence of our use of sequent calculi in place of natural deduction calculi is that computation corresponds to *cut-elimination*—a procedure which transforms a given sequent proof containing instances of cut to one with no instances of cut (called a cut-free proof). This procedure does not eliminate all cuts from a proof immediately, rather it replaces every cut with simpler cuts, and by iteration one eventually ends up with a cut-free proof.

Zucker [1974] and Pottinger [1977] have shown that normalisation and cut-elimination are closely related in intuitionistic logic. Thus the question of the computational

[1] "One may doubt that this is the proper way of analysing classical inferences." [Prawitz, 1971, Page 244]

meaning of cut-elimination is settled in the intuitionistic case. Surprisingly, there is little attention paid in the literature to an analysis of what cut-elimination in classical logic means computationally. At the time of writing we are aware only of work by Danos et al. [1997], which considers a variant of Gentzen's sequent calculus LK as a programming language. However, we shall show that there are still many open questions concerning the relationship between cut-elimination and computation.

One of the reasons for the little attention is that, in contrast to cut-elimination in intuitionistic logic, cut-elimination in classical logic had been dismissed in the past as being not very interesting from a computational point of view. For example, Lafont argued in the influential book [Girard et al., 1989] that in classical logic a correspondence between cut-elimination and computation is unachievable, owing to an "inconsistency". He noted that the classical proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\pi_L}{\vdash B}}{\vdash B, C}\,\text{Weak}_R \qquad \dfrac{\dfrac{\pi_R}{\vdash B}}{C \vdash B}\,\text{Weak}_L}{\vdash B, B}\,\text{Cut}}{\vdash B}\,\text{Contr}_R \quad \longrightarrow \quad \dfrac{\pi_L}{\vdash B} \quad \text{or} \quad \dfrac{\pi_R}{\vdash B} \qquad (1.1)$$

reduces, as shown, non-deterministically to one of its subproofs, and based upon a categorical insight of Lambek and Scott [1988, Page 67] he concluded:

> More generally, all proofs of a given sequent $\Gamma \vdash \Delta$ are identified. So classical logic is *inconsistent*, not from a *logical* viewpoint ($\perp$ is not provable), but from an *algorithmic* one [Girard et al., 1989, Page 152].

His argument is that if one takes cut-elimination as an equality preserving operation, then in the example above all proofs of an arbitrary formula B are identified—therefore there is an "inconsistency" when viewing classical proofs as programs. Let us stress that Lafont assumes that cut-elimination is an equality preserving operation; otherwise the "inconsistency" cannot arise. Clearly, this assumption is the prevailing doctrine in the simply-typed lambda calculus where reduction does not change the (denotational) meaning of terms. Furthermore, it is a very plausible assumption in the context of natural deduction, to the extent that Kreisel [1971, Page 112] viewed it as a *minimum requirement*.

> A minimum requirement is... that *any derivation can be normalized*, that is transformed into a unique normal form by a series of steps, so-called "conversions", each of which preserve the proof described by the derivation.

In this thesis we shall propose that in the context of classical logic it is however prudent to reconsider this assumption. But before doing so, let us analyse how the "inconsistency" in Lafont's example was dealt with earlier.

According to Lafont, the problem with the "inconsistency" can be remedied, if classical logic is restricted so that the symmetric instance of the cut given in (1.1) is no longer derivable. Examples of such logics are intuitionistic logic and linear logic. In intuitionistic logic the sequents are restricted to be of the form $\Gamma \vdash B$, thus eliminating the inference

$$\frac{\vdash B}{\vdash B, C}\ \mathrm{Weak}_R$$

and in linear logic the structural rules are formulated such that

$$\dfrac{\dfrac{\vdash B}{\vdash B, ?C}\ \mathrm{Weak}_R \qquad \dfrac{\vdash B}{!C \vdash B}\ \mathrm{Weak}_L}{\vdash B, B}\ \mathrm{Cut}$$

is not a valid instance of the cut-rule. Consequently, Lafont's example cannot arise in intuitionistic logic or in linear logic.

For intuitionistic logic we have, as mentioned earlier, the Curry-Howard correspondence with the simply-typed lambda calculus. For linear logic the situation is unclear. At its inception it was thought to have direct connection with concurrency, and some connections have been uncovered by Abramsky [1993]. However, it is still fair to say that in linear logic the question of what kind of computation the process of cut-elimination corresponds is largely unanswered.

Recently, a number of other solutions for the "inconsistency" in Lafont's example have been proposed. Instead of restricting classical logic so that Proof (1.1) is not derivable, they restrict the reduction rules of classical logic. For example in Parigot's $\lambda\mu$-calculus, although formulated as a natural deduction calculus, Lafont's proof is derivable, but reduces to one subproof only—the one on the right [Parigot, 1992, Bierman, 1999]. In this way Lafont's argument does not apply; it is completely bypassed. As a pleasing consequence, Bierman [1998] was able to show that $\lambda\mu$ has a simple computational interpretation: it is a simply-typed lambda calculus which is able to save and restore the runtime environment.

Another solution is given by Danos et al. [1997]. In their sequent calculus, named LK^{tq}, Lafont's proof is derivable, but every formula is required to be annotated with a colour. In effect, their solution of the "inconsistency" is similar to the approach taken by Parigot, in the sense that Proof (1.1) reduces to one subproof only. However, whereas in $\lambda\mu$ the reduction system is restricted so that Lafont's proof reduces to only the subproof on the right, in LK^{tq} this proof can reduce to either subproof, but the colour annotation determines which one. The colour annotation seems to correspond to the restriction imposed by linear logic, since every cut-elimination step in LK^{tq} can be mapped onto a series of cut-elimination steps in linear logic. However, the precise relationship needs yet to be worked out. Danos et al. give a computational interpretation for LK^{tq}-proofs, but their proposal is somewhat blurred.

Although Lafont's reasoning has found its way into a number of treatises on the proof theory of classical logic, for example [Girard et al., 1989, Girard, 1991, Schellinx, 1994, Bierman, 1999], there is an obvious question whether the restrictions mentioned above are *really* necessary to solve the problem with the "inconsistency" in Lafont's proof. There is a fear that certain computational features of classical logic are lost by these restrictions. In classical logic we do not have access to new functions, a fact established several years ago (for example by Friedman [1978]), but we do have access to new proofs, compared to intuitionistic logic. Thus we may hope to have access to a substantially larger class of programs capturing computational behaviour not expressible in intuitionistic typed lambda calculi. Unfortunately, none of the proposed solutions for Lafont's example have yet fulfilled this hope.

In this thesis we shall show that the restrictions are *not* necessary to obtain a strongly normalising cut-elimination procedure. Also, we want to argue that the restrictions are *not* required to solve the problem arising from Lafont's example. The "inconsistency" only appears if we regard cut-elimination as an equality preserving operation. This is of course a plausible doctrine coming from the simply-typed lambda calculus, but there are many calculi, notably calculi for concurrency, where reduction is *not* an equality preserving operation. Take for example a non-deterministic choice operator, say $+$, with the reduction

$$M + N \;\longrightarrow\; M \;\text{ or }\; N\ .$$

Clearly, we cannot hope that reduction preserves equality. Therefore, we shall accept the view that cut-elimination corresponds to computation which may or may not preserve equality between proofs and in this way avoid the "inconsistency" in Lafont's example. Under this assumption it seems prudent however to reconsider the proof theory of classical logic, despite the long list of textbooks, for instance [Schütte, 1960, Takeuti, 1975, Girard, 1987b, Troelstra and Schwichtenberg, 1996, Buss, 1998], already written on this subject.

1.1 Outline of the Thesis

We shall introduce several reduction systems in this thesis; Figure 1.1 gives an overview and roughly indicates the dependencies between them. The plan of the thesis is as follows:

- Chapter 2 reviews Gentzen's sequent calculus LK and his cut-elimination procedure. Then we introduce a new sequent calculus for classical logic and show how its proofs can be annotated with terms. Next, two cut-elimination procedures for this sequent calculus are formulated; they include a notion of proof substitution. The principal result of this thesis is a proof that establishes strong normalisation for both cut-elimination procedures. This result is then extended to first-order classical logic, to other formulations of classical logic and to intuitionistic logic. We also prove strong normalisation for a cut-elimination procedure where the proof substitution is replaced with completely local reduction rules. Having considered several strongly normalising cut-elimination procedures, we conclude this chapter comparing our work with work by Dragalin [1988] and work by Barbanera and Berardi [1994].

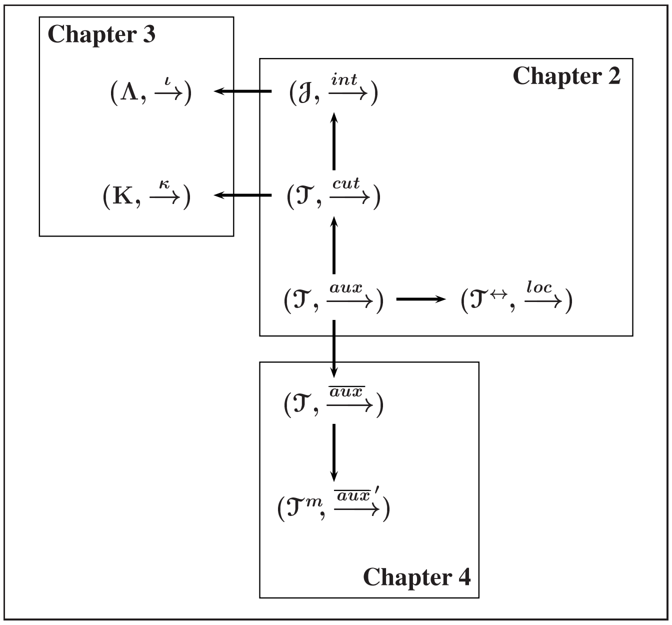


Figure 1.1: Overview of the thesis.

- Chapter 3 concerns the relation between normalisation and cut-elimination. After reviewing some standard material on natural deduction in intuitionistic logic, we show the correspondence between normalisation and cut-elimination in the $(\supset, \wedge)$-fragment of intuitionistic logic. Then, we consider a sequent-conclusion natural deduction formulation for classical logic and establish the correspondence between normalisation and cut-elimination for all connectives. We close this chapter with some remarks about the natural deduction calculus introduced by Parigot [1992] and compare our approach with the one taken by Ungar [1992].

- Chapter 4 describes first an implementation for one of our cut-elimination procedures. For this cut-elimination procedure we prove several properties that considerably simplify the implementation. Next, we analyse a particular classical proof and show that the process of eliminating its cuts corresponds to non-deterministic computation. We then introduce a typed, non-deterministic lambda calculus, which can be seen as a prototypical functional programming language with an erratic choice operator, and show how computation in this calculus can be simulated by cut-elimination in a fragment of classical logic. In the last section of this chapter we comment on the colour annotations introduced in LK^{tq}.

- Chapter 5 completes the dissertation by drawing some conclusions and suggesting further work.

- Appendix A contains two normal forms of the classical proof analysed in Chapter 4.

- Appendix B gives the details for some proofs omitted in the main text.

1.2 Contributions

The main contribution of this thesis is to re-examine the proof theory of classical logic under the assumption that cut-elimination is an operation which may or may not preserve the equality between proofs. Some specific contributions are listed below.

Chapter 2:

- A sequent calculus with completely implicit structural rules is developed for classical logic. A term assignment is given for the proofs of this calculus.
- Two strongly normalising cut-elimination procedures are designed. They are less restrictive than previous strongly normalising procedures and allow, in general, more normal forms to be reached.
- A strongly normalising cut-elimination procedure with completely local reduction rules is developed. As far as I am aware, it is the only strongly normalising cut-elimination procedure for classical logic whose reduction rules are local and which allows cuts to be permuted with other cuts.

Chapter 3:

- The correspondence between the standard normalisation procedure of NJ and the intuitionistic variant of one of our cut-elimination procedures is established in the $(\supset, \wedge)$-fragment.
- A counter example is given, which shows that certain cut-elimination reductions cannot be mapped, contrary to commonly accepted belief, onto reductions of the explicit substitution calculus $\lambda\mathsf{x}$, provided one uses the standard translations between sequent proofs and natural deduction proofs.
- A sequence-conclusion natural deduction calculus for classical logic is introduced. It is shown that its normalisation procedure is strongly normalising and that the normal natural deduction proofs respect the subformula property. For this natural deduction calculus the correspondence between normalisation and cut-elimination is established for all connectives.

Chapter 4:

- An implementation for a leftmost-outermost cut-reduction strategy is developed. This strategy does not restrict the normal forms reachable from a proof.
- The classical proof studied by Barbanera et al. [1997] is translated into our sequent calculus. The behaviour of this proof under cut-elimination is analysed, and it is shown that two of its normal forms differ in “essential” features. The normal forms are given in Appendix A.
- It is shown that normalisation in the simply-typed lambda calculus extended with an erratic choice operator can be simulated by cut-elimination in classical logic.

Chapter 2

Sequent Calculi

> In order to be able to enunciate and prove the *Hauptsatz* in a convenient form, I had to provide a logical calculus especially suited to the purpose. For this the natural deduction calculus proved unsuitable.
>
> —G. Gentzen
> in Investigations into Logical Deductions, 1935.

Two of Gentzen's most striking and highly original inventions are *sequent calculi* and *cut-elimination procedures*. In this chapter we shall show that only a small restriction on the standard cut-elimination procedure for classical logic is sufficient to obtain strongly normalising proof transformations. Prior to our work, some strongly normalising cut-elimination procedures, notably in [Dragalin, 1988, Danos et al., 1997], have been developed, but all of them impose some quite strong restrictions. We shall show that these restrictions are unnecessary to ensure strong normalisation. The basic idea of our cut-elimination procedure is to provide some means to prevent interference between proof transformations that simply permute cut-rules upwards in a proof.

We shall first describe Gentzen's sequent calculi LK and LJ, and the main ideas behind his *Hauptsatz* (cut-elimination theorem). Next, we shall devise term annotations for proofs of a sequent calculus of classical logic, whose inference rules are inspired by a variant of Kleene's sequent calculus G3. The main part of this chapter is occupied by an, unfortunately, rather lengthy proof of strong normalisation in which we adapt the technique of symmetric reducibility candidates developed by Barbanera and Berardi [1994]. Subsequently, we extend this result to the first-order case and to other formulations of the propositional fragment. The cut-elimination procedure for which we prove strong normalisation depends on a global, or non-local, notion of proof substitution. Using results concerning explicit substitution calculi we can replace the global operation with completely local proof transformations and obtain again a strongly normalising cut-elimination procedure.

2.1 Gentzen's Sequent Calculi

In his seminal paper [1935] Gentzen introduced the sequent calculi LJ and LK for intuitionistic and classical logic, respectively. He was able to show that for each sequent proof a normal form can be found in which all applications of the cut-rule are eliminated. Gentzen not only proved that all occurrences of this rule can be eliminated, but also gave a simple procedure for doing so. Before we outline this procedure, let us first examine LK in more detail.

A *sequent* in LK is a pair (Γ, Δ) of finite multisets of formulae, commonly written as $\Gamma \vdash \Delta$, where the formulae are given by the grammar

$$B ::= A \mid \neg B \mid B \wedge B \mid B \vee B \mid B \supset B\ .$$

We adopt the convention that A stands for atomic formulae, also called propositional symbols, and $B, C, \ldots$ for arbitrary formulae. The turnstile divides a sequent into two zones, called the *antecedent* and the *succedent*. The intuitive meaning of a sequent of the form

$$B_1, \ldots, B_n \vdash C_1, \ldots, C_m$$

is that the conjunction of the formulae, B_i, in the antecedent implies the disjunction of the formulae, C_j, in the succedent. There is no explicit constant for falsum in LK; however, a sequent with an empty succedent, say $\Gamma \vdash$, can be interpreted as Γ implies falsum.

The proofs in Gentzen's sequent calculi are tree-structures where the leaves are *axioms* of the form $\overline{B \vdash B}$ and the nodes are *inference rules*. There are three kinds of inference rules: logical rules, structural rules and the cut-rule. The logical and structural inference rules fall into two groups—left rules and right rules—depending on which half of the sequent they act on. Some of the logical rules are listed below.

$$\frac{B_i, \Gamma \vdash \Delta}{B_1 \wedge B_2, \Gamma \vdash \Delta}\ \wedge_{L_i} \qquad \frac{\Gamma \vdash \Delta, B \quad \Gamma \vdash \Delta, C}{\Gamma \vdash \Delta, B \wedge C}\ \wedge_R$$

$$\frac{B, \Gamma \vdash \Delta \quad C, \Gamma \vdash \Delta}{B \vee C, \Gamma \vdash \Delta}\ \vee_L \qquad \frac{\Gamma \vdash \Delta, B_i}{\Gamma \vdash \Delta, B_1 \vee B_2}\ \vee_{R_i}$$

$$\frac{\Gamma_1 \vdash \Delta_1, B \quad C, \Gamma_2 \vdash \Delta_2}{B \supset C, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ \supset_L \qquad \frac{B, \Gamma \vdash \Delta, C}{\Gamma \vdash \Delta, B \supset C}\ \supset_R$$

We say the $\wedge_{L_i}$-rule introduces the formula $B_1 \wedge B_2$—the *main formula* of this inference rule—and refer to Γ and Δ as *contexts* (multisets of formulae). In both the premise and conclusion of this rule the comma is interpreted as the multiset union. The terminology given for $\wedge_{L_i}$ also applies to the other rules.

There is a slight peculiarity in Gentzen's $\supset_L$-rule, in which, if we read it from bottom to top, the contexts are split up into two multisets. In contrast, in the inference rules $\wedge_R$

and $\vee_L$ both premises share the same context. The rules sharing the contexts are said to be *additive*, while the rules splitting the contexts are said to be *multiplicative*. Every inference rule with two premises can be defined either way.

In our formulation of Gentzen's LK there are four structural rules again divided into left and right rules.[1]

$$\frac{\Gamma \vdash \Delta}{B, \Gamma \vdash \Delta}\ \mathrm{Weak}_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, B}\ \mathrm{Weak}_R$$

$$\frac{B, B, \Gamma \vdash \Delta}{B, \Gamma \vdash \Delta}\ \mathrm{Contr}_L \qquad \frac{\Gamma \vdash \Delta, B, B}{\Gamma \vdash \Delta, B}\ \mathrm{Contr}_R$$

The only inference rule that is neither a left nor a right rule is the cut-rule

$$\frac{\Gamma_1 \vdash \Delta_1, B \quad B, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ \mathrm{Cut}$$

in which the formula B is called the *cut-formula*.

The intuitionistic sequent calculus LJ can be obtained from LK by restricting the succedent of sequents to at most a single formula, by dropping the Contr_R-rule and by replacing the $\supset_L$-rule with

$$\frac{\Gamma_1 \vdash B \quad C, \Gamma_2 \vdash D}{B \supset C, \Gamma_1, \Gamma_2 \vdash D}\ \supset_L .$$

One of Gentzen's motives when designing the sequent calculi LK and LJ was to provide deductive systems for classical and intuitionistic logic, respectively, for which *consistency* could be established by simple methods. Let us briefly illustrate his argument for consistency. First notice that in sequent calculi consistency coincides with the absence of a proof for the empty sequent (i.e., both the antecedent and succedent are empty). Assume we would be given a proof of this sequent, then we can construct a proof of the sequent $\vdash B \wedge \neg B$ simply by applying Weak_R. In fact, we would be able to construct for any formula C a proof of the sequent $\vdash C$. Obviously, LK and LJ would then be inconsistent. Fortunately, there is no proof of the empty sequent in either LK or LJ. Although this is not obvious, it is easy to verify that there cannot be any such proof that enjoys the subformula property.

Definition 2.1.1 (Subformula and Subformula Property)**:**

- The relation B *is a subformula of* C is defined as the transitive, reflexive closure of the clauses:
  - B, C are subformulae of $B \wedge C$, $B \vee C$ and $B \supset C$, and
  - B is a subformula of $\neg B$.
- A proof π of the sequent $\Gamma \vdash \Delta$ enjoys the *subformula property*, if and only if all formulae occurring in π are subformulae of the formulae in Γ and Δ.

[1] Gentzen defined sequents using sequences of formulae and therefore also had an explicit Exchange-rule.

The crux of Gentzen's argument for consistency is that in LJ and LK all cut-free proofs enjoy the subformula property: the premises of the logical and structural inference rules contain only subformulae of their respective conclusion. If there were a proof of the empty sequent, then according to his *Hauptsatz* there must be one without cuts. But this cut-free proof would lack the subformula property. Consequently, there cannot be any proof of the empty sequent in either LK or LJ.

Since Gentzen's work many *Hauptsätze* have appeared for various sequent calculus formulations. They all use his technique of proof transformations, referred to as *cut-reductions*, which convert a proof containing cuts not immediately to a cut-free proof, but replace the cuts by simpler cuts, or permute the cuts towards the leaves where they eventually vanish. Iterating this process of applying proof transformations will lead to a cut-free proof. We shall sketch the main lines of Gentzen's *Hauptsatz* by giving some of his cut-reductions. However, in what follows we shall study several different cut-reduction systems, and for this it is convenient to introduce some notation and terminology commonly used in connection with term rewriting (see for example [Baader and Nipkow, 1998]).

Definition 2.1.2:

- A *reduction system* is the pair $(\mathcal{A}, \longrightarrow)$, in which
  - $\mathcal{A}$ is a set of objects (in this thesis usually a set of terms or proofs), and
  - $\longrightarrow$ is a binary relation over $\mathcal{A}$; it is often referred to as a *reduction*.

  Instead of writing $(a, b) \in\longrightarrow$, we write $a \longrightarrow b$ and say a reduces to b. The transitive and the reflexive transitive closure of $\longrightarrow$ is written as $\longrightarrow^{+}$ and $\longrightarrow^{*}$, respectively. We shall often annotate the '$\longrightarrow$' to qualify the reduction in question.

  Whenever we define a reduction, say $a \longrightarrow b$, we automatically assume that the reduction is closed under context formation. This is a standard convention in term rewriting.

- A term is said to be a *normal form* if and only if it does not reduce. In this thesis a normal form often corresponds to a cut-free proof.

- Let a be an element of $\mathcal{A}$, then a is said to be *strongly normalising* if and only if all reduction sequences starting from a are finite. In this case we write $a \in SN_r$ and use $\textsc{maxred}_r(a)$ to denote the longest reduction sequence starting from a (the annotation r indicates to which reduction these notions refer to). Correspondingly, a reduction system is said to be strongly normalising if and only if all elements of $\mathcal{A}$ are strongly normalising.

Cut-reductions come in two flavours depending on whether they apply to *logical cuts* or *commuting cuts*. A cut is said to be a logical cut when the cut-formula is introduced on both sides by axioms or logical inference rules; otherwise the cut is said to be a commuting cut. An instance of a logical cut is as follows.

$$\dfrac{\dfrac{\Gamma_1 \vdash \Delta_1, B \quad \Gamma_1 \vdash \Delta_1, C}{\Gamma_1 \vdash \Delta_1, B\wedge C}\,\wedge_R \qquad \dfrac{B, \Gamma_2 \vdash \Delta_2}{B\wedge C, \Gamma_2 \vdash \Delta_2}\,\wedge_{L_1}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,\text{Cut}$$

Here the cut-formula, $B\wedge C$, is the main formula in both the $\wedge_R$ and $\wedge_{L_1}$-rule. According to Gentzen's procedure this cut reduces to

$$\dfrac{\Gamma_1 \vdash \Delta_1, B \quad B, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,\text{Cut}$$

where the degree of the cut-formula has decreased (the degree of a formula is defined as usual). Another instance of a logical cut is as follows.

$$\dfrac{\dfrac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, B\vee C}\,\vee_{R_1} \qquad \dfrac{}{B\vee C \vdash B\vee C}}{\Gamma \vdash \Delta, B\vee C}\,\text{Cut}$$

Here the cut-formula is introduced by a logical rule and by an axiom. Clearly, this application of the cut can disappear, to yield the following proof.

$$\dfrac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, B\vee C}\,\vee_{R_1}$$

In the cases where a cut cannot be transformed by a cut-reduction for logical cuts, then the cut is a commuting cut, for which proof transformations are defined permuting the cut towards the leaves. A typical instance of a commuting cut is the proof

$$\dfrac{\dfrac{B, \Gamma_1 \vdash \Delta_1, C, D}{\Gamma_1 \vdash \Delta_1, B\supset C, D}\,\supset_R \qquad \dfrac{D, E, \Gamma_2 \vdash \Delta_2}{D, E\wedge F, \Gamma_2 \vdash \Delta_2}\,\wedge_{L_1}}{E\wedge F, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, B\supset C}\,\text{Cut} \tag{2.1}$$

whose cut-formula, D, is not introduced by the inference rules directly above the cut. In this case, the cut is permuted upwards in the proof yielding either one of the following two proofs.

$$\dfrac{\dfrac{B, \Gamma_1 \vdash \Delta_1, C, D \quad \dfrac{D, E, \Gamma_2 \vdash \Delta_2}{D, E\wedge F, \Gamma_2 \vdash \Delta_2}\,\wedge_{L_1}}{B, E\wedge F, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, C}\,\text{Cut}}{E\wedge F, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, B\supset C}\,\supset_R \qquad \dfrac{\dfrac{\dfrac{B, \Gamma_1 \vdash \Delta_1, C, D}{\Gamma_1 \vdash \Delta_1, B\supset C, D}\,\supset_R \quad D, E, \Gamma_2 \vdash \Delta_2}{E, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, B\supset C}\,\text{Cut}}{E\wedge F, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, B\supset C}\,\wedge_{L_1}$$

Usually, it is left unspecified which alternative is taken, and therefore commuting cuts are a source of non-determinism. This is a point we shall study more closely in the following sections of this chapter.

The intricacies of a cut-elimination theorem lie in the fact that one has to show that the process of applying cut-reductions terminates; this is complicated in LK and other sequent calculi by the presence of contractions. Consider the following commuting cut.

$$\cfrac{\cfrac{\Gamma_1 \vdash \Delta_1, B, B}{\Gamma_1 \vdash \Delta_1, B}\,\mathrm{Contr}_R \qquad B, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,\mathrm{Cut} \tag{2.2}$$

This cut reduces to

$$\begin{array}{c} \cfrac{\cfrac{\Gamma_1 \vdash \Delta_1, B, B \qquad B, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, B}\,\mathrm{Cut} \qquad B, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2, \Gamma_2 \vdash \Delta_1, \Delta_2, \Delta_2}\,\mathrm{Cut} \\ \hline\hline \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2 \end{array}\,\mathrm{Contr} \tag{2.3}$$

where the right subproof has been duplicated and where the double lines stand for several contractions. The reader will see that the proof-size of the reduct is vastly bigger than the proof from which we started. In fact, it is not hard to check that the cut-elimination process can result into super-exponential growth of the proof-size; see for example [Girard et al., 1989, Page 111].

When designing a cut-elimination procedure one therefore has to take care that for every sequent proof the process of applying cut-reductions actually terminates; that means it leads to a cut-free proof. In this thesis we shall introduce several novel cut-elimination procedures for classical logic. The design of these procedures is motivated by the following three criteria:

- first, the cut-elimination procedure should *not* restrict the collection of normal forms reachable from a given proof in such a way that "essential" normal forms are no longer reachable,

- second, the cut-elimination procedure should be *strongly normalising*, i.e., all possible reduction strategies should terminate, and

- third, the cut-elimination procedure should allow cuts to pass over other cuts.

At the time of writing, we are unaware of any other cut-elimination procedure for a sequent calculus formulation of classical logic that satisfies all three criteria. So let us justify these criteria.

As we have seen above, some cut-reductions can be applied in a non-deterministic fashion so that applying different cut-reductions may result in different normal forms. With respect to our first criterion, most existing cut-elimination procedures, including Gentzen's original, are thus quite unsatisfactory since they terminate only if a particular strategy for applying cut-reductions is employed—they are *weakly normalising*. Common examples being an innermost reduction strategy, or the elimination of the cut with the highest rank. An unpleasant consequence of these strategies is that they surly restrict the number of normal forms reachable from a given proof. As we shall demonstrate in Chapter 4, the normal forms reachable from a proof play however an important rôle in investigating a computational interpretation for classical proofs. Therefore our first two criteria.

As a first attempt for a strongly normalising cut-elimination procedure one might take simply an unrestricted version of Gentzen's cut-elimination procedure; that is by removing the strategy. Unfortunately, this would, as stated earlier, allow infinite reduction sequences, one of which is illustrated in the following example given by Gallier [1993] and Danos et al. [1997].

Example 2.1.3: Consider the proof

$$
\begin{prooftree}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\vee_L$}\BinaryInfC{$A\vee A\vdash A, A$}
\RightLabel{$\textbf{Contr}_R$}\UnaryInfC{$A\vee A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\wedge_R$}\BinaryInfC{$A, A\vdash A\wedge A$}
\RightLabel{$\textbf{Contr}_L$}\UnaryInfC{$A\vdash A\wedge A$}
\RightLabel{\textbf{Cut}}\BinaryInfC{$A\vee A\vdash A\wedge A$}
\end{prooftree}
$$

The problem lies with the lower cut, which needs to be permuted upwards. There are two possible reductions: either the cut can be permuted upwards in the left proof branch or in the right proof branch. In both cases a subproof needs to be duplicated. If one is not careful, applying these reductions in alternation can lead to arbitrary big normal forms and to non-termination. For example, consider the reduction sequence starting with the proof above and continuing as follows

$$
\begin{prooftree}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\vee_L$}\BinaryInfC{$A\vee A\vdash A, A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\wedge_R$}\BinaryInfC{$A, A\vdash A\wedge A$}
\RightLabel{Contr_L}\UnaryInfC{$A\vdash A\wedge A$}
\RightLabel{Cut}\BinaryInfC{$A\vee A\vdash A, A\wedge A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\wedge_R$}\BinaryInfC{$A, A\vdash A\wedge A$}
\RightLabel{Contr_L}\UnaryInfC{$A\vdash A\wedge A$}
\RightLabel{Cut}\BinaryInfC{$A\vee A\vdash A\wedge A, A\wedge A$}
\RightLabel{Contr_R}\UnaryInfC{$A\vee A\vdash A\wedge A$}
\end{prooftree}
$$

where the cut is permuted to the left, creating two copies of the right subproof. Now permute the upper cut to the right, which gives the following proof.

$$
\begin{prooftree}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\vee_L$}\BinaryInfC{$A\vee A\vdash A, A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\vee_L$}\BinaryInfC{$A\vee A\vdash A, A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\wedge_R$}\BinaryInfC{$A, A\vdash A\wedge A$}
\RightLabel{Cut}\BinaryInfC{$A\vee A, A\vdash A, A\wedge A$}
\RightLabel{Cut}\BinaryInfC{$A\vee A, A\vee A\vdash A, A, A\wedge A$}
\RightLabel{Contr_L}\UnaryInfC{$A\vee A\vdash A, A, A\wedge A$}
\RightLabel{$\textbf{Contr}_R$}\UnaryInfC{$A\vee A\vdash A, A\wedge A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\AxiomC{}\UnaryInfC{$A\vdash A$}
\RightLabel{$\wedge_R$}\BinaryInfC{$A, A\vdash A\wedge A$}
\RightLabel{$\textbf{Contr}_L$}\UnaryInfC{$A\vdash A\wedge A$}
\RightLabel{\textbf{Cut}}\BinaryInfC{$A\vee A\vdash A\wedge A, A\wedge A$}
\RightLabel{Contr_R}\UnaryInfC{$A\vee A\vdash A\wedge A$}
\end{prooftree}
$$

This proof contains an instance of the reduction applied in the first step (bold face). Even worse, it is bigger than the proof with which we started, and so in effect we can construct reduction sequences with possibly infinitely big normal forms.

It seems difficult to avoid the infinite reduction sequence given in the example above using an unrestricted Gentzen-like formulation of the cut-reductions. A number of people, for example Dragalin [1988], Herbelin [1994], Cichon et al. [1996], Danos et al. [1997] and Bittar [1999], have managed to develop strongly normalising cut-elimination procedures, but they all impose fairly strong restrictions on the cut-reductions.

Here is one common restriction: consider the following reduction rule, which allows a cut-rule (Suffix 2) to pass over another cut-rule (Suffix 1).

$$\dfrac{\dfrac{\ldots\vdash\ldots \quad \ldots\vdash\ldots}{\ldots\vdash\ldots}\,\text{Cut}_1 \quad \ldots\vdash\ldots}{\ldots\vdash\ldots}\,\text{Cut}_2 \longrightarrow \dfrac{\dfrac{\ldots\vdash\ldots \quad \ldots\vdash\ldots}{\ldots\vdash\ldots}\,\text{Cut}_2 \quad \ldots\vdash\ldots}{\ldots\vdash\ldots}\,\text{Cut}_1$$

Clearly, this reduction would immediately break strong normalisation because the reduct is again an instance of this rule, and we can loop by constantly applying this rule. Thus a common restriction is not to allow a cut-rule to pass over another cut-rule in any circumstances. However, this has several serious drawbacks. In the intuitionistic case, for example, such a restriction limits the correspondence between cut-elimination and beta-reduction. In particular, strong normalisation of beta-reduction cannot be inferred from strong normalisation of cut-elimination, as noted by Herbelin [1994] and by Dyckhoff and Pinto [1998]. Therefore our third criterion.

In the rest of this chapter we shall develop strongly normalising cut-elimination procedures which contain the standard Gentzen-like cut-elimination steps for logical cuts and allow commuting cuts to pass over other cuts. As a pleasing result, we can simulate beta-reduction and infer strong normalisation for the simply-typed lambda calculus from the strong normalisation result of one of our cut-elimination procedures. We shall prove this result in Chapter 3.

Danos et al. allow cut-rules to pass over other cut-rules in their strongly normalising cut-elimination procedure introduced for the sequent calculus LK^{tq} [Danos et al., 1997, Joinet et al., 1998]. So this cut-elimination procedure satisfies our second and third criterion, but as we shall see it violates the first. In LK^{tq} every formula (and its subformulae) are required to be coloured with either '$\leftharpoonup$' or '$\rightharpoonup$'. Here is an instance of a cut-rule in LK^{tq}.

$$\dfrac{\overleftharpoon{A} \vdash \overleftharpoon{\overleftharpoon{B} \wedge \overrightharpoon{C}} \qquad \overleftharpoon{\overleftharpoon{B} \wedge \overrightharpoon{C}} \vdash \overrightharpoon{D}}{\overleftharpoon{A} \vdash \overrightharpoon{D}}\,\text{Cut}$$

Recall the problematic infinite reduction sequence in Example 2.1.3. This sequence is avoided in LK^{tq} by devising a specific protocol for cut-elimination, which uses the additional information provided by the colours. If in a commuting cut the colour '$\leftharpoonup$' is attached to the cut-formula, then the commuting cut is permuted to the left, and similarly for the '$\rightharpoonup$'-colour (hence the use of an arrow to denote a colour!). Suppose we have the following colour annotation for the commuting cut given in (2.1).

$$\dfrac{\dfrac{\overrightharpoon{B} \vdash \overrightharpoon{C}, \overleftharpoon{D}}{\vdash \overleftharpoon{\overrightharpoon{B} \supset \overrightharpoon{C}}, \boxed{\overleftharpoon{D}}}\,\supset_R \qquad \dfrac{\overleftharpoon{D}, \overrightharpoon{E} \vdash}{\boxed{\overleftharpoon{D}}, \overrightharpoon{\overrightharpoon{E} \wedge \overrightharpoon{F}} \vdash}\,\wedge_{L_1}}{\overrightharpoon{\overrightharpoon{E} \wedge \overrightharpoon{F}} \vdash \overrightharpoon{\overrightharpoon{B} \supset \overrightharpoon{C}}}\,\text{Cut}$$

Then this cut, because the cut-formula D is annotated with '$\leftharpoonup$', reduces to the following proof instance, only.

$$\dfrac{\dfrac{\overrightharpoon{B} \vdash \overrightharpoon{C}, \overleftharpoon{D} \qquad \dfrac{\overleftharpoon{D}, \overrightharpoon{E} \vdash}{\overleftharpoon{D}, \overrightharpoon{\overrightharpoon{E} \wedge \overrightharpoon{F}} \vdash}\,\wedge_{L_1}}{\overrightharpoon{B}, \overrightharpoon{\overrightharpoon{E} \wedge \overrightharpoon{F}} \vdash \overrightharpoon{C}}\,\text{Cut}}{\overrightharpoon{\overrightharpoon{E} \wedge \overrightharpoon{F}} \vdash \overrightharpoon{\overrightharpoon{B} \supset \overrightharpoon{C}}}\,\supset_R$$

By enforcing that commuting cuts can be permuted into one direction only, the infinite reduction sequence given in Example 2.1.3 cannot be constructed. However, there are two annoying restrictions in their cut-elimination procedure for LK^{tq}, both of which violate our first criterion.

- First, there is a problem with the compositionality of the colour annotation, in the sense that some cuts require the same colour annotation for their cut-formulae: the choice of a colouring can permeate through a proof. In particular, the colour annotation has to respect, using terminology introduced for LK^{tq}, *identity classes* [Schellinx, 1994, Page 107]. For example, in the following LK-proof with two commuting cuts

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \vdash B\end{matrix} \qquad \begin{matrix}\overline{B \vdash B}\\ \vdots\\ B,\Gamma \vdash \Delta, B\end{matrix}}{\Gamma \vdash \Delta, B}\text{Cut} \qquad \begin{matrix}\vdots\\ B \vdash\end{matrix}}{\Gamma \vdash \Delta}\text{Cut}$$

  all occurrences of B must in LK^{tq} have the same colour. Consequently, the normal forms that arise in LK by permuting both cuts towards the axiom, where they merge into a single cut, cannot be obtained in LK^{tq}.

- Second, the colour annotation is invariant under cut-reductions. Thus whenever a cut is duplicated in a reduction sequence, the colour annotation prevents both instances from reducing differently. Figure 2.1 gives an example of such a reduction sequence that exists in LK, but not in LK^{tq}.

Making the cut-elimination procedure dependent on the colour annotations is a strong restriction: certainly some normal forms can no longer be reached. In fact, the colours constrain the cut-elimination procedure to be confluent, which is an essential property in the strong normalisation proof of Danos et al.: it enabled them to exploit the strong normalisation result for proof-nets in linear logic [Girard, 1987a]. The colours are used ingeniously to map every LK^{tq}-proof to a corresponding proof-net in linear logic and every cut-elimination step to a series of reductions on proof-nets.

We shall show that the colour annotations of LK^{tq} are, in fact, unnecessary to ensure strong normalisation of cut-elimination. As stated earlier, a pleasing consequence is that, in general, more normal forms can be reached from a given proof containing cuts. One not-so-pleasing consequence of the generality of our cut-elimination procedure is that strong normalisation is more difficult to prove; it cannot, for example, be proved by a translation into proof-nets. In the end, we found it extremely useful to develop a term calculus for sequent derivations. This then allowed us to adapt directly a powerful proof technique from the term rewriting literature [Barbanera and Berardi, 1994].

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vee A\vdash A\vee A}\qquad \dfrac{}{A\vee A\vdash A\vee A}}{(A\vee A)\vee(A\vee A)\vdash A\vee A, A\vee A}\,\vee_L}{(A\vee A)\vee(A\vee A)\vdash A\vee A}\,\mathrm{Contr}_R \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A, A}\,\vee_L}{A\vee A\vdash A}\,\mathrm{Contr}_R \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A, A\vdash A\wedge A}\,\wedge_R}{A\vdash A\wedge A}\,\mathrm{Contr}_L}{A\vee A\vdash A\wedge A}\,\mathrm{Cut}\star}{(A\vee A)\vee(A\vee A)\vdash A\wedge A}\,\mathrm{Cut}$$

$$\downarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A, A}\,\vee_L}{A\vee A\vdash A}\,\mathrm{Contr}_R \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A, A\vdash A\wedge A}\,\wedge_R}{A\vdash A\wedge A}\,\mathrm{Contr}_L}{A\vee A\vdash A\wedge A}\,\mathrm{Cut}\star \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A, A}\,\vee_L}{A\vee A\vdash A}\,\mathrm{Contr}_R \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A, A\vdash A\wedge A}\,\wedge_R}{A\vdash A\wedge A}\,\mathrm{Contr}_L}{A\vee A\vdash A\wedge A}\,\mathrm{Cut}\star}{(A\vee A)\vee(A\vee A)\vdash A\vee A, A\vee A}\,\vee_L}{(A\vee A)\vee(A\vee A)\vdash A\vee A}\,\mathrm{Contr}_R$$

$$\downarrow$$

$$\vdots$$

$$\downarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A, A\vdash A\wedge A}\,\wedge_R}{A\vdash A\wedge A}\,\mathrm{Contr}_L \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A, A\vdash A\wedge A}\,\wedge_R}{A\vdash A\wedge A}\,\mathrm{Contr}_L}{A\vee A\vdash A\wedge A, A\wedge A}\,\vee_L}{A\vee A\vdash A\wedge A}\,\mathrm{Contr}_R \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A, A}\,\vee_L}{A\vee A\vdash A}\,\mathrm{Contr}_R \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A, A}\,\vee_L}{A\vee A\vdash A}\,\mathrm{Contr}_R}{A\vee A, A\vee A\vdash A\wedge A}\,\wedge_R}{A\vee A\vdash A\wedge A}\,\mathrm{Contr}_L}{(A\vee A)\vee(A\vee A)\vdash A\wedge A, A\wedge A}\,\vee_L}{(A\vee A)\vee(A\vee A)\vdash A\wedge A}\,\mathrm{Contr}_R$$

Figure 2.1: The displayed LK-proofs are taken from a reduction sequence that starts with the first proof and ends with the third proof—a normal form. The second proof is an intermediate step. The cuts in the top proof are eliminated such that first the right subproof is duplicated creating two instances of the cut marked with a star (second proof). Subsequently, each copy of this cut is reduced applying different reduction rules. This reduction sequence is impossible in a cut-reduction system with colour-annotation, because the colours prevent the two copies of the cut from reducing differently. In effect, starting from the first proof the given normal form is not reachable in LK^{tq} (in Section 4.4.1 we shall show that this is an "essential" normal form).

To be able to present our cut-elimination procedures in a convenient form, we shall introduce a non-standard formulation of the sequent calculus. This sequent calculus is inspired by a variant of Kleene's sequent calculus G3. Kleene [1952a, Page 481] mentioned this variant as G3a, but did not formalise it. Later, a complete formalisation of a calculus similar to G3a, named G3c, was given by Troelstra and Schwichtenberg [1996]. The distinguishing feature of these sequent calculi is that they have no explicit rules for weakening and contraction. In our calculus contraction and weakening are implicit in the form of the axioms and logical rules, which is somewhat analogous to the approach taken by Kleene in G3a. On one hand, this reduces the number of inference rules and thus simplifies the calculus; however on the other hand, the formalisation of the contexts and inference rules is quite intricate.

In the sequel we shall regard contexts as sets of (label,formula) pairs, as in type theory, and *not* as multisets, as in LK or LJ. This is to avoid well-known problems reported by Leivant [1979], which would break the correspondence between the sequent proofs and our term annotations (the term annotations will be given in Section 2.2). Since we have contexts on both sides of the turnstile, it is convenient to separate the labels into *names* and *co-names*; a, b, c,... stand for co-names and $\ldots, x, y, z$ for names. In this way, the contexts on the left-hand side of the turnstile, called left-contexts, will be built up by (name,formula) pairs and on the right-hand side by (co-name,formula) pairs; we shall refer to those contexts as right-contexts. Suppose we have the contexts

$$\begin{aligned}\Gamma &= \{(x,B),(y,C),(z,D)\} \text{ and}\\ \Delta &= \{(a,E),(b,F)\}\end{aligned}$$

then we can form the sequent $\Gamma \vdash \Delta$. With some minor abuse of language we say $\{x,y,z\}$ is the *domain* of Γ, written $dom(\Gamma)$; $\{a,b\}$ respectively is the domain of Δ. In the rest of the thesis we shall employ some shorthand notation for contexts; rather than writing for example $\{(x,B),(y,C),(z,D)\}$, we shall simply write $x:B, y:C, z:D$.

Figure 2.2 gives the inference rules of our sequent calculus. Note that there are a number of subtleties concerning contexts implicit in our inference rules. First, we assume the convention that a context is ill-formed, if it contains more than one occurrence of a name or co-name. For example, the left-context $x:B, x:C$ is not allowed. Hereafter, this will be referred to as the *context convention*, and it will be assumed that all inference rules having as contexts sets of (label,formula) pairs respect this convention.

Second, we have the following conventions for the commas in Figure 2.2: a comma in a conclusion stands for set union and a comma in a premise stands for *disjoint* set union. Consider for example the $\supset_R$-rule.

$$\frac{x:B,\Gamma \vdash \Delta, a:C}{\Gamma \vdash \Delta, b:B\supset C}\supset_R$$

This rule introduces the (co-name,formula) pair $b:B\supset C$ in the conclusion. However, there might be an *implicit contraction*. By this we mean b might already occur in the

$$\overline{x:B,\Gamma\vdash\Delta,a:B}$$

$$\frac{\Gamma\vdash\Delta,a:B}{x:\neg B,\Gamma\vdash\Delta}\,\neg_L \qquad \frac{x:B,\Gamma\vdash\Delta}{\Gamma\vdash\Delta,a:\neg B}\,\neg_R$$

$$\frac{x:B_i,\Gamma\vdash\Delta}{y:B_1\wedge B_2,\Gamma\vdash\Delta}\,\wedge_{L_i} \qquad \frac{\Gamma\vdash\Delta,a:B \quad \Gamma\vdash\Delta,b:C}{\Gamma\vdash\Delta,c:B\wedge C}\,\wedge_R$$

$$\frac{x:B,\Gamma\vdash\Delta \quad y:C,\Gamma\vdash\Delta}{z:B\vee C,\Gamma\vdash\Delta}\,\vee_L \qquad \frac{\Gamma\vdash\Delta,a:B_i}{\Gamma\vdash\Delta,b:B_1\vee B_2}\,\vee_{R_i}$$

$$\frac{\Gamma\vdash\Delta,a:B \quad x:C,\Gamma\vdash\Delta}{y:B\supset C,\Gamma\vdash\Delta}\,\supset_L \qquad \frac{x:B,\Gamma\vdash\Delta,a:C}{\Gamma\vdash\Delta,b:B\supset C}\,\supset_R$$

$$\frac{\Gamma_1\vdash\Delta_1,a:B \quad x:B,\Gamma_2\vdash\Delta_2}{\Gamma_1,\Gamma_2\vdash\Delta_1,\Delta_2}\,\text{Cut}$$

Figure 2.2: Variant of Kleene's sequent calculus G3a.

premise. Thus the conclusion of the $\supset_R$-rule is of the form

$$\Gamma\vdash\Delta\oplus b:B\supset C$$

where $\oplus$ denotes set union. Note that $x : B$ and $a : C$ are not part of the conclusion because they are intended to be discharged. Hence the premise must be of the form

$$x:B\otimes\Gamma\vdash\Delta\otimes a:C$$

where $\otimes$ denotes disjoint set union.

There is one point worth mentioning in the cut-rule, because it is the only inference rule in our sequent calculus that does not share the contexts, but requires that the two contexts of the premises are joined on each side of the conclusion. Thus we take the cut-rule to be of the form

$$\frac{\Gamma_1\vdash\Delta_1\otimes a:B \quad x:B\otimes\Gamma_2\vdash\Delta_2}{\Gamma_1\oplus\Gamma_2\vdash\Delta_1\oplus\Delta_2}\,\text{Cut}\ .$$

In consequence, this rule is only applicable, if it does not break the context convention, which can always be achieved by renaming some labels appropriately. Note, however, we do not require that cuts have to be "fully" multiplicative: the Γ_i's can share some formulae (similarly the Δ_j's).

2.2 Cut-Elimination in Propositional Classical Logic

In this section we shall introduce a strongly normalising cut-elimination procedure. Usually, sequent proofs are written pictorially in a tree-like fashion using inference rules of the form $\frac{P_1}{C}$ or $\frac{P_1\,P_2}{C}$. This is however a rather space consuming notation. In order to present our strong normalisation proof in a manageable form, we shall annotate our sequent proofs with terms and express the cut-elimination procedure as a term rewriting system. In particular, we are able to express a proof transformation as a special sort of proof substitution.

For sequent calculi, even in the intuitionistic case, it is not entirely clear what appropriate term annotations are. In fact, there are a number of choices, and there is no real consensus on the best. Our term calculus encodes the entire structure of a sequent calculus proof and thus allows us to define a complete cut-elimination procedure as a term rewriting system.[2] Other proposals use lambda terms as annotation [Pottinger, 1977, Abramsky, 1993, Barendregt and Ghilezan, 2000], which however do not fully encode the structure of the sequent proofs and so would seem less useful for our purposes.

Definition 2.2.1 (Raw Terms)**:** The set of raw terms, $\mathcal{R}$, is defined by the grammar

$$\begin{array}{rcll}
M,N & ::= & \mathsf{Ax}(x,a) & \text{axiom} \\
& \mid & \mathsf{Cut}(\langle a{:}B\rangle M,(x{:}B)N) & \text{cut} \\
& \mid & \mathsf{Not}_R((x{:}B)M,a) & \text{not-right} \\
& \mid & \mathsf{Not}_L(\langle a{:}B\rangle M,x) & \text{not-left} \\
& \mid & \mathsf{And}_R(\langle a{:}B\rangle M,\langle b{:}C\rangle N,c) & \text{and-right} \\
& \mid & \mathsf{And}^i_L((x{:}B)M,y) & \text{and-left}_i \quad i=1,2 \\
& \mid & \mathsf{Or}^i_R(\langle a{:}B\rangle M,b) & \text{or-right}_i \quad i=1,2 \\
& \mid & \mathsf{Or}_L((x{:}B)M,(y{:}C)N,z) & \text{or-left} \\
& \mid & \mathsf{Imp}_R((x{:}B)\langle a{:}C\rangle M,b) & \text{implication-right} \\
& \mid & \mathsf{Imp}_L(\langle a{:}B\rangle M,(x{:}C)N,y) & \text{implication-left}
\end{array}$$

where B and C are types (formulae); x, y, z are taken from a set of names and a, b, c from a set of co-names.

In a term we use round brackets to signify that a name becomes bound and angle brackets that a co-name becomes bound. In what follows we shall often omit the type annotations on bindings, and assume that $FN(M)$ and $FC(M)$ stand, respectively, for the set of free names and free co-names of the term M. We do not give the formal definitions of $FN(\_)$ and $FC(\_)$, rather give the reader two illustrative examples. Consider the following terms.

$$\mathsf{Ax}(x,a) \qquad \mathsf{Imp}_L(\langle b\rangle M,(y)N,z)$$

Here, x and z are free names, and a is a free co-name. On the other hand, y and b are a bound name and bound co-name, respectively. For convenience $FN(\_)$ and $FC(\_)$ will not be restricted to just a single term, but also be used for sets of terms. Notice that names and co-names are not the same notions as a variable in the lambda calculus: whilst a term can be substituted for a variable, a name or a co-name can only be "renamed". Rewriting

[2] Pfenning [1995] developed similar terms for an encoding of a weakly normalising cut-elimination procedure into LF.

a name x to y in M is written as $M[x \mapsto y]$, and rewriting a co-name a to b in M is written as $M[a \mapsto b]$. The routine formalisation of these rewriting operations is omitted.

To annotate terms to the sequent proofs of the calculus given in Figure 2.2, we shall replace every sequent of the form $\Gamma \vdash \Delta$ with a *typing judgement* of the form $\Gamma \triangleright M \triangleright \Delta$ in which M is a term, Γ a left-context and Δ a right-context. Henceforth we shall be interested in *well-typed* terms, only; this means those for which there are two contexts, Γ and Δ, such that $\Gamma \triangleright M \triangleright \Delta$ holds given the inference rules in Figure 2.3. We define the following set of well-typed terms.

$$\mathcal{T} \stackrel{\text{def}}{=} \left\{ M \;\middle|\; M \in \mathcal{R} \text{ and well-typed by the rules shown in Figure 2.3} \right\}$$

The definition we give next corresponds to the traditional notion of the main formula of an inference rule.

Definition 2.2.2:

- A term, M, *introduces* the name z or co-name c, if and only if M is of the form:

for z:		for c:	
	$\mathsf{Ax}(z, c)$		$\mathsf{Ax}(z, c)$
	$\mathsf{Not}_L(\langle a\rangle S, z)$		$\mathsf{Not}_R((x)S, c)$
	$\mathsf{And}^i_L((x)S, z)$		$\mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c)$
	$\mathsf{Or}_L((x)S, (y)T, z)$		$\mathsf{Or}^i_R(\langle a\rangle S, c)$
	$\mathsf{Imp}_L(\langle a\rangle S, (x)T, z)$		$\mathsf{Imp}_R((x)\langle a\rangle S, c)$

- A term, M, *freshly* introduces a name, if and only if M introduces this name, but none of its proper subterms. In other words, the name must not be free in a proper subterm of M, just on the top-level of M. Similarly for co-names.

Let us illustrate the definition just given and our implicit conventions for contexts with the following example: consider the $\supset_R$-rule again, this time annotated with terms.

$$\frac{x : B, \Gamma \triangleright M \triangleright \Delta, a : C}{\Gamma \triangleright \mathsf{Imp}_R((x)\langle a\rangle M, b) \triangleright \Delta, b : B \supset C} \supset_R$$

Now this rule introduces the (co-name,type) pair, $b : B \supset C$, in the conclusion, and consequently, b is a free co-name in $\mathsf{Imp}_R((x)\langle a\rangle M, b)$. In general, however, b could already be free in the subterm M, and thus the conclusion of this rule is of the form

$$\Gamma \triangleright \mathsf{Imp}_R((x)\langle a\rangle M, b) \triangleright \Delta \oplus b : B \supset C \ .$$

In case the term $\mathsf{Imp}_R((x)\langle a\rangle M, b)$ freshly introduces $b : B \supset C$, then the conclusion is of the form

$$\Gamma \triangleright \mathsf{Imp}_R((x)\langle a\rangle M, b) \triangleright \Delta \otimes b : B \supset C$$

where $b : B \supset C$ is not in Δ. The binders x and a cannot be part of the conclusion, and therefore the premise must be of the form

$$x : B \otimes \Gamma \triangleright M \triangleright \Delta \otimes a : C \ .$$

We add two new syntactic categories of terms, which are *not* proof annotations, but play an important rôle in the definition of substitution and in our strong normalisation proof.

$$\frac{}{x:B,\Gamma \triangleright \mathsf{Ax}(x,a) \triangleright \Delta, a:B}$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B}{x:\neg B,\Gamma \triangleright \mathsf{Not}_L(\langle a\rangle M, x) \triangleright \Delta}\ \neg_L \qquad \frac{x:B,\Gamma \triangleright M \triangleright \Delta}{\Gamma \triangleright \mathsf{Not}_R((x)M, a) \triangleright \Delta, a:\neg B}\ \neg_R$$

$$\frac{x:B_i,\Gamma \triangleright M \triangleright \Delta}{y:B_1\wedge B_2,\Gamma \triangleright \mathsf{And}^i_L((x)M, y) \triangleright \Delta}\ \wedge_{L_i} \qquad \frac{\Gamma \triangleright M \triangleright \Delta, a:B \qquad \Gamma \triangleright N \triangleright \Delta, b:C}{\Gamma \triangleright \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c) \triangleright \Delta, c:B\wedge C}\ \wedge_R$$

$$\frac{x:B,\Gamma \triangleright M \triangleright \Delta \qquad y:C,\Gamma \triangleright N \triangleright \Delta}{z:B\vee C,\Gamma \triangleright \mathsf{Or}_L((x)M, (y)N, z) \triangleright \Delta}\ \vee_L \qquad \frac{\Gamma \triangleright M \triangleright \Delta, a:B_i}{\Gamma \triangleright \mathsf{Or}^i_R(\langle a\rangle M, b) \triangleright \Delta, b:B_1\vee B_2}\ \vee_{R_i}$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B \qquad x:C,\Gamma \triangleright N \triangleright \Delta}{y:B\supset C,\Gamma \triangleright \mathsf{Imp}_L(\langle a\rangle M, (x)N, y) \triangleright \Delta}\ \supset_L \qquad \frac{x:B,\Gamma \triangleright M \triangleright \Delta, a:C}{\Gamma \triangleright \mathsf{Imp}_R((x)\langle a\rangle M, b) \triangleright \Delta, b:B\supset C}\ \supset_R$$

$$\frac{\Gamma_1 \triangleright M \triangleright \Delta_1, a:B \qquad x:B,\Gamma_2 \triangleright N \triangleright \Delta_2}{\Gamma_1,\Gamma_2 \triangleright \mathsf{Cut}(\langle a\rangle M, (x)N) \triangleright \Delta_1,\Delta_2}\ \text{Cut}$$

Figure 2.3: Term assignment for sequent proofs of the propositional fragment of classical logic.

Definition 2.2.3: Let M and N be well-typed terms, then $(x{:}B)M$ and $\langle a{:}C\rangle N$ are referred to as a *named term* and *co-named term*, respectively. More formally, we have the following two families of sets indexed by types:

$$\mathcal{T}_{(B)} \stackrel{\text{def}}{=} \Big\{ (x{:}B)M \;\Big|\; M \in \mathcal{T} \text{ with the typing judgement } x:B,\Gamma \triangleright M \triangleright \Delta \Big\}$$
$$\mathcal{T}_{\langle C\rangle} \stackrel{\text{def}}{=} \Big\{ \langle a{:}C\rangle N \;\Big|\; N \in \mathcal{T} \text{ with the typing judgement } \Gamma \triangleright N \triangleright \Delta, a:C \Big\}$$

We extend the notion of free names and free co-names to the named and co-named terms in the obvious way.

Now let us focus on term rewriting rules. One reason for introducing terms is that they greatly simplify the formalisation of the cut-reduction rules, most notably the rules for commuting cuts. In the sequent calculus given in Figure 2.2 there are 300 different cases of commuting cuts! To consider each case separately is clearly rather laborious, and unfortunately simplifications using sequent proofs seem difficult to obtain. For example Ungar [1992, Page 192] remarks on a cut-elimination procedure in a much simpler setting:

> Unfortunately, there are a large number of cases to consider and I have not been able to find a notation that enables more than a couple of these to be amalgamated.

Therefore traditional treatments of cut-elimination either omit these permutation rules entirely or present only a handful of cases; for example [Gentzen, 1935, Girard et al., 1989, Gallier, 1993]. This is unfortunate as a careful study of these rules sheds some light on the problems of non-termination of cut-elimination.

A substitution operation, which we shall introduce for terms, allows us to formalise the proof transformations necessary for commuting cuts in only twenty clauses. Clearly, this is an advantage of our use of terms. To give the definition of this substitution operation in a compact form, we assume the following convention, which is standard in term rewriting.

Convention 2.2.4 (Barendregt-style Naming Convention)**:** We assume that terms are equal up to alpha-conversion; that means rewriting bound names or co-names does not change terms. This conversion generates alpha-equivalence classes of terms. When writing terms we observe a Barendregt-style naming convention, which says that free and bound (co-)names are always distinct (cf. Convention 2.1.13 given by Barendregt [1981]). Thus we can view our terms as representatives of the alpha-equivalence classes and work with these terms in the naïve way.

Remark 2.2.5: An alternative to this convention avoiding the problems arising from alpha-conversions would be to use de Bruijn's nameless terms. This an elegant method for an implementation, however, at the expense of making our terms illegible for human readers. Another alternative is to define directly the alpha-equivalence classes of our terms using the meta-language provided by Gabbay and Pitts [1999]. Both methods seem less useful for our purposes, since they either complicate the definition of terms or the definition of substitution. Therefore we shall use the Barendregt-style naming convention throughout the thesis.

Before we formalise the substitution operation, let us give some intuition behind this operation. Consider the following sequent proof where we have omitted the term annotations and labels for brevity.

$$\pi_1\left\{\dfrac{\dfrac{\overline{B\vee C\vdash D\supset E, B\vee C}\;\star \qquad \dfrac{\overline{B\vee C, D\vdash E, B\vee C}\;\star}{B\vee C\vdash D\supset E, B\vee C}\,{\supset_R}}{(B\vee C)\vee(B\vee C)\vdash D\supset E, B\vee C}\,{\vee_L}\right. \qquad \left.\dfrac{\dfrac{\overline{B,F\vdash B,C} \qquad \overline{C,F\vdash B,C}}{B\vee C, F\vdash B,C}\,{\vee_L}\bullet}{B\vee C, F\wedge G\vdash B,C}\,{\wedge_{L_1}}\right\}\pi_2$$
$$\dfrac{\qquad (B\vee C)\vee(B\vee C)\vdash D\supset E, B\vee C \qquad\qquad B\vee C, F\wedge G\vdash B,C \qquad}{(B\vee C)\vee(B\vee C), F\wedge G\vdash D\supset E, B, C}\,\text{Cut}$$

The cut-formula, $B\vee C$, is neither introduced in the inference rule $\vee_L$ nor in $\wedge_{L_1}$. Therefore the cut is, by definition, a commuting cut. In π_1 the cut-formula is introduced in the axioms marked with a star, and in π_2, respectively, in the inference rule marked with a disc. Eliminating the cut in the proof above means to either permute the derivation π_2 to the places marked with a star and replace the corresponding axioms with π_2, or to permute π_1 and "cut it against" the inference rule marked with a disc. In the former case the derivation being permuted is duplicated.

We realise these operations at the term level with two symmetric forms of substitution, which we shall write as

$$P[x\!:\!B := \langle a\!:\!B\rangle Q] \quad \text{or} \quad S[b\!:\!B := (y\!:\!B)T]\;.$$

For brevity we shall usually omit the type annotations in the substitution operation, provided they are clear from the context.

Returning to our motivating example, assume that M and N are the terms corresponding to the subproofs π_1 and π_2, which thus have the typing judgements:

$$\begin{array}{c} x:(B{\vee}C){\vee}(B{\vee}C) \;\triangleright\; M \;\triangleright\; a:D{\supset}E, b:B{\vee}C \\ y:B{\vee}C, z:F{\wedge}G \;\triangleright\; N \;\triangleright\; c:B, d:C \end{array}$$

Consequently, we have the term annotation

$$x:(B{\vee}C){\vee}(B{\vee}C), z:F{\wedge}G \;\triangleright\; \mathsf{Cut}(\langle b\rangle M, (y)N) \;\triangleright\; a:D{\supset}E, c:B, d:C$$

for the conclusion of the example proof. The term $M[b := (y)N]$ denotes the following proof, where we have again omitted all term annotations and labels.

$$\dfrac{\dfrac{\dfrac{\overline{B,F\vdash D{\supset}E,B,C} \quad \overline{C,F\vdash D{\supset}E,B,C}}{B{\vee}C,F\vdash D{\supset}E,B,C}\,{\vee}_L}{B{\vee}C,F{\wedge}G\vdash D{\supset}E,B,C}\,{\wedge}_{L_1} \quad \dfrac{\dfrac{\dfrac{\overline{B,F,D\vdash E,B,C} \quad \overline{C,F,D\vdash E,B,C}}{B{\vee}C,F,D\vdash E,B,C}\,{\vee}_L}{B{\vee}C,F{\wedge}G,D\vdash E,B,C}\,{\wedge}_{L_1}}{B{\vee}C,F{\wedge}G\vdash D{\supset}E,B,C}\,{\supset}_R}{(B{\vee}C){\vee}(B{\vee}C),F{\wedge}G\vdash D{\supset}E,B,C}\,{\vee}_L$$

Similarly, the symmetric case $N[y := \langle b\rangle M]$ denotes the proof

$$\dfrac{\dfrac{\dfrac{\overline{B{\vee}C\vdash D{\supset}E,B{\vee}C} \quad \dfrac{\overline{B{\vee}C,D\vdash E,B{\vee}C}}{B{\vee}C\vdash D{\supset}E,B{\vee}C}\,{\supset}_R}{(B{\vee}C){\vee}(B{\vee}C)\vdash D{\supset}E,B{\vee}C}\,{\vee}_L \quad \dfrac{\overline{B,F\vdash B,C} \quad \overline{C,F\vdash B,C}}{B{\vee}C,F\vdash B,C}\,{\vee}_L}{(B{\vee}C){\vee}(B{\vee}C),F\vdash D{\supset}E,B,C}\,\text{Cut}}{(B{\vee}C){\vee}(B{\vee}C),E{\wedge}F\vdash D{\supset}E,B,C}\,{\wedge}_{L_1}$$

As we have seen, commuting cuts need to permute, or “jump”, to the places where the cut-formula is a main formula. At the level of terms this means the cuts need to be permuted to every subterm that introduces the cut-formula. Therefore, whenever a substitution is next to a term in which the cut-formula is introduced, the substitution becomes an instance of the cut-term constructor. In the following two examples we shall write $[\sigma]$ and $[\tau]$ for the substitutions $[c := (x)P]$ and $[x := \langle b\rangle Q]$, respectively.

$$\begin{array}{rcl} \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)[\sigma] & = & \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle\, M[\sigma], \langle b\rangle\, N[\sigma], c), (x)P) \\ \mathsf{Imp}_L(\langle a\rangle M, (y)N, x)[\tau] & = & \mathsf{Cut}(\langle b\rangle Q, (x)\mathsf{Imp}_L(\langle a\rangle\, M[\tau], (y)\, N[\tau], x)) \end{array}$$

In the first example the formula labelled with c is the main formula and in the second the formula labelled with x is the main formula. So in both cases the substitutions “expand” to cuts, and in addition the substitutions are pushed inside the subterms. This is because there might be several occurrences of c and x (both labels need not have been freshly introduced). An exception applies to axioms, where the substitution is defined differently.

$$\begin{array}{rcl}\mathsf{Ax}(x,a)[x := \langle b\rangle P] & = & P[b\mapsto a]\\ \mathsf{Ax}(x,a)[a := (y)Q] & = & Q[y\mapsto x]\end{array}$$

Recall that $P[b\mapsto a]$ stands for the term P where every free occurrence of the co-name b is rewritten to a (similarly $Q[y\mapsto x]$). We are left with the cases where the name or co-name that is being substituted for is not a label of the main formula. In these cases the substitutions are pushed inside the subterms or vanish in case of the axioms. Suppose the substitution $[\sigma]$ is *not* of the form $[z := \ldots]$ and $[a := \ldots]$, then we have for example:

$$\begin{array}{rcl}\mathsf{Or}_L((x)M,(y)N,z)[\sigma] & = & \mathsf{Or}_L((x)\ M[\sigma],(y)\ N[\sigma],z)\\ \mathsf{Ax}(z,a)[\sigma] & = & \mathsf{Ax}(z,a)\end{array}$$

The complete definition of the substitution operation is given in Figure 2.4. We do not need to worry about inserting contraction rules when a term is duplicated, since our contexts are sets of labelled formulae, and thus contractions are made implicitly. Another simplification is due to our use of the Barendregt-style naming convention, because we do not need to worry about possible capture of free names or co-names. Let us now introduce some useful terminology for substitutions.

Terminology 2.2.6: We shall use the expression $[\sigma]$ to range over substitutions of the form $[x:B := \langle a:B\rangle\, Q]$ and $[b:B := (y:B)\, T]$. In the first case $x:B$ is the *domain* of $[\sigma]$, written as $dom([\sigma])$, and the co-named term $\langle a{:}B\rangle Q$ is the *co-domain* of $[\sigma]$, written as $codom([\sigma])$. Similarly in the second case.

Next, let us focus on the cut-reductions for logical cuts. Consider an instance of an $\wedge_R/\wedge_{L_1}$-cut for which a naïve definition of reduction might be as follows.

$$\mathsf{Cut}(\langle c\rangle\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c),(y)\mathsf{And}^1_L((x)P,y)) \longrightarrow \mathsf{Cut}(\langle a\rangle M,(x)P)$$

However, there is a problem with this reduction rule. In our sequent calculus the structural rules are *implicit*: this means that not only does the calculus have fewer inference rules, but more importantly, we have a very convenient way of defining substitution (we do not need explicit contractions when a term is duplicated). On the other hand, there is a subtle side-effect of this design decision; consider the following instance of the redex[3] above

$$\dfrac{\dfrac{\Gamma_1 \triangleright M \triangleright \Delta_1, \boxed{c:B\wedge C}, a:B \quad \Gamma_1 \triangleright N \triangleright \Delta_1, b:C}{\Gamma_1 \triangleright \mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c) \triangleright \Delta_1, c:B\wedge C}\wedge_R \quad \dfrac{x:B,\Gamma_2 \triangleright P \triangleright \Delta_2}{y:B\wedge C,\Gamma_2 \triangleright \mathsf{And}^1_L((x)P,y) \triangleright \Delta_2}\wedge_{L_1}}{\Gamma_1,\Gamma_2 \triangleright \mathsf{Cut}(\langle c\rangle\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c),(y)\mathsf{And}^1_L((x)P,y)) \triangleright \Delta_1,\Delta_2}\mathsf{Cut}$$

where c is a free co-name in M (shaded box). Our naïve reduction rule would yield

$$\dfrac{\Gamma_1 \triangleright M \triangleright \Delta_1, c:B\wedge C, a:B \quad x:B,\Gamma_2 \triangleright P \triangleright \Delta_2}{\Gamma_1,\Gamma_2 \triangleright \mathsf{Cut}(\langle a\rangle M,(x)P) \triangleright \Delta_1,\Delta_2, \boxed{c:B\wedge C}}\mathsf{Cut}\ .$$

Unfortunately, c has now become free in the conclusion! The problem here is that the

[3] *Redex* refers to a place in a term where a reduction may occur.

$$
\begin{array}{rcl}
\mathsf{Ax}(x,c)[c := (y)P] & \stackrel{\text{def}}{=} & P[y \mapsto x] \\
\mathsf{Ax}(y,a)[y := \langle c\rangle P] & \stackrel{\text{def}}{=} & P[c \mapsto a] \\
\mathsf{Not}_R((x)M,a)[a := (y)P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)M[a := (y)P],a),(y)P) \\
\mathsf{Not}_L(\langle a\rangle M,x)[x := \langle c\rangle P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P,(x)\mathsf{Not}_L(\langle a\rangle M[x := \langle c\rangle P],x)) \\
\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)[c := (y)P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M[c := (y)P],\langle b\rangle N[c := (y)P],c),(y)P) \\
\mathsf{And}^i_L((x)M,y)[y := \langle c\rangle P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{And}^i_L((x)M[y := \langle c\rangle P],y)) \\
\mathsf{Or}^i_R(\langle a\rangle M,c)[c := (y)P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle \mathsf{Or}^i_R(\langle a\rangle M[c := (y)P],c),(y)P) \\
\mathsf{Or}_L((x)M,(y)N,z)[z := \langle c\rangle P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P,(z)\mathsf{Or}_L((x)M[z := \langle c\rangle P],(y)N[z := \langle c\rangle P],z)) \\
\mathsf{Imp}_R((x)\langle a\rangle M,b)[b := (y)P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M[b := (y)P],b),(y)P) \\
\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)[y := \langle c\rangle P] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{Imp}_L(\langle a\rangle M[y := \langle c\rangle P],(x)N[y := \langle c\rangle P],y))
\end{array}
$$

Otherwise:

$$
\begin{array}{rcl}
\mathsf{Ax}(x,a)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Ax}(x,a) \\
\mathsf{Cut}(\langle a\rangle M,(x)N)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle\ M[\sigma],(x)\ N[\sigma]) \\
\mathsf{Not}_R((x)M,a)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Not}_R((x)\ M[\sigma],a) \\
\mathsf{Not}_L(\langle a\rangle M,x)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Not}_L(\langle a\rangle\ M[\sigma],x) \\
\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{And}_R(\langle a\rangle\ M[\sigma],\langle b\rangle\ N[\sigma],c) \\
\mathsf{And}^i_L((x)M,y)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{And}^i_L((x)\ M[\sigma],y) \\
\mathsf{Or}^i_R(\langle a\rangle M,b)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Or}^i_R(\langle a\rangle\ M[\sigma],b) \\
\mathsf{Or}_L((x)M,(y)N,z)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Or}_L((x)\ M[\sigma],(y)\ N[\sigma],z) \\
\mathsf{Imp}_R((x)\langle a\rangle M,b)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Imp}_R((x)\langle a\rangle\ M[\sigma],b) \\
\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Imp}_L(\langle a\rangle\ M[\sigma],(x)\ N[\sigma],y)
\end{array}
$$

Figure 2.4: Proof substitution.

original proof, despite first appearances, is *not* a logical cut, but is in fact a commuting cut, and should really be reduced to

$$\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)[c := (y)\mathsf{And}^1_L((x)P, y)] \ .$$

Consequently, we ensure that logical reduction rules apply only where the cut-formula is *freshly* introduced. Figure 2.5 gives the cut-reductions for logical cuts, denoted by $\xrightarrow{l}$, and commuting cuts, denoted by $\xrightarrow{c}$. We automatically assume that the reductions are closed under context formation, which is a standard convention in term rewriting. For the cut-reduction $\supset_R/\supset_L$ (Rule 4) there are a few remarks worth making.

Remark 2.2.7:

- First, there are two ways to reduce a cut-rule having an implication as the cut-formula. Consider the following proof-fragment

$$\dfrac{\dfrac{x:B \triangleright M \triangleright a:C}{\triangleright \mathsf{Imp}_R((x)\langle a\rangle M, b) \triangleright b:B\supset C}\supset_R \quad \dfrac{\triangleright N \triangleright c:B \qquad y:C \triangleright P \triangleright}{z:B\supset C \triangleright \mathsf{Imp}_L(\langle c\rangle N, (y)P, z) \triangleright}\supset_L}{\triangleright \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle N, (y)P, z)) \triangleright}\mathrm{Cut}$$

  which can be reduced either to

$$\dfrac{\dfrac{\triangleright N \triangleright c:B \quad x:B \triangleright M \triangleright a:C}{\triangleright \mathsf{Cut}(\langle c\rangle N, (x)M) \triangleright a:C}\mathrm{Cut} \qquad y:C \triangleright P \triangleright}{\triangleright \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle N, (x)M), (y)P) \triangleright}\mathrm{Cut}$$

  or to

$$\dfrac{\triangleright N \triangleright c:B \quad \dfrac{x:B \triangleright M \triangleright a:C \quad y:C \triangleright P \triangleright}{x:B \triangleright \mathsf{Cut}(\langle a\rangle M, (y)P) \triangleright}\mathrm{Cut}}{\triangleright \mathsf{Cut}(\langle c\rangle N, (x)\mathsf{Cut}(\langle a\rangle M, (y)P)) \triangleright}\mathrm{Cut}$$

  Therefore we have included two reductions, which entails that our cut-elimination procedure is non-deterministic.

- Second, special care needs to be taken so that there is no clash between bound and free (co-)names. The term $\mathsf{Imp}_R((x)\langle a\rangle M, b)$ binds x and a simultaneously; however in the reducts the cut-rules bind x and a, separately. Therefore in the first reduction rule we need to ensure that a is not a free co-name in $\langle c\rangle N$ and in the second rule that x is not a free name in $(y)P$. This can always be achieved by renaming a and x appropriately: they are binders in $\mathsf{Imp}_R((x)\langle a\rangle M, b)$. We assume that the renaming is done implicitly in the cut-elimination procedure.

We are now ready to formulate the cut-elimination procedure, which we shall define in terms of a reduction system.

Definition 2.2.8 (Cut-Elimination Procedure)**:**
The cut-elimination procedure $(\mathcal{T}, \xrightarrow{cut})$ is a reduction system where:

- $\mathcal{T}$ is the set of well-typed terms, and
- $\xrightarrow{cut}$ consists of the reductions for logical cuts and commuting cuts; that is

$$\xrightarrow{cut} \stackrel{\mathrm{def}}{=} \xrightarrow{l} \cup \xrightarrow{c} \ .$$

Logical Cuts $(i = 1, 2)$:

1. $\mathsf{Cut}(\langle a\rangle\mathsf{Not}_R((x)M,a),(y)\mathsf{Not}_L(\langle b\rangle N,y)) \xrightarrow{l} \mathsf{Cut}(\langle b\rangle N,(x)M)$

   if $\mathsf{Not}_R((x)M,a)$ and $\mathsf{Not}_L(\langle b\rangle N,y)$ freshly introduce a and y

2. $\mathsf{Cut}(\langle b\rangle\mathsf{And}_R(\langle a_1\rangle M_1,\langle a_2\rangle M_2,b),(y)\mathsf{And}^i_L((x)N,y)) \xrightarrow{l} \mathsf{Cut}(\langle a_i\rangle M_i,(x)N)$

   if $\mathsf{And}_R(\langle a_1\rangle M_1,\langle a_2\rangle M_2,b)$ and $\mathsf{And}^i_L((x)N,y)$ freshly introduce b and y

3. $\mathsf{Cut}(\langle b\rangle\mathsf{Or}^i_R(\langle a\rangle M,b),(y)\mathsf{Or}_L((x_1)N_1,(x_2)N_2,y)) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle M,(x_i)N_i)$

   if $\mathsf{Or}_L((x_1)N_1,(x_2)N_2,y)$ and $\mathsf{Or}^i_R(\langle a\rangle M,b)$ freshly introduce y and b

4. $\mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle M,b),(z)\mathsf{Imp}_L(\langle c\rangle N,(y)P,z))$
   $\xrightarrow{l} \mathsf{Cut}(\langle a\rangle\mathsf{Cut}(\langle c\rangle N,(x)M),(y)P)$ or
   $\xrightarrow{l} \mathsf{Cut}(\langle c\rangle N,(x)\mathsf{Cut}(\langle a\rangle M,(y)P))$

   if $\mathsf{Imp}_L(\langle c\rangle N,(y)P,z)$ and $\mathsf{Imp}_R((x)\langle a\rangle M,b)$ freshly introduce z and b

5. $\mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Ax}(x,b)) \xrightarrow{l} M[a\mapsto b]$

   if M freshly introduces a

6. $\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(y,a),(x)M) \xrightarrow{l} M[x\mapsto y]$

   if M freshly introduces x

Commuting Cuts:

7. $\mathsf{Cut}(\langle a\rangle M,(x)N)$
   $\xrightarrow{c} M[a := (x)N]$ if M does not freshly introduce a, or
   $\xrightarrow{c} N[x := \langle a\rangle M]$ if N does not freshly introduce x

Figure 2.5: Cut-reductions for logical cuts and commuting cuts.

Notice that $\xrightarrow{l}$ and $\xrightarrow{c}$ are, by assumption, closed under context formation. The *completeness* of $\xrightarrow{cut}$ is simply the fact, obvious by inspection, that every term beginning with a cut matches at least one left-hand side of the reduction rules. So each irreducible term, also called a normal form, is cut-free.

We should prove that the cut-reductions satisfy the subject reduction property, which states that well-typed terms reduce to well-typed terms. An auxiliary lemma required in this proof follows.

Lemma 2.2.9 (Weakening and Contraction Lemma)**:**

- Suppose Γ, Γ', Δ and Δ' are contexts. If M is a term with the typing judgement $\Gamma \triangleright M \triangleright \Delta$, then also $\Gamma', \Gamma \triangleright M \triangleright \Delta, \Delta'$ is a typing judgement for M, provided the context convention is observed.
- If M is a term with the typing judgement $x : B, y : B, \Gamma \triangleright M \triangleright \Delta$, then for the term $M[y \mapsto x]$ we have a typing judgement $x : B, \Gamma \triangleright M[y \mapsto x] \triangleright \Delta$. If M is a term with the typing judgement $\Gamma \triangleright M \triangleright \Delta, a : B, b : B$, then for the term $M[b \mapsto a]$ we have a typing judgement $\Gamma \triangleright M[b \mapsto a] \triangleright \Delta, a : B$.

Proof. All by trivial induction on the structure of M. ☐

Proposition 2.2.10 (Subject Reduction)**:**
Suppose M is a term with the typing judgement $\Gamma \triangleright M \triangleright \Delta$ and $M \xrightarrow{cut} N$, then N is a term with the typing judgement $\Gamma \triangleright N \triangleright \Delta$.

Proof. By inspection of the reduction rules (see Page 141). ☐

Let us mention that the Barendregt-style naming convention is indispensable for strong normalisation of $\xrightarrow{cut}$. If we do not observe this convention, we lose strong normalisation. To give an example of a non-terminating reduction sequence we need the following lemma.

Lemma 2.2.11: For all terms $M \in \mathcal{T}$ we have

(i) $M[x := \langle a\rangle \mathsf{Ax}(y, a)] \xrightarrow{cut}{}^{*} M[x \mapsto y]$

(ii) $M[a := (x)\mathsf{Ax}(x, b)] \xrightarrow{cut}{}^{*} M[a \mapsto b]$

Proof. By routine induction on the structure of M. ☐

Example 2.2.12: We start with the term $\mathsf{Not}_R((x)M, a)[a := (y)\mathsf{Ax}(y, a)]$ for which we assume that there are several free occurrences of a in M; that means $\mathsf{Not}_R((x)M, a)$ does not freshly introduce a. We apply our definition of substitution (see Figure 2.4) so as to yield

$$\mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)M[a := (y)\mathsf{Ax}(y, a)], a), (y)\mathsf{Ax}(y, a))\ . \tag{2.4}$$

According to the lemma just given this term reduces, assuming that a is free in M, by some reductions to

$$\mathsf{Cut}(\langle a\rangle\mathsf{Not}_R((x)M[a\mapsto a],a),(y)\mathsf{Ax}(y,a))\ .$$

Clearly, $M[a\mapsto a]$ is equivalent to M, which contains some free occurrences of a though. Therefore we cannot apply a logical rule, but have to apply a reduction rule for commuting cuts. This gives us

$$\mathsf{Not}_R((x)M,a)[a:=(y)\mathsf{Ax}(y,a)]$$

and we are "back" where we started from. A closer analysis of this erroneous reduction sequence reveals that in Step (2.4) the co-name a became bound by the cut when we pushed the substitution inside the term $\mathsf{Not}_R((x)M,a)$. To avoid this capture of the free occurrence of a in $(y)\mathsf{Ax}(y,a)$, we have to rename the corresponding binder. Accordingly, we should have written

$$\begin{array}{ll} & \mathsf{Not}_R((x)M,a)[a:=(y)\mathsf{Ax}(y,a)] \\ \equiv & \mathsf{Not}_R((x)M[a\mapsto a'],a')[a':=(y)\mathsf{Ax}(y,a)] \\ = & \mathsf{Cut}(\langle a'\rangle\mathsf{Not}_R((x)M[a\mapsto a'][a':=(y)\mathsf{Ax}(y,a)],a'),(y)\mathsf{Ax}(y,a)) \end{array}$$

assuming a' does not occur freely in M. Now we may apply Lemma 2.2.11 and receive performing some reductions

$$\mathsf{Cut}(\langle a'\rangle\mathsf{Not}_R((x)M[a\mapsto a'][a'\mapsto a],a'),(y)\mathsf{Ax}(y,a))\ .$$

Because $\mathsf{Not}_R((x)M[a\mapsto a'][a'\mapsto a],a')$ freshly introduces a', we can now apply a logical reduction and obtain

$$\mathsf{Not}_R((x)M[a\mapsto a'][a'\mapsto a],a')[a'\mapsto a]\ .$$

This term is equivalent to $\mathsf{Not}_R((x)M,a)$—the term we expected given Lemma 2.2.11.

2.3 Proof of Strong Normalisation

In this section we shall give a direct proof of strong normalisation for the reduction system $(\mathcal{T},\xrightarrow{cut})$. The proof adapts the technique of the symmetric reducibility candidates[4] developed by Barbanera and Berardi [1994]. Unfortunately, we cannot apply this technique directly to prove strong normalisation for $(\mathcal{T},\xrightarrow{cut})$, because in order to strengthen an induction hypothesis, we need the following property

$$M[x:=\langle a\rangle P][b:=(y)Q]\equiv M[b:=(y)Q][x:=\langle a\rangle P]$$

for b not free in $\langle a\rangle P$ and x not free in $(y)Q$. However this does *not* hold for the substitu-

[4]It seems this type of candidates was first used by Girard [1987a] for proving strong normalisation of proof-nets.

tion operation defined in Figure 2.4. In effect, "independent" substitutions, in general, do not commute! The (only) problematic case is where M is of the form $\mathsf{Ax}(x, b)$:

$$\begin{aligned}\mathsf{Ax}(x,b)[x := \langle a\rangle P][b := (y)Q] &= P[a \mapsto b][b := (y)Q]\ \text{, but}\\ \mathsf{Ax}(x,b)[b := (y)Q][x := \langle a\rangle P] &= Q[y \mapsto x][x := \langle a\rangle P]\,.\end{aligned}$$

Clearly, there is no reason for the two resultant terms to be equal. To remedy this situation, we shall define an auxiliary cut-reduction system, $(\mathcal{T}, \xrightarrow{aux})$, which has a more subtle definition of substitution including two special clauses to handle the problematic example above. Intuitively, we should expect

$$\left.\begin{array}{l}\mathsf{Ax}(x,b)[x := \langle a\rangle P][b := (y)Q]\\ \mathsf{Ax}(x,b)[b := (y)Q][x := \langle a\rangle P]\end{array}\right\} = \mathsf{Cut}(\langle a\rangle P, (y)Q)\ .$$

The new substitution operation, written as $M\{a := (x)N\}$ and $N\{x := \langle a\rangle M\}$, is given in Figure 2.6. We apply the same terminology as for substitutions of the form $[\sigma]$ (see Terminology 2.2.6). For $\{\_\}$ we have the following substitution lemma.

Lemma 2.3.1 (Substitution Lemma)**:**
For all $M \in \mathcal{T}$ and two arbitrary proof substitutions, $\{\sigma\}$ and $\{\tau\}$, such that $dom(\{\sigma\})$ is not free in $codom(\{\tau\})$, we have

$$M\{\sigma\}\{\tau\} \equiv M\{\tau\}\{\sigma\{\tau\}\}\ .$$

Proof. By induction on the structure of M (see Page 141). □

Since we changed the substitution operation we need to change the reduction rule for commuting cuts. The modified rule is as follows.

$$\begin{array}{llll}\mathsf{Cut}(\langle a\rangle M, (x)N) & \xrightarrow{c'} & M\{a := (x)N\} & \text{if } M \text{ does not freshly introduce } a\text{, or}\\ & \xrightarrow{c'} & N\{x := \langle a\rangle M\} & \text{if } N \text{ does not freshly introduce } x.\end{array}$$

The auxiliary cut-elimination procedure is then

Definition 2.3.2 (Auxiliary Cut-Elimination Procedure)**:**
The auxiliary cut-elimination procedure, $(\mathcal{T}, \xrightarrow{aux})$, is a reduction system where:

- $\mathcal{T}$ is the set of well-typed terms, and
- $\xrightarrow{aux}$ consists of the rules for the logical cuts and the modified reduction for commuting cuts; that is

$$\xrightarrow{aux} \stackrel{\text{def}}{=} \xrightarrow{l} \cup \xrightarrow{c'}\ .$$

We have to verify that $\xrightarrow{aux}$ satisfies the subject reduction property. However, given the proof of Proposition 2.2.10, this is a routine matter, and so we omit the details. Let us

$$
\begin{aligned}
\mathsf{Ax}(x,c)\{c:=(y)P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle\mathsf{Ax}(x,c),(y)P)\\
\mathsf{Ax}(y,a)\{y:=\langle c\rangle P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{Ax}(y,a))\\
\mathsf{Not}_R((x)M,a)\{a:=(y)P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle a\rangle\mathsf{Not}_R((x)M\{a:=(y)P\},a),(y)P)\\
\mathsf{Not}_L(\langle a\rangle M,x)\{x:=\langle c\rangle P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle P,(x)\mathsf{Not}_L(\langle a\rangle M\{x:=\langle c\rangle P\},x))\\
\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)\{c:=(y)P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle\mathsf{And}_R(\langle a\rangle M\{c:=(y)P\},\langle b\rangle N\{c:=(y)P\},c),(y)P)\\
\mathsf{And}_L^i((x)M,y)\{y:=\langle c\rangle P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{And}_L^i((x)M\{y:=\langle c\rangle P\},y))\\
\mathsf{Or}_R^i(\langle a\rangle M,c)\{c:=(y)P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle\mathsf{Or}_R^i(\langle a\rangle M\{c:=(y)P\},c),(y)P)\\
\mathsf{Or}_L((x)M,(y)N,z)\{z:=\langle c\rangle P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle P,(z)\mathsf{Or}_L((x)M\{z:=\langle c\rangle P\},(y)N\{z:=\langle c\rangle P\},z))\\
\mathsf{Imp}_R((x)\langle a\rangle M,b)\{b:=(y)P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle M\{b:=(y)P\},b),(y)P)\\
\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)\{y:=\langle c\rangle P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{Imp}_L(\langle a\rangle M\{y:=\langle c\rangle P\},(x)N\{y:=\langle c\rangle P\},y))
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)M)\{x:=\langle b\rangle P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle b\rangle P,(y)M\{x:=\langle b\rangle P\})\\
\mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Ax}(x,b))\{b:=(y)P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle a\rangle M\{b:=(y)P\},(y)P)
\end{aligned}
$$

Otherwise:

$$
\begin{aligned}
\mathsf{Ax}(x,a)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Ax}(x,a)\\
\mathsf{Cut}(\langle a\rangle M,(x)N)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle a\rangle\ M\{\sigma\},(x)\ N\{\sigma\})\\
\mathsf{Not}_R((x)M,a)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Not}_R((x)\ M\{\sigma\},a)\\
\mathsf{Not}_L(\langle a\rangle M,x)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Not}_L(\langle a\rangle\ M\{\sigma\},x)\\
\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{And}_R(\langle a\rangle\ M\{\sigma\},\langle b\rangle\ N\{\sigma\},c)\\
\mathsf{And}_L^i((x)M,y)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{And}_L^i((x)\ M\{\sigma\},y)\\
\mathsf{Or}_R^i(\langle a\rangle M,b)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Or}_R^i(\langle a\rangle\ M\{\sigma\},b)\\
\mathsf{Or}_L((x)M,(y)N,z)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Or}_L((x)\ M\{\sigma\},(y)\ N\{\sigma\},z)\\
\mathsf{Imp}_R((x)\langle a\rangle M,b)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Imp}_R((x)\langle a\rangle\ M\{\sigma\},b)\\
\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)\{\sigma\} &\stackrel{\text{def}}{=} \mathsf{Imp}_L(\langle a\rangle\ M\{\sigma\},(x)\ N\{\sigma\},y)
\end{aligned}
$$

Figure 2.6: Auxiliary proof substitution.

outline how we shall proceed in our proof of strong normalisation for $(\mathcal{T}, \xrightarrow{aux})$.

(1) Define the sets of candidates over types using a fixed point construction (Definition 2.3.4).

(2) Prove that candidates are closed under reduction (Lemma 2.3.12).

(3) Show that a named or co-named term in a candidate implies strong normalisation for the corresponding term (Lemma 2.3.13).

(4) Extend the notion of safe substitution to simultaneous substitutions (Definition 2.3.15).

(5) Prove that all terms are strongly normalising (Theorem 2.3.19).

Finally, we shall show that every $\xrightarrow{cut}$-reduction maps onto a series of $\xrightarrow{aux}$-reductions and thus prove that $(\mathcal{T}, \xrightarrow{cut})$ is strongly normalising, too.

First, we shall define for every type two candidates, written as $[\![\langle B\rangle]\!]$ and $[\![(B)]\!]$. These candidates are subsets of named or co-named terms; i.e., $[\![\langle B\rangle]\!] \subseteq \mathcal{T}_{\langle B\rangle}$ and $[\![(B)]\!] \subseteq \mathcal{T}_{(B)}$. Whereas traditional notions of candidates are defined by a simple induction over types, our candidates are inductively defined over types, but also include fixed point operations. Before we give the definition of the candidates, we shall introduce some set operators, which fix certain closure properties for the candidates. Each of the operators is defined over sets of (co-)named terms having a specific term constructor at the top-level.

Definition 2.3.3 (Set Operators)**:**

$$\textsc{axioms}_{(B)} \stackrel{\text{def}}{=} \left\{ (x{:}B)\mathsf{Ax}(y,b) \;\middle|\; (x{:}B)\mathsf{Ax}(y,b) \in \mathcal{T}_{(B)} \right\}$$

$$\textsc{axioms}_{\langle B\rangle} \stackrel{\text{def}}{=} \left\{ \langle a{:}B\rangle\mathsf{Ax}(y,b) \;\middle|\; \langle a{:}B\rangle\mathsf{Ax}(y,b) \in \mathcal{T}_{\langle B\rangle} \right\}$$

Note that x can be equal to y, and a to b. Figure 2.7 gives the set operators which correspond to the other term constructors. Additionally we have

$$\textsc{binding}_{(B)}(X) \stackrel{\text{def}}{=} \left\{ (x{:}B)M \;\middle|\; \text{for all } \langle a{:}B\rangle P \in X \,.\, M\{x := \langle a{:}B\rangle P\} \in SN_{aux} \right\}$$

$$\textsc{binding}_{\langle B\rangle}(Y) \stackrel{\text{def}}{=} \left\{ \langle a{:}B\rangle M \;\middle|\; \text{for all } (x{:}B) P \in Y \,.\, M\{a := (x{:}B)P\} \in SN_{aux} \right\}$$

The set operators AXIOMS and the set operators given in Figure 2.7 correspond to the properties we need to prove that a logical cut is strongly normalising; BINDING is sufficient to show strong normalisation of a commuting cut. For the definition of the candidates we use fixed points of increasing set operators. A set operator, *op*, is said to be:

increasing, if and only if $S \subseteq S' \Rightarrow op(S) \subseteq op(S')$, and

decreasing, if and only if $S \subseteq S' \Rightarrow op(S) \supseteq op(S')$.

$$\textsc{NotRight}_{\langle\neg B\rangle}(X) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} \langle a{:}\neg B\rangle\mathsf{Not}_R((x{:}B)M,a) \mid \\ \qquad \mathsf{Not}_R((x{:}B)M,a) \text{ freshly introduces } a, \\ \qquad (x{:}B)M \in X \end{array} \right\}$$

$$\textsc{NotLeft}_{(\neg B)}(X) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} (x{:}\neg B)\mathsf{Not}_L(\langle a{:}B\rangle M,x) \mid \\ \qquad \mathsf{Not}_L(\langle a{:}B\rangle M,x) \text{ freshly introduces } x, \\ \qquad \langle a{:}B\rangle M \in X \end{array} \right\}$$

$$\textsc{AndRight}_{\langle B\wedge C\rangle}(X,Y) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} \langle c{:}B\wedge C\rangle\mathsf{And}_R(\langle a{:}B\rangle M,\langle b{:}C\rangle N,c) \mid \\ \qquad \mathsf{And}_R(\langle a{:}B\rangle M,\langle b{:}C\rangle N,c) \text{ freshly introduces } c, \\ \qquad \langle a{:}B\rangle M \in X, \\ \qquad \langle b{:}C\rangle N \in Y \end{array} \right\}$$

$$\textsc{AndLeft}^i_{(B_1\wedge B_2)}(X) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} (y{:}B_1\wedge B_2)\mathsf{And}^i_L((x{:}B_i)M,y) \mid \\ \qquad \mathsf{And}^i_L((x{:}B_i)M,y) \text{ freshly introduces } y, \\ \qquad (x{:}B_i)M \in X \end{array} \right\}$$

$$\textsc{OrRight}^i_{\langle B_1\vee B_2\rangle}(X) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} \langle b{:}B_1\vee B_2\rangle\mathsf{Or}^i_R(\langle a{:}B_i\rangle M,b) \mid \\ \qquad \mathsf{Or}^i_R(\langle a{:}B_i\rangle M,b) \text{ freshly introduces } b, \\ \qquad \langle a{:}B_i\rangle M \in X \end{array} \right\}$$

$$\textsc{OrLeft}_{(B\vee C)}(X,Y) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} (z{:}B\vee C)\mathsf{Or}_L((x{:}B)M,(y{:}C)N,z) \mid \\ \qquad \mathsf{Or}_L((x{:}B)M,(y{:}C)N,z) \text{ freshly introduces } z, \\ \qquad (x{:}B)M \in X, \\ \qquad (y{:}C)N \in Y \end{array} \right\}$$

$$\textsc{ImpRight}_{\langle B\supset C\rangle}(X,Y) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} \langle b{:}B\supset C\rangle\mathsf{Imp}_R((x{:}B)\langle a{:}C\rangle M,b) \mid \\ \qquad \mathsf{Imp}_R((x{:}B)\langle a{:}C\rangle M,b) \text{ freshly introduces } b, \\ \qquad \text{for all } (z{:}C)P \in Y\ . \\ \qquad\qquad\qquad (x{:}B)M\{a := (z)P\} \in X, \\ \qquad \text{for all } \langle c{:}B\rangle Q \in X\ . \\ \qquad\qquad\qquad \langle a{:}C\rangle M\{x := \langle c\rangle Q\} \in Y \end{array} \right\}$$

$$\textsc{ImpLeft}_{(B\supset C)}(X,Y) \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} (y{:}B\supset C)\mathsf{Imp}_L(\langle a{:}B\rangle M,(x{:}C)N,y) \mid \\ \qquad \mathsf{Imp}_L(\langle a{:}B\rangle M,(x{:}C)N,y) \text{ freshly introduces } y, \\ \qquad \langle a{:}B\rangle M \in X, \\ \qquad (x{:}C)N \in Y \end{array} \right\}$$

Figure 2.7: Definition of the set operators for propositional connectives.

We are now ready to define the set operator NEG and the candidates.

Definition 2.3.4 (Reducibility Candidates)**:**
The mutually recursive definition over types for NEG and the candidates is as follows:

$\mathbf{NEG}_{\langle B\rangle}$:

$$\left.\begin{array}{lcl} \mathrm{NEG}_{\langle A\rangle}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{\langle \neg C\rangle}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{\langle C\wedge D\rangle}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{\langle C_1\vee C_2\rangle}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{\langle C\supset D\rangle}(X) & \stackrel{\text{def}}{=} \end{array}\right\} \begin{array}{l} \mathrm{AXIOMS}_{\langle B\rangle} \cup \\ \quad \mathrm{BINDING}_{\langle B\rangle}(X) \cup \end{array} \left\{\begin{array}{l} - \\ \mathrm{NOTRIGHT}_{\langle \neg C\rangle}(\llbracket (C)\rrbracket) \\ \mathrm{ANDRIGHT}_{\langle C\wedge D\rangle}(\llbracket \langle C\rangle\rrbracket, \llbracket \langle D\rangle\rrbracket) \\ \bigcup_{i=1,2} \mathrm{ORRIGHT}^{i}_{\langle C_1\vee C_2\rangle}(\llbracket \langle C_i\rangle\rrbracket) \\ \mathrm{IMPRIGHT}_{\langle C\supset D\rangle}(\llbracket (C)\rrbracket, \llbracket \langle D\rangle\rrbracket) \end{array}\right.$$

$\mathbf{NEG}_{(B)}$:

$$\left.\begin{array}{lcl} \mathrm{NEG}_{(A)}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{(\neg C)}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{(C_1\wedge C_2)}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{(C\vee D)}(X) & \stackrel{\text{def}}{=} \\ \mathrm{NEG}_{(C\supset D)}(X) & \stackrel{\text{def}}{=} \end{array}\right\} \begin{array}{l} \mathrm{AXIOMS}_{(B)} \cup \\ \quad \mathrm{BINDING}_{(B)}(X) \cup \end{array} \left\{\begin{array}{l} - \\ \mathrm{NOTLEFT}_{(\neg C)}(\llbracket \langle C\rangle\rrbracket) \\ \bigcup_{i=1,2} \mathrm{ANDLEFT}^{i}_{(C_1\wedge c_2)}(\llbracket (C_i)\rrbracket) \\ \mathrm{ORLEFT}_{(C\vee D)}(\llbracket (C)\rrbracket, \llbracket (D)\rrbracket) \\ \mathrm{IMPLEFT}_{(C\supset D)}(\llbracket \langle C\rangle\rrbracket, \llbracket (D)\rrbracket) \end{array}\right.$$

Candidates:

$$\begin{array}{lcl} \llbracket (B)\rrbracket & \stackrel{\text{def}}{=} & X_0 \\ \llbracket \langle B\rangle\rrbracket & \stackrel{\text{def}}{=} & \mathrm{NEG}_{\langle B\rangle}(\llbracket (B)\rrbracket) \end{array}$$

where X_0 is the least fixed point of the operator $\mathrm{NEG}_{(B)} \circ \mathrm{NEG}_{\langle B\rangle}$.

Remark 2.3.5: The least fixed point of the operator $\mathrm{NEG}_{(B)} \circ \mathrm{NEG}_{\langle B\rangle}$ is defined, since both $\mathrm{BINDING}_{\langle B\rangle}$ and $\mathrm{BINDING}_{(B)}$ are decreasing operators. Consequently, $\mathrm{NEG}_{\langle B\rangle}$ and $\mathrm{NEG}_{(B)}$ are decreasing. But then $\mathrm{NEG}_{(B)}\circ \mathrm{NEG}_{\langle B\rangle}$ must be increasing, and the least fixed point X_0 exists according to Tarski's fixed point theorem [Davey and Priestley, 1990].

Two basic properties of the candidates are as follows.

Lemma 2.3.6: For the candidates we have:

$$\begin{array}{llll} \text{(i)} & \llbracket (B)\rrbracket = \mathrm{NEG}_{(B)}(\llbracket \langle B\rangle\rrbracket) & \text{(ii)} & \mathrm{AXIOMS}_{(B)} \subseteq \llbracket (B)\rrbracket \\ & \llbracket \langle B\rangle\rrbracket = \mathrm{NEG}_{\langle B\rangle}(\llbracket (B)\rrbracket) & & \mathrm{AXIOMS}_{\langle B\rangle} \subseteq \llbracket \langle B\rangle\rrbracket \end{array}$$

Proof. (i) is immediate from Definition 2.3.4, and (ii) holds trivially since NEG is closed under AXIOMS. □

Let us analyse some of the motivations behind the completely symmetric definition of the candidates.

Remark 2.3.7: Given the symmetry stated in Lemma 2.3.6(i), we have a simple method to check whether a named or co-named term belongs to a candidate. For example, take a co-named term of the form $\langle a{:}B\wedge C\rangle M$ for which we wish to know whether it belongs to $[\![\langle B\wedge C\rangle]\!]$. Because of the equation $[\![\langle B\wedge C\rangle]\!] = \text{NEG}_{\langle B\wedge C\rangle}([\![(B\wedge C)]\!])$ it is sufficient to show that $\langle a{:}B\wedge C\rangle M$ belongs to $\text{NEG}_{\langle B\wedge C\rangle}([\![(B\wedge C)]\!])$. By definition of $\text{NEG}_{\langle B\wedge C\rangle}$ we therefore have to show that $\langle a{:}B\wedge C\rangle M$ is in at least one of the following three sets:

$$\begin{array}{rl}
\text{(i)} & \text{AXIOMS}_{\langle B\wedge C\rangle}\\
\text{(ii)} & \text{ANDRIGHT}_{\langle B\wedge C\rangle}([\![\langle B\rangle]\!],[\![\langle C\rangle]\!])\\
\text{(iii)} & \text{BINDING}_{\langle B\wedge C\rangle}([\![(B\wedge C)]\!])
\end{array}$$

This means that $\langle a{:}B\wedge C\rangle M$ must satisfy certain conditions depending on its top-level term constructor. For example, in (i) it is required that the co-named term $\langle a{:}B\wedge C\rangle M$ is of the form $\langle a{:}B\wedge C\rangle \mathsf{Ax}(x,b)$; in (ii) of the form $\langle a{:}B\wedge C\rangle \mathsf{And}_R(\langle b\rangle S,\langle c\rangle T,a)$ where it is presupposed that $\langle b\rangle S$ and $\langle c\rangle T$ belong to $[\![\langle B\rangle]\!]$ and to $[\![\langle C\rangle]\!]$, respectively; in (iii) it is required that M is strongly normalising under any substitution on a with a named term belonging to the candidate $[\![(B\wedge C)]\!]$.

In the next three lemmas we shall deduce some properties for $\xrightarrow{aux}$.

Lemma 2.3.8: Suppose $M\in\mathcal{T}$ and $M \xrightarrow{aux} M'$.

(i) If M freshly introduces the name x, then M' freshly introduces x.

(ii) If M freshly introduces the co-name a, then M' freshly introduces a.

Proof. If M freshly introduces a name or a co-name, then M cannot be of the form $\mathsf{Cut}(\_,\_)$ (see Definition 2.2.2). The lemma follows by inspection of the reduction rules of $\xrightarrow{aux}$. ☐

Lemma 2.3.9: For all terms $M\in\mathcal{T}$ we have

(i) $M\{x := \langle a\rangle\mathsf{Ax}(y,a)\} \xrightarrow{aux}{}^{*} M[x\mapsto y]$

(ii) $M\{a := (x)\mathsf{Ax}(x,b)\} \xrightarrow{aux}{}^{*} M[a\mapsto b]$

Proof. By routine induction on the structure of M (see the proof of Lemma 2.2.11). ☐

Notation 2.3.10: The expression $M \xrightarrow{aux}{}^{0/1} M'$ stands for either $M\equiv M'$ or $M \xrightarrow{aux} M'$.

Lemma 2.3.11: For an arbitrary substitution $\{\sigma\}$,

if $M \xrightarrow{aux} M'$, then $M\{\sigma\} \xrightarrow{aux}{}^{0/1} M'\{\sigma\}$.

Proof. By induction on the structure of M (see Page 142). ☐

In effect, the substitution operation $\{\_\}$ behaves quite different under reduction in comparison with the standard notion of variable substitution. Note, on the other hand, if $N \xrightarrow{aux} N'$, then $M\{a := (x)N\} \xrightarrow{aux}{}^{*} M\{a := (x)N'\}$, as expected.

The next two lemmas establish important properties of the candidates. The first shows that the candidates are closed under reductions, and the second shows how the candidates are linked to the property of strong normalisation.

Lemma 2.3.12:

(i) If $\langle a{:}B\rangle M \in [\![\langle B\rangle]\!]$ and $M \xrightarrow{aux} M'$, then $\langle a{:}B\rangle M' \in [\![\langle B\rangle]\!]$.

(ii) If $(x{:}B)M \in [\![(B)]\!]$ and $M \xrightarrow{aux} M'$, then $(x{:}B)M' \in [\![(B)]\!]$.

Proof. By simultaneous induction on the degree of B (the number of logical symbols in B). Given Lemma 2.3.6(i), the proof analyses all possible sets where $\langle a{:}B\rangle M$ and $(x{:}B)M$ could be member in (see Page 142). $\square$

Lemma 2.3.13:

(i) If $\langle a{:}B\rangle M \in [\![\langle B\rangle]\!]$, then $M \in SN_{aux}$.

(ii) If $(x{:}B)M \in [\![(B)]\!]$, then $M \in SN_{aux}$.

Proof. The proof is similar to the one given for Lemma 2.3.12 (see Page 144). $\square$

We are now in the position to prove that a cut is strongly normalising provided its immediate subterms are strongly normalising and in a candidate corresponding to the cut-formula. This is a lengthy lemma: the cases for the logical reductions require rather difficult arguments.

Lemma 2.3.14: If $M, N \in SN_{aux}$, $\langle a{:}B\rangle M \in [\![\langle B\rangle]\!]$ and $(x{:}B)N \in [\![(B)]\!]$, then

$$\mathsf{Cut}(\langle a{:}B\rangle M, (x{:}B)N) \in SN_{aux}.$$

Proof. The proof of this lemma is inspired by a technique applied in [Prawitz, 1971]. It shows by induction that every immediate reduct of the term $\mathsf{Cut}(\langle a{:}B\rangle M, (x{:}B)N)$ is strongly normalising (see Pages 145–148). The induction proceeds over a lexicographically ordered induction value of the form (δ, μ, ν), where δ is the degree of the cut-formula B; μ and ν are the longest reduction sequences (relative to $\xrightarrow{aux}$) starting from M and N, respectively—by assumption both μ and ν are finite. $\square$

It is left to show that all well-typed terms are strongly normalising. In order to do so, we shall consider a special class of substitutions, which are called *safe*. Two substitutions, say $\{\sigma\}$ and $\{\tau\}$, are safe, if and only if the domain of $\{\sigma\}$ is not free in the co-domain of $\{\tau\}$ and the domain of $\{\tau\}$ is not free in the co-domain of $\{\sigma\}$. For example $\{x := \langle a\rangle P\}$ and $\{b := (y)Q\}$ are safe, provided that x is not free in $(y)Q$ and b is not free in $\langle a\rangle P$. As explained earlier, the substitution operation $\{\_\}$ is defined with the property in mind that safe substitutions can commute. As a special case of the substitution lemma for $\{\_\}$ (Lemma 2.3.1) we have for all M and two safe substitutions $\{\sigma\}$ and $\{\tau\}$

$$M\{\sigma\}\{\tau\} \equiv M\{\tau\}\{\sigma\} \ .$$

We shall now extend the notion of safety from substitutions to simultaneous substitutions; that is to sets of substitutions.

Definition 2.3.15 (Safe Simultaneous Substitution, $\boldsymbol{sss}$)**:**

$\emptyset$ is an sss.

$\hat{\sigma} \cup \{\sigma\}$ is an sss, if and only if

- $\hat{\sigma}$ is an sss,
- $dom(\{\sigma\}) \notin dom(\hat{\sigma})$,
- $dom(\{\sigma\})$ not free in $codom(\hat{\sigma})$, and
- $dom(\hat{\sigma})$ not free in $codom(\{\sigma\})$.

In the presence of our Barendregt-style naming convention and alpha-conversion, which we assumed in Convention 2.2.4, any set of substitutions can be transformed into a safe simultaneous substitution. We shall, however, omit a formal proof and rather give the reader the following example.

Example 2.3.16: Suppose we have a term, say M, and a safe simultaneous substitution, say

$$\hat{\sigma} = \Big\{\{x := \langle c \rangle \mathsf{Ax}(x, b)\}, \{a := (z)\mathsf{Ax}(z, c)\}\Big\},$$

and assume we have the substitution $\{\sigma\} \equiv \{b := (y)\mathsf{Ax}(x, a)\}$. Clearly, $\hat{\sigma} \cup \{\sigma\}$ is not safe, and therefore $M(\hat{\sigma} \cup \{\sigma\})$ is an ill-defined expression. However, x, a and b are considered as binders and thus can be rewritten. In effect, we can form the following safe version of $\hat{\sigma} \cup \{\sigma\}$

$$\hat{\sigma}_{\mathit{safe}} = \Big\{\{x' := \langle c \rangle \mathsf{Ax}(x, b)\}, \{a' := (z)\mathsf{Ax}(z, c)\}, \{b' := (y)\mathsf{Ax}(x, a)\}\Big\}$$

assuming that x', a' and b' are fresh. Subsequently, we need to rewrite the corresponding names and co-names in M. Now the expression $M[x \mapsto x'][a \mapsto a'][b \mapsto b']\,\hat{\sigma}_{\mathit{safe}}$ is well-defined.

In the next lemma we shall show that a specific substitution built up by axioms is an sss.

Lemma 2.3.17: Let $\hat{\sigma}$ be of the form

$$\left\{ \bigcup_{i=0,\ldots,n} \{x_i := \langle c \rangle \mathsf{Ax}(x_i, c)\} \right\} \cup \left\{ \bigcup_{j=0,\ldots,m} \{a_j := (y)\mathsf{Ax}(y, a_j)\} \right\}$$

where the x_i's and a_i's are distinct names and co-names, respectively. Substitution $\hat{\sigma}$ is an sss.

Proof. By induction on the length of $\hat{\sigma}$. $\square$

Now we can show that every well-typed term together with a closing substitution is strongly normalising. This is again a rather lengthy proof.

Lemma 2.3.18:

- For every well-typed term M—not necessarily strongly normalising—with a typing judgement $\Gamma \triangleright M \triangleright \Delta$, and
- for every sss, $\hat{\sigma}$, such that $dom(\Gamma) \cup dom(\Delta) \subseteq dom(\hat{\sigma})$, i.e., $\hat{\sigma}$ is a closing substitution,[5] and
  - for every $(x{:}B)P \in codom(\hat{\sigma})$ we require that $(x{:}B)P \in [\![(B)]\!]$ and
  - for every $\langle a{:}C\rangle Q \in codom(\hat{\sigma})$ we require that $\langle a{:}C\rangle Q \in [\![\langle C\rangle]\!]$;

  we have $M\hat{\sigma} \in SN_{aux}$.

Proof. By induction over the structure of M (see Pages 148–150). □

We are now able to prove that $(\mathfrak{T}, \xrightarrow{aux})$ is strongly normalising.

Theorem 2.3.19: The reduction system $(\mathfrak{T}, \xrightarrow{aux})$ is strongly normalising.

Proof. By Lemma 2.3.18 we know that for an arbitrary well-typed term, say M, with the typing judgement $\Gamma \triangleright M \triangleright \Delta$ and an arbitrary safe simultaneous substitution, say $\hat{\sigma}$, where all free names and co-names of M are amongst the domain of $\hat{\sigma}$, the term $M\hat{\sigma}$ is strongly normalising. Taking $\hat{\sigma}$ to be the safe simultaneous substitution from Lemma 2.3.17, we can infer, using Lemma 2.3.9, that $M\hat{\sigma} \xrightarrow{aux}{}^{*} M$, and we therefore have that M is strongly normalising. Thus we are done. □

From this result we can deduce strong normalisation for $(\mathfrak{T}, \xrightarrow{cut})$, which is relatively straightforward since every $\xrightarrow{cut}$-reduction maps onto a series of $\xrightarrow{aux}$-reductions. First, we prove that $M\{\sigma\}$ reduces to or is equivalent to $M[\sigma]$.

Lemma 2.3.20: For all terms $M, N \in \mathfrak{T}$ we have

(i) $M\{a := (x)N\} \xrightarrow{aux}{}^{*} M[a := (x)N]$

(ii) $M\{x := \langle a\rangle N\} \xrightarrow{aux}{}^{*} M[x := \langle a\rangle N]$

Proof. By induction on the structure of M (see Page 150). □

The next lemma shows that every $\xrightarrow{cut}$-reduction maps onto a series of $\xrightarrow{aux}$-reductions.

Lemma 2.3.21: For all terms $M, N \in \mathfrak{T}$, if $M \xrightarrow{cut} N$, then $M \xrightarrow{aux}{}^{+} N$.

Proof. By induction on the structure of $\xrightarrow{cut}$ (see Page 151). □

[5] All free names and co-names of M are amongst the domain of $\hat{\sigma}$.

It is rather easy now to show that $(\mathcal{T}, \xrightarrow{cut})$ is strongly normalising.

Theorem 2.3.22: The reduction system $(\mathcal{T}, \xrightarrow{cut})$ is strongly normalising.

Proof. The reduction system $(\mathcal{T}, \xrightarrow{aux})$ is strongly normalising, and whenever $M \xrightarrow{cut} N$ we have that $M \xrightarrow{aux}{}^{+} N$. Consequently, the reduction system $(\mathcal{T}, \xrightarrow{cut})$ must be strongly normalising, too. □

2.4 First-Order Classical Logic

In this section we extend the cut-elimination procedure $(\mathcal{T}, \xrightarrow{cut})$ to the first-order fragment of classical logic. To do so, we extend the notion of a formula by allowing atomic formulae to have arguments ranging over *expressions* and introduce the quantifiers $\forall$ and $\exists$. We use 'expression' instead of 'term' (the standard terminology) in order to avoid confusion with our terminology introduced in previous sections. Moreover, we use a sans serif font for expressions to clearly distinguish them from terms. The grammar for expressions is

$$\mathsf{t} ::= \mathsf{x} \mid \mathsf{f}\,\mathsf{t}\ldots\mathsf{t}$$

where x is taken from a set of *variables* and f from a set of functional symbols. An *arity* (a natural number) is assigned to each functional symbol specifying the number of its arguments. We shall often write $\mathsf{x}, \mathsf{y}, \mathsf{z}$ for variables and t for arbitrary expressions. The formulae of first-order classical logic are given by the grammar

$$B ::= A\,\mathsf{t}_1 \ldots \mathsf{t}_\mathsf{n} \mid \neg B \mid B{\wedge}B \mid B{\vee}B \mid B{\supset}B \mid \forall\mathsf{x}.B \mid \exists\mathsf{x}.B$$

where A ranges over *predicate symbols*, each of which take a fixed number of expressions as arguments, and where x ranges over variables. An occurrence of the variable x is said to be *bound*, if it is inside a formula of the form $\forall\mathsf{x}.B$ or $\exists\mathsf{x}.B$. We have the standard notion of free variables, and for the set of free variables of an expression, say t, and of a formula, say B, we shall write $FV(\mathsf{t})$ and $FV(B)$, respectively.

Just as with terms, we identify formulae that differ only in the names of their bound variables. Moreover, we shall observe a Barendregt-style naming convention for bound and free variables. This is a standard convention allowing us to assume that distinct quantifier occurrences are followed by distinct variables. For example, it excludes formulae such as $\exists\mathsf{x}.(A\mathsf{x}{\wedge}\forall\mathsf{x}.B\mathsf{x})$, which can equally be written as $\exists\mathsf{y}.(A\mathsf{y}{\wedge}\forall\mathsf{x}.B\mathsf{x})$. It also allows a simple definition of the notion of capture avoiding variable substitution, written as $[\mathsf{x} := \mathsf{t}]$, but we omit to give the definition (see for example [Gallier, 1990, Page 4]). Later, it will be convenient to extend this notion to contexts. For instance $\Gamma[\mathsf{x} := \mathsf{t}]$ will be taken to mean that the substitution $[\mathsf{x} := \mathsf{t}]$ is applied to every formula in Γ.

We are now ready to define the sequent calculus for first-order classical logic. There are four rules to govern the quantifiers, *viz.*

$$\frac{x : B[\mathsf{x} := \mathsf{t}], \Gamma \vdash \Delta}{y : \forall \mathsf{x}.B, \Gamma \vdash \Delta}\ \forall_L \qquad \frac{\Gamma \vdash \Delta, a : B[\mathsf{x} := \mathsf{y}]}{\Gamma \vdash \Delta, b : \forall \mathsf{x}.B}\ \forall_R$$

$$\frac{x : B[\mathsf{x} := \mathsf{y}], \Gamma \vdash \Delta}{y : \exists \mathsf{x}.B, \Gamma \vdash \Delta}\ \exists_L \qquad \frac{\Gamma \vdash \Delta, a : B[\mathsf{x} := \mathsf{t}]}{\Gamma \vdash \Delta, b : \exists \mathsf{x}.B}\ \exists_R$$

The $\forall_R$ and $\exists_L$ rules are subject to the usual proviso that y should not appear freely in Γ and Δ. We obtain the first-order system by adding these rules to the calculus shown in Figure 2.2. The new rules give rise to the term annotations given next.

Definition 2.4.1 (Raw Terms for Quantifiers)**:** We extend the grammar given in Definition 2.2.1 with the following clauses:

$$\begin{array}{rcll} M, N & ::= & \mathsf{Exists}_R(\langle a{:}B\rangle M, \mathsf{t}, b) & \text{exists-right} \\ & | & \mathsf{Exists}_L((x{:}B)[\mathsf{y}]M, y) & \text{exists-left} \\ & | & \mathsf{Forall}_R(\langle a{:}B\rangle[\mathsf{y}]M, b) & \text{forall-right} \\ & | & \mathsf{Forall}_L((x{:}B)M, \mathsf{t}, y) & \text{forall-left} \end{array}$$

where y is a variable and t is an expression; square brackets denote a new binding operation for variables. Thus $[\mathsf{y}]M$ means that y becomes bound in M.

Notice that in the clauses for exists-right and forall-left we have to record an expression (this will become clearer when we define the cut-reductions). We can now give the rules for forming typing judgements.

$$\frac{x : B[\mathsf{x} := \mathsf{t}], \Gamma \triangleright M \triangleright \Delta}{y : \forall \mathsf{x}.B, \Gamma \triangleright \mathsf{Forall}_L((x)M, \mathsf{t}, y) \triangleright \Delta}\ \forall_L \qquad \frac{\Gamma \triangleright M \triangleright \Delta, a : B[\mathsf{x} := \mathsf{y}]}{\Gamma \triangleright \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M, b) \triangleright \Delta, b : \forall \mathsf{x}.B}\ \forall_R$$

$$\frac{x : B[\mathsf{x} := \mathsf{y}], \Gamma \triangleright M \triangleright \Delta}{y : \exists \mathsf{x}.B, \Gamma \triangleright \mathsf{Exists}_L((x)[\mathsf{y}]M, y) \triangleright \Delta}\ \exists_L \qquad \frac{\Gamma \triangleright M \triangleright \Delta, a : B[\mathsf{x} := \mathsf{t}]}{\Gamma \triangleright \mathsf{Exists}_R(\langle a\rangle M, \mathsf{t}, b) \triangleright \Delta, b : \exists \mathsf{x}.B}\ \exists_R$$

Again, the $\forall_R$ and $\exists_L$ rules are subject to the proviso that y does not occur freely in Γ and Δ. Since the logical cut-reductions for the quantifiers include variable substitutions, we need a corresponding operation for terms. Assuming the Barendregt-style naming convention for variables, this operation is defined as follows.

Definition 2.4.2 (Variable Substitution for Terms)**:** The four non-trivial cases are:

$$\begin{array}{rcl} \mathsf{Exists}_R(\langle a\rangle M, \mathsf{s}, b)[\mathsf{x} := \mathsf{t}] & \stackrel{\text{def}}{=} & \mathsf{Exists}_R(\langle a\rangle\ M[\mathsf{x} := \mathsf{t}], \mathsf{s}[\mathsf{x} := \mathsf{t}], b) \\ \mathsf{Exists}_L((x)[\mathsf{y}]M, y)[\mathsf{x} := \mathsf{t}] & \stackrel{\text{def}}{=} & \mathsf{Exists}_L((x)[\mathsf{y}]\ M[\mathsf{x} := \mathsf{t}], b) \\ \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M, b)[\mathsf{x} := \mathsf{t}] & \stackrel{\text{def}}{=} & \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]\ M[\mathsf{x} := \mathsf{t}], b) \\ \mathsf{Forall}_L((x)M, \mathsf{s}, y)[\mathsf{x} := \mathsf{t}] & \stackrel{\text{def}}{=} & \mathsf{Forall}_L((x)\ M[\mathsf{x} := \mathsf{t}], \mathsf{s}[\mathsf{x} := \mathsf{t}], b) \end{array}$$

We should point out that our use of the Barendregt-style naming convention for variables corresponds to the *pure-variable convention* commonly used in treatises of cut-elimination based on sequent proofs (for example [Troelstra and Schwichtenberg, 1996, Page 58]). This convention is vital in the cut-elimination procedure, since without it not all first-order proofs will have a cut-free normal form.

Whilst the extension of the proof substitution shown in Figure 2.4 and the corresponding adaptation of the reduction $\xrightarrow{c}$ for the first-order case are quite easy, and thus not explained, the reduction rules for logical cuts need some explanation. Consider the definition below.

Definition 2.4.3 (Cut-Reductions for Quantifiers)**:**

$$\begin{array}{l}\mathsf{Cut}(\langle b\rangle\mathsf{Exists}_R(\langle a\rangle M,\mathsf{t},b),(y)\mathsf{Exists}_L((x)[\mathsf{y}]N,y))\\ \xrightarrow{l}\ \mathsf{Cut}(\langle a\rangle M,(x)N[\mathsf{y}:=\mathsf{t}])\\ \qquad\qquad \text{if } \mathsf{Exists}_R(\langle a\rangle M,\mathsf{t},b) \text{ and } \mathsf{Exists}_L((x)[\mathsf{y}]N,y) \text{ freshly introduce } a \text{ and } x\end{array}$$

$$\begin{array}{l}\mathsf{Cut}(\langle b\rangle\mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M,b),(y)\mathsf{Forall}_L((x)N,\mathsf{t},y))\\ \xrightarrow{l}\ \mathsf{Cut}(\langle a\rangle M[\mathsf{y}:=\mathsf{t}],(x)N)\\ \qquad\qquad \text{if } \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M,b) \text{ and } \mathsf{Forall}_L((x)N,\mathsf{t},y) \text{ freshly introduce } a \text{ and } x\end{array}$$

We must ensure that reducing a valid proof yields again a valid proof (in the sense of being derivable by the inference rules). This holds only if some alpha-conversions are made. An example where such a conversion is necessary was given by Kleene [1952a, Page 450], and Troelstra and Schwichtenberg [1996, Page 58]. We assume these conversions are done implicitly.

In the sequel we shall take $\xrightarrow{l}$, $\xrightarrow{c}$, $[\sigma]$ and $\mathcal{T}$ to denote the notions for first-order classical logic. The definitions are laborious, but straightforward given the details provided for the propositional calculus. The cut-elimination procedure for the first-order system is defined as follows.

Definition 2.4.4 (Cut-Elimination for First-Order Classical Logic)**:**
The cut-elimination procedure for first-order classical logic, $(\mathcal{T},\xrightarrow{cut})$, is a reduction system where:

- $\mathcal{T}$ is the set of well-typed (first-order) terms, and
- $\xrightarrow{cut}$ consists of the reductions for logical cuts and commuting cuts; that is

$$\xrightarrow{cut} \stackrel{\text{def}}{=} \xrightarrow{l} \cup \xrightarrow{c}\ .$$

We should now prove that $\xrightarrow{cut}$ still respects the subject reduction property. However, this proof is a straightforward extension of the proof of Proposition 2.2.10. The only additional lemma we need in this proof is as follows.

Lemma 2.4.5: If $\Gamma \triangleright M \triangleright \Delta$ is a typing judgement, then for an arbitrary substitution $[\mathsf{x} := \mathsf{t}]$ also $\Gamma[\mathsf{x} := \mathsf{t}] \triangleright M[\mathsf{x} := \mathsf{t}] \triangleright \Delta[\mathsf{x} := \mathsf{t}]$.

Proof. By induction on the structure of M. □

We conclude this section with the strong normalisation theorem for the first-order system.

Theorem 2.4.6: The reduction system $(\mathcal{T}, \xrightarrow{cut})$ for first order classical logic is strongly normalising.

Proof. The proof is similar to the strong normalisation proof for the propositional calculus, and for space reasons all but the following details are omitted.

- An auxiliary substitution, $\{\_\}$, and an auxiliary reduction, $\xrightarrow{aux}$, need to be defined along the lines of the definitions given for the propositional calculus. This avoids the problem of $[\_]$ where safe substitutions do not commute.
- The candidates need to be extended for quantified formulae; Figure 2.8 gives the corresponding set operators.
- The induction in Lemma 2.3.18 includes in the first-order system a closing substitution for variables. □

Unless otherwise stated, we henceforth assume that $(\mathcal{T}, \xrightarrow{cut})$ and $(\mathcal{T}, \xrightarrow{aux})$ refer to first-order classical logic.

2.5 Variations of the Sequent Calculus

Designing a logical calculus is like composing a fugue: there are many variations. A natural question is to what extent the strong normalisation result depends on our particular formulation of the inference rules. Some aspects of our rules are crucial in the strong normalisation proof; others are not. The use of sets of labelled formulae as contexts, for example, plays an important rôle. Labels allow us to distinguish different formula occurrences in a typing derivation. This is an indispensable proviso for a strong normalisation proof built around the notion of candidates; see [Leivant, 1979]. On the other hand, the specific form of our inference rules is irrelevant, in the sense that our candidate method extends to other formulations of the inference rules. We shall give some examples below.

First, there are alternative formulations for and-left and or-right. The and-left rule, for instance, may be of the form

$$\frac{x : B_i, \Gamma \vdash \Delta}{z : B_1 \wedge B_2, \Gamma \vdash \Delta}\ \wedge_{L_i}\quad (i = 1, 2) \qquad \text{or} \qquad \frac{x : B, y : C, \Gamma \vdash \Delta}{z : B \wedge C, \Gamma \vdash \Delta}\ \wedge'_L\ .$$

The first rule, which we included in our sequent calculus, is often referred to as the additive variant of the and-left rule, and the second as the multiplicative variant. If we are only interested in provability, it is not difficult to show that in classical logic the two variants are interderivable. By this we mean that one can be derived from the other. However, their two behaviours in the process of cut-elimination are rather different. Whereas a cut-instance $\wedge_R/\wedge_{L_i}$ is replaced by a single cut on a smaller formula, a cut-instance $\wedge_R/\wedge'_L$

$$\textsc{forallleft}_{(\forall\mathsf{x}.B)}(X) \overset{\text{def}}{=} \left\{ \begin{array}{l} (y{:}\forall\mathsf{x}.B)\mathsf{Forall}_L((x{:}B[\mathsf{x}:=\mathsf{t}])M,\mathsf{t},y) \mid \\ \qquad \mathsf{Forall}_L((x{:}B[\mathsf{x}:=\mathsf{t}])M,\mathsf{t},y) \text{ freshly introduces } y, \\ \qquad (x{:}B[\mathsf{x}:=\mathsf{t}])M \in X \end{array} \right\}$$

$$\textsc{forallright}_{\langle\forall\mathsf{x}.B\rangle}(X) \overset{\text{def}}{=} \left\{ \begin{array}{l} (b{:}\forall\mathsf{x}.B)\mathsf{Forall}_R(\langle a{:}B[\mathsf{x}:=\mathsf{y}]\rangle[\mathsf{y}]M,b) \mid \\ \qquad \mathsf{Forall}_R(\langle a{:}B[\mathsf{x}:=\mathsf{y}]\rangle[\mathsf{y}]M,b) \text{ freshly introduces } b, \\ \qquad \text{for all } \mathsf{t} \text{ s.t. } (x{:}B[\mathsf{x}:=\mathsf{t}])M \in X \end{array} \right\}$$

$$\textsc{existsleft}_{(\exists\mathsf{x}.B)}(X) \overset{\text{def}}{=} \left\{ \begin{array}{l} (y{:}\exists\mathsf{x}.B)\mathsf{Exists}_L((x{:}B[\mathsf{x}:=\mathsf{t}])[\mathsf{y}]M,y) \mid \\ \qquad \mathsf{Exists}_L((x{:}B[\mathsf{x}:=\mathsf{t}])[\mathsf{y}]M,y) \text{ freshly introduces } y, \\ \qquad \text{for all } \mathsf{t} \text{ s.t. } (x{:}B[\mathsf{x}:=\mathsf{t}])M \in X \end{array} \right\}$$

$$\textsc{existsright}_{\langle\exists\mathsf{x}.B\rangle}(X) \overset{\text{def}}{=} \left\{ \begin{array}{l} (b{:}\exists\mathsf{x}.B)\mathsf{Exists}_R(\langle a{:}B[\mathsf{x}:=\mathsf{y}]\rangle M,\mathsf{t},b) \mid \\ \qquad \mathsf{Exists}_R(\langle a{:}B[\mathsf{x}:=\mathsf{y}]\rangle M,\mathsf{t},b) \text{ freshly introduces } b, \\ \qquad (x{:}B[\mathsf{x}:=\mathsf{t}])M \in X \end{array} \right\}$$

Figure 2.8: Definition of the set operators for quantifiers.

is replaced by two cuts, and one can choose the order in which these cuts appear in the reduct. Given the term assignment for $\wedge'_L$ shown in Figure 2.9, we thus have the reduction rule

$$\begin{array}{l} \mathsf{Cut}(\langle c\rangle\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c),(z)\mathsf{And}'_L((x)(y)P,z)) \\ \xrightarrow{l} \mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Cut}(\langle b\rangle N,(y)P)) \quad \text{or} \\ \xrightarrow{l} \mathsf{Cut}(\langle b\rangle N,(y)\mathsf{Cut}(\langle a\rangle M,(x)P)) \end{array}$$

$$\text{if } \mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c) \text{ and } \mathsf{And}'_L((x)(y)P,z) \text{ freshly introduce } c \text{ and } z,$$

This reduction is similar to that given in Figure 2.5 for $\supset_R/\supset_L$, and also requires similar side-conditions as specified in Remark 2.2.7. Much the same discussion as given for and-left applies to or-right, whose multiplicative variant, $\vee'_R$, is shown in Figure 2.9.

Another source for variation is how we treat the contexts in the rules with two premises: these contexts can be either shared or split. In effect, we have the following two possibilities for and-right.

$$\frac{\Gamma \vdash \Delta, a:B \quad \Gamma \vdash \Delta, b:C}{\Gamma \vdash \Delta, c:B\wedge C}\wedge_R \qquad \frac{\Gamma_1 \vdash \Delta_1, a:B \quad \Gamma_2 \vdash \Delta_2, b:C}{\Gamma_1,\Gamma_2 \vdash \Delta_1,\Delta_2, c:B\wedge C}\wedge'_R$$

Note that in the $\wedge'_R$-rule the context are treated as in the cut-rule; that means $\Gamma_1 \cap \Gamma_2$ or $\Delta_1 \cap \Delta_2$ may not be empty. Thus $\wedge'_R$ is actually a generalisation of $\wedge_R$. Similar remarks apply to the other inference rules with two premises (see Figure 2.9).

Our strong normalisation proof can be easily modified to accommodate the alternative formulations of the inference rules mentioned above. In fact, it is robust enough to prove strong normalisation for a sequent calculus containing all variants and their respective

$$\frac{x:B, y:C, (z:B\wedge C), \Gamma \triangleright M \triangleright \Delta}{z:B\wedge C, \Gamma \triangleright \mathsf{And}'_L((x)(y)M, z) \triangleright \Delta}\wedge'_L \qquad \frac{\Gamma \triangleright M \triangleright \Delta, (c:B\wedge C), a:B, b:C}{\Gamma \triangleright \mathsf{Or}'_R(\langle a\rangle\langle b\rangle M, c) \triangleright \Delta, c:B\wedge C}\vee'_R$$

$$\frac{\Gamma_1 \triangleright M \triangleright \Delta_1, a:B \qquad \Gamma_2 \triangleright N \triangleright \Delta_2, b:C}{\Gamma_1, \Gamma_2 \triangleright \mathsf{And}'_R(\langle a\rangle M, \langle b\rangle N, c) \triangleright \Delta_1, \Delta_2, c:B\wedge C}\wedge'_R$$

$$\frac{x:B, \Gamma_1 \triangleright M \triangleright \Delta_1 \qquad y:C, \Gamma_2 \triangleright N \triangleright \Delta_2}{z:B\vee C, \Gamma_1, \Gamma_2 \triangleright \mathsf{Or}'_L((x)M, (y)N, z) \triangleright \Delta_1, \Delta_2}\vee'_L$$

$$\frac{\Gamma_1 \triangleright M \triangleright \Delta_1, a:B \qquad x:C, \Gamma_2 \triangleright N \triangleright \Delta_2}{y:B\supset C, \Gamma_1, \Gamma_2 \triangleright \mathsf{Imp}'_L(\langle a\rangle M, (x)N, y) \triangleright \Delta_1, \Delta_2}\supset'_L$$

Figure 2.9: Alternative formulation of some inference rules.

cut-reductions. Since the proof does not pose any new challenges, the details are omitted. In what follows we shall employ the formulation of the inference rules that seems most suited for the problem at hand.

The last variation for classical logic we shall present arises from the wonderful symmetry of classical logic. Every sequent of the form $\Gamma \vdash \Delta$ can be replaced by $\vdash \neg\Gamma, \Delta$ where $\neg\Gamma$ is taken to mean that every formula in Γ is negated. The corresponding sequent calculus is commonly referred to as single-sided sequent calculus or Gentzen-Schütte system [Schütte, 1960]. The cut-rule, for example, is in this calculus of the following form.

$$\frac{\vdash \Gamma, a:B \qquad \vdash \Delta, b:\neg B}{\vdash \Gamma, \Delta}\text{Cut}$$

It is easy to adapt Definition 2.3.4 (the symmetric reducibility candidates) for a single-sided sequent calculus, and the proof of strong normalisation for cut-elimination proceeds just as the proof given for $\xrightarrow{cut}$. We shall omit the details.

To conclude this section, we shall introduce term annotations for intuitionistic sequent proofs and prove a corresponding strong normalisation theorem. For simplicity, we shall treat only the propositional connectives, $\wedge$, $\vee$ and $\supset$, and not negation and the units, as they add no substantial new issues for strong normalisation.

The sequent calculus for intuitionistic logic can be obtained by restricting the succedent of our classical sequents to at most a single formula. So our intuitionistic sequents are of the form $\Gamma \vdash a:B$. We could drop the labels on the right-hand side of the turnstile when assigning terms to intuitionistic sequents, because on this side contraction is not allowed. However, we shall keep this slight inconvenience of having to label this formula: it saves us from redefining the substitution operation and cut-reductions. Thus our intuitionistic typing judgements are of the form $\Gamma \triangleright M \triangleright a:B$, in which M is a term taken from the set of well-typed terms, denoted by $\mathcal{I}$ and inductively defined by the inference rules given in Figure 2.10. The intuitionistic cut-elimination procedure is as follows.

$$\frac{}{x:B,\Gamma \triangleright \mathsf{Ax}(x,a) \triangleright a:B}$$

$$\frac{x:B_i,\Gamma \triangleright M \triangleright a:C}{y:B_1\wedge B_2,\Gamma \triangleright \mathsf{And}^i_L((x)M,y) \triangleright a:C}\ \wedge_{L_i} \qquad \frac{\Gamma \triangleright M \triangleright a:B \qquad \Gamma \triangleright N \triangleright b:C}{\Gamma \triangleright \mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c) \triangleright c:B\wedge C}\ \wedge_R$$

$$\frac{x:B,\Gamma \triangleright M \triangleright a:D \qquad y:C,\Gamma \triangleright N \triangleright a:D}{z:B\vee C,\Gamma \triangleright \mathsf{Or}_L((x)M,(y)N,z) \triangleright a:D}\ \vee_L \qquad \frac{\Gamma \triangleright M \triangleright a:B_i}{\Gamma \triangleright \mathsf{Or}^i_R(\langle a\rangle M,b) \triangleright b:B_1\vee B_2}\ \vee_{R_i}$$

$$\frac{\Gamma \triangleright M \triangleright a:B \qquad x:C,\Gamma \triangleright N \triangleright b:D}{y:B\supset C,\Gamma \triangleright \mathsf{Imp}_L(\langle a\rangle M,(x)N,y) \triangleright b:D}\ \supset_L \qquad \frac{x:B,\Gamma \triangleright M \triangleright a:C}{\Gamma \triangleright \mathsf{Imp}_R((x)\langle a\rangle M,b) \triangleright b:B\supset C}\ \supset_R$$

$$\frac{\Gamma_1 \triangleright M \triangleright a:B \qquad x:B,\Gamma_2 \triangleright N \triangleright b:C}{\Gamma_1,\Gamma_2 \triangleright \mathsf{Cut}(\langle a\rangle M,(x)N) \triangleright b:C}\ \text{Cut}$$

Figure 2.10: Term assignment for sequent proofs in the propositional fragment of intuitionistic logic.

Definition 2.5.1 (Cut-Elimination for Intuitionistic Logic)**:**
The cut-elimination procedure for intuitionistic logic, $(\mathcal{I},\xrightarrow{int})$, is a reduction system where:

- $\mathcal{I}$ is the set of well-typed intuitionistic terms, and
- $\xrightarrow{int}$ consists of the reductions for logical cuts and commuting cuts; that is

$$\xrightarrow{int} \stackrel{\text{def}}{=} \xrightarrow{l} \cup \xrightarrow{c} \quad .$$

Given that intuitionistic logic, as we defined it, is a restricted version of classical logic, we expect that $(\mathcal{I},\xrightarrow{int})$ is strongly normalising. This is indeed the case.

Theorem 2.5.2: The reduction system $(\mathcal{I},\xrightarrow{int})$ is strongly normalising.

Proof. Assume there is a well-typed intuitionistic term, say M, from which an infinite reduction sequence starts. Since we know that $\Gamma \triangleright M \triangleright a:B$ is derivable using the rules given in Figure 2.10, we can translate this derivation into the classical system shown in Figure 2.3 (only the $\supset_L$-rule requires some changes). It is not hard to see that the infinite reduction sequence starting from M would entail an infinite reduction sequence in $(\mathcal{T},\xrightarrow{cut})$. Since we proved that $(\mathcal{T},\xrightarrow{cut})$ is strongly normalising, $(\mathcal{I},\xrightarrow{int})$ must be strongly normalising, too. $\square$

We shall use this result in Chapter 3 in order to prove strong normalisation for the simply-typed lambda calculus.

2.6 Localised Version

In the previous section we claimed that our strong normalisation result extends to a variety of formulations of the inference rules. In this section we shall show that strong normalisation can also be obtained with a different set of cut-reduction rules. To save space, we shall restrict the presentation in this section to only the propositional fragment, but it should be emphasised that the corresponding result for first-order classical logic may be obtained by a simple adaptation of the proof we shall give.

The cut-elimination procedures defined so far include global, or non-local, cut-reductions. A cut-reduction is said to be *local*, if only neighbouring inference rules are rewritten, possibly duplicating a subderivation. Most of the traditional cut-elimination procedures based on Gentzen's *Hauptsatz* have such local cut-reductions. One reason for our use of the non-local cut-reductions was to avoid the infinite reduction given in Example 2.1.3. Using results concerning explicit substitution calculi, we shall show that our substitution operation can, in fact, be replaced by local cut-reductions without breaking strong normalisation. The local cut-reductions will be defined such that they bridge the gap between our global cut-reductions given earlier and the Gentzen-like local cut-reductions.

Explicit substitution calculi have been developed in order to internalise the substitution operation, a meta-level operation on lambda terms, arising from beta-reductions. In these calculi the reductions are completely primitive in the sense that they are part of the calculus. Note, however, if the (non-local) substitution operation is replaced by a naïve definition of a (local) explicit substitution operator, the resulting calculus may no longer be strongly normalising, as shown by Melliès [1995]. Nevertheless, it is possible to give an explicit substitution operation that preserves strong normalisation.

In $\lambda\mathrm{x}$, for example, the beta-reduction $(\lambda x.M)N \xrightarrow{\beta} M[x := N]$ is replaced by the reduction $(\lambda x.M)N \xrightarrow{b} M\langle x := N\rangle$ where the reduct contains a new syntactic constructor. The following reduction rules apply to this term constructor.

$$
\begin{array}{rcl}
x\langle x := P\rangle & \xrightarrow{x} & P \\
y\langle x := P\rangle & \xrightarrow{x} & y \\
(\lambda y.M)\langle x := P\rangle & \xrightarrow{x} & \lambda y.\, M\langle x := P\rangle \\
(MN)\langle x := P\rangle & \xrightarrow{x} & M\langle x := P\rangle\, N\langle x := P\rangle
\end{array}
$$

The reader is referred to [Rose, 1996] for a detailed study of $\lambda\mathrm{x}$ including a proof of strong normalisation. Here, we shall show that a similar feat is possible for our term calculus, where the substitution operation realises the proof transformations corresponding to the elimination of commuting cuts.

We begin by adding to our term calculus two new constructors which we refer to as *labelled cuts*. They are written:

$$\mathsf{C\overleftarrow{u}t}(\langle a\rangle M, (x)N) \qquad \text{and} \qquad \mathsf{Cu\overrightarrow{t}}(\langle a\rangle M, (x)N)\ .$$

Consequently, we have two new rules for forming typing judgements.

$$
\dfrac{\Gamma_1 \triangleright M \triangleright \Delta_1, a:B \quad x:B, \Gamma_2 \triangleright N \triangleright \Delta_2}{\Gamma_1, \Gamma_2 \triangleright \mathsf{C\overleftarrow{u}t}(\langle a\rangle M, (x)N) \triangleright \Delta_1, \Delta_2}\,\mathrm{C\overleftarrow{u}t}
\quad
\dfrac{\Gamma_1 \triangleright M \triangleright \Delta_1, a:B \quad x:B, \Gamma_2 \triangleright N \triangleright \Delta_2}{\Gamma_1, \Gamma_2 \triangleright \mathsf{Cu\overrightarrow{t}}(\langle a\rangle M, (x)N) \triangleright \Delta_1, \Delta_2}\,\mathrm{Cu\overrightarrow{t}}
\tag{2.5}
$$

Convention 2.2.4 concerning the Barendregt-style naming convention and alpha-equivalence extends to the new terms in the obvious way. Henceforth, we shall only be interested in well-typed terms; that means terms M for which there are contexts Γ and Δ such that $\Gamma \triangleright M \triangleright \Delta$ is derivable given the rules in (2.5) and Figure 2.3. The corresponding set of well-typed terms is defined as

$$\mathcal{T}^{\leftrightarrow} \stackrel{\text{def}}{=} \Big\{ M \;\Big|\; M \text{ is well-typed by the rules shown in (2.5) and Figure 2.3} \Big\}$$

With respect to the terms given earlier, clearly we have $\mathcal{T} \subset \mathcal{T}^{\leftrightarrow}$. In the sequel, we shall refer to a term whose outermost term constructor is a labelled cut as a *labelled term*.

Next, we replace the non-local reductions of $\xrightarrow{aux}$ with local ones. The only reduction that needs to be replaced is the reduction for commuting cuts, since it is the only one which reduces a term by applying a proof substitution. The new reductions are:

- the reduction $\xrightarrow{c''}$, which "labels" cuts when they need to be commuted, *viz.*

$$\begin{array}{lll}\mathsf{Cut}(\langle a\rangle M,(x)N) & \xrightarrow{c''} & \overleftarrow{\mathsf{Cut}}(\langle a\rangle M,(x)N) \\ & & \quad\text{if } M \text{ does not freshly introduce } a \text{ and is not labelled, or} \\ & \xrightarrow{c''} & \overrightarrow{\mathsf{Cut}}(\langle a\rangle M,(x)N) \\ & & \quad\text{if } N \text{ does not freshly introduce } x \text{ and is not labelled}\end{array}$$

- the reduction $\xrightarrow{x}$, which permutes labelled cuts to the places where the cut-formula is introduced (Figure 2.11 gives the corresponding reduction rules).

We shall assume that terms have been suitably alpha-converted before applying an $\xrightarrow{x}$-reduction, so that name or co-name capture will not occur, and we assume these conversions are done implicitly.

Furthermore, we shall introduce a *garbage reduction*, which is somewhat similar to a traditional cut-reduction where the cut-formula is weakened, and remove the side-conditions on the logical cut-reductions with axioms. Thus we have the following reductions:

- the $\xrightarrow{gc}$-reduction eliminates labelled cuts provided their corresponding name or co-name is not free, *viz.*

$$\begin{array}{llll}\overleftarrow{\mathsf{Cut}}(\langle a\rangle M,(x)N) & \xrightarrow{gc} & M & \text{if } a \notin FC(M) \\ \overrightarrow{\mathsf{Cut}}(\langle a\rangle M,(x)N) & \xrightarrow{gc} & N & \text{if } x \notin FN(N)\end{array}$$

- the $\xrightarrow{l'}$-reduction is just $\xrightarrow{l}$ except for the logical cuts with axioms, which are as follows:

$$\begin{array}{lll}\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(x,a),(y)M) & \xrightarrow{l'} & M[y\mapsto x] \\ \mathsf{Cut}(\langle b\rangle M,(x)\mathsf{Ax}(x,a)) & \xrightarrow{l'} & M[b\mapsto a]\end{array}$$

At first sight, it seems as if a $\xrightarrow{gc}$-reduction can be simulated by some $\xrightarrow{x}$-reductions, but in general this is not true, because labelled cuts cannot be permuted with other labelled cuts. In the same way, a $\xrightarrow{l'}$-reduction cannot, in general, be simulated by some $\xrightarrow{l}$- and $\xrightarrow{x}$-reductions. Since $\xrightarrow{gc}$ and $\xrightarrow{l'}$ can be easily accommodated in our strong normalisation proof, we include them in the local cut-elimination procedure.

$$
\begin{aligned}
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle \mathsf{Ax}(x,c),(y)P) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle \mathsf{Ax}(x,c),(y)P)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle \mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Ax}(x,c)),(y)P) &\xrightarrow{x} \mathsf{Cut}(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle M,(y)P),(y)P)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle \mathsf{Not}_R((x)M,a),(y)P) &\xrightarrow{x} \mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle M,(y)P),),(y)P)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c),(y)P) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle M,(y)P),\langle b\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle N,(y)P),c),(y)P)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle \mathsf{Or}^i_R(\langle a\rangle M,c),(y)P) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle \mathsf{Or}^i_R(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle M,(y)P),c),(y)P)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M,b),(y)P) &\xrightarrow{x} \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle M,(y)P),b),(y)P)
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)\mathsf{Ax}(y,a)) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{Ax}(y,a))\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y,a),(z)M)) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle P,(z)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)M))\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(x)\mathsf{Not}_L(\langle a\rangle M,x)) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle P,(x)\mathsf{Not}_L(\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(x)M),x))\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)\mathsf{And}^i_L((x)M,y)) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{And}^i_L((x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)M),y))\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(z)\mathsf{Or}_L((x)M,(y)N,z)) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle P,(z)\mathsf{Or}_L((x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(z)M),(y)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(z)N),z))\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)) &\xrightarrow{x} \mathsf{Cut}(\langle c\rangle P,(y)\mathsf{Imp}_L(\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)M),(x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)N),y))
\end{aligned}
$$

Otherwise:

$$
\begin{aligned}
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle \mathsf{Ax}(x,a),(y)P) &\xrightarrow{x} \mathsf{Ax}(x,a)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle \mathsf{Cut}(\langle a\rangle M,(x)N),(y)P) &\xrightarrow{x} \mathsf{Cut}(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle M,(y)P),(x)\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle N,(y)P))\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle \mathsf{Not}_R((x)M,a),(y)P) &\xrightarrow{x} \mathsf{Not}_R((x)\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle M,(y)P),a)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle \mathsf{Not}_L(\langle a\rangle M,x),(y)P) &\xrightarrow{x} \mathsf{Not}_L(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle M,(y)P),x)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle d\rangle \mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c),(y)P) &\xrightarrow{x} \mathsf{And}_R(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle d\rangle M,(y)P),\langle b\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle d\rangle N,(y)P),c)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle \mathsf{And}^i_L((x)M,y),(z)P) &\xrightarrow{x} \mathsf{And}^i_L((x)\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle M,(z)P),y)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle \mathsf{Or}^i_R(\langle a\rangle M,b),(x)P) &\xrightarrow{x} \mathsf{Or}^i_R(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle M,(x)P),b)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle \mathsf{Or}_L((x)M,(y)N,z),(v)P) &\xrightarrow{x} \mathsf{Or}_L((x)\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle M,(v)P),(y)\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle N,(v)P),z)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle \mathsf{Imp}_R((x)\langle a\rangle M,b),(y)P) &\xrightarrow{x} \mathsf{Imp}_R((x)\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle c\rangle M,(y)P),b)\\
\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle \mathsf{Imp}_L(\langle a\rangle M,(x)N,y),(z)P) &\xrightarrow{x} \mathsf{Imp}_L(\langle a\rangle \mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle M,(z)P),(x)\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle b\rangle N,(z)P),y)\\[1ex]
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)\mathsf{Ax}(x,a)) &\xrightarrow{x} \mathsf{Ax}(x,a)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)\mathsf{Cut}(\langle a\rangle M,(x)N)) &\xrightarrow{x} \mathsf{Cut}(\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)M),(x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)N))\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)\mathsf{Not}_R((x)M,a)) &\xrightarrow{x} \mathsf{Not}_R((x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)M),a)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)\mathsf{Not}_L(\langle a\rangle M,x)) &\xrightarrow{x} \mathsf{Not}_L(\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(y)M),x)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle d\rangle P,(y)\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)) &\xrightarrow{x} \mathsf{And}_R(\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle d\rangle P,(y)M),\langle b\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle d\rangle P,(y)N),c)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle P,(z)\mathsf{And}^i_L((x)M,y)) &\xrightarrow{x} \mathsf{And}^i_L((x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle P,(z)M),y)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(x)\mathsf{Or}^i_R(\langle a\rangle M,b)) &\xrightarrow{x} \mathsf{Or}^i_R(\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(x)M),b)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle P,(v)\mathsf{Or}_L((x)M,(y)N,z)) &\xrightarrow{x} \mathsf{Or}_L((x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle P,(v)M),(y)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle P,(v)N),z)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)\mathsf{Imp}_R((x)\langle a\rangle M,b)) &\xrightarrow{x} \mathsf{Imp}_R((x)\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle c\rangle P,(y)M),b)\\
\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(z)\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)) &\xrightarrow{x} \mathsf{Imp}_L(\langle a\rangle \mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(z)M),(x)\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle b\rangle P,(z)N),y)
\end{aligned}
$$

Figure 2.11: Cut-reductions for labelled cuts.

Definition 2.6.1 (Local Cut-Elimination Procedure)**:**
The local cut-elimination procedure, $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$, is a reduction system where:

- $\mathcal{T}^{\leftrightarrow}$ is the set of terms well-typed by the rules shown in (2.5) and Figure 2.3, and
- $\xrightarrow{loc}$ consists of the rules for logical, commuting and labelled cuts as well as the rules for garbage collection; that is

$$\xrightarrow{loc} \stackrel{\text{def}}{=} \xrightarrow{l'} \cup \xrightarrow{c''} \cup \xrightarrow{x} \cup \xrightarrow{gc} .$$

As usual, we assume the reductions are closed under context formation. An immediate observation is that $\xrightarrow{loc}$ is complete, in the sense that it eliminates all cuts (routine by inspection of the reduction rules). Another is that $\xrightarrow{loc}$ respects the subject reduction property.

Proposition 2.6.2 (Subject Reduction)**:** Suppose M belongs to $\mathcal{T}^{\leftrightarrow}$ and has the typing judgement $\Gamma \triangleright M \triangleright \Delta$. Let $M \xrightarrow{loc} N$, then $N \in \mathcal{T}^{\leftrightarrow}$ with the typing judgement $\Gamma \triangleright N \triangleright \Delta$.

Proof. By inspection of the reduction rules; analogous to proof of Proposition 2.2.10. □

To prove strong normalisation for $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$, let us first examine if there is a straightforward method inferring this property from the strong normalisation result of $(\mathcal{T}, \xrightarrow{aux})$. A naïve attempt might use the following lemma, often applied successfully in the explicit substitution community (see for example [Bloo, 1997]). It establishes strong normalisation for a reduction system with the help of another one, for which strong normalisation is already known to hold.[6]

Lemma 2.6.3: Let $(\mathcal{R}, \xrightarrow{R})$ and $(\mathcal{S}, \xrightarrow{S})$ be two reduction systems where $\xrightarrow{S}$ consists of two reductions, $\xrightarrow{S_1}$ and $\xrightarrow{S_2}$, and suppose that $(\mathcal{R}, \xrightarrow{R})$ and $(\mathcal{S}, \xrightarrow{S_1})$ are strongly normalising. Let $|\_|^\bullet$ be a translation which maps elements of $\mathcal{S}$ to elements of $\mathcal{R}$ such that (for all $a, b \in \mathcal{S}$):

- if $a \xrightarrow{S_1} b$, then $|a|^\bullet \xrightarrow{R}{}^* |b|^\bullet$
- if $a \xrightarrow{S_2} b$, then $|a|^\bullet \xrightarrow{R}{}^+ |b|^\bullet$

Then we have that $(\mathcal{S}, \xrightarrow{S})$ is strongly normalising.

Proof. Assume for the sake of deriving a contradiction that $(\mathcal{S}, \xrightarrow{S})$ is not strongly normalising. Without loss of generality we may assume that every reduction sequence in $(\mathcal{S}, \xrightarrow{S})$ is of the form

$$a \xrightarrow{S_1}{}^* b \xrightarrow{S_2} c \xrightarrow{S_1}{}^* d \xrightarrow{S_2} \ldots$$

Suppose the above reduction sequence denotes an infinite reduction sequence, then using

[6] We will not use this lemma in this section, but in a later one.

the translation $|\_|^\bullet$ we can map it onto an infinite reduction sequence in $(\mathcal{R}, \xrightarrow{R})$, as shown below.

$$\begin{array}{ccccccccc}
a & \xrightarrow{S_1}{}^{*} & b & \xrightarrow{S_2} & c & \xrightarrow{S_1}{}^{*} & d & \xrightarrow{S_2} & \dots \\
\bullet\downarrow & & \bullet\downarrow & & \bullet\downarrow & & \bullet\downarrow & & \\
|a|^\bullet & \xrightarrow{R}{}^{*} & |b|^\bullet & \xrightarrow{R}{}^{+} & |c|^\bullet & \xrightarrow{R}{}^{*} & |d|^\bullet & \xrightarrow{R}{}^{+} & \dots
\end{array}$$

By assumption we know that every $\xrightarrow{S_1}{}^{*}$-reduction is finite. Thus we know that there must be infinitely many $\xrightarrow{S_2}$-reductions. However, this implies that we have infinitely many $\xrightarrow{R}{}^{+}$-reductions, which contradicts our assumption about $(\mathcal{R}, \xrightarrow{R})$ being strongly normalising. $\square$

Let us return to the question of whether or not $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$ is strongly normalising. It seems as if the lemma given above should be helpful, since it is easy to show that $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{x})$ and $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{gc})$ are strongly normalising (we do not allow, using terminology from explicit substitution calculi, substitution composition). Furthermore, we should be able to find a translation which maps $\xrightarrow{x}$-reductions and $\xrightarrow{gc}$-reductions onto identities, and $\xrightarrow{l'}$- and $\xrightarrow{c''}$-reductions onto a series of $\xrightarrow{aux}$-reductions. Whilst this approach seems plausible, in fact, it fails when using the obvious candidate for the translation, namely the one that replaces labelled cuts with substitutions, as indicated below.

$$\begin{aligned}
|\overleftarrow{\mathsf{Cut}}(\langle a\rangle M, (x)N)|^\bullet &\stackrel{\text{def}}{=} |M|^\bullet\{a := (x)|N|^\bullet\} \\
|\overrightarrow{\mathsf{Cut}}(\langle a\rangle M, (x)N)|^\bullet &\stackrel{\text{def}}{=} |N|^\bullet\{x := \langle a\rangle|M|^\bullet\}
\end{aligned}$$

Using this translation, the proof fails because in $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$ we can perform reductions that cannot be "mirrored" by any reduction in $(\mathcal{T}, \xrightarrow{aux})$. Consider for example the term $M \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)$ with a not free in S. So M is translated to $|S|^\bullet\{a := (x)|T|^\bullet\}$, which is just $|S|^\bullet$ because of the assumption about a. Clearly, there is no reason that any reduction T may perform can be mirrored by $|S|^\bullet$, and consequently, we cannot guarantee that an infinite reduction sequence occurs in (using again terminology from explicit substitution calculi) garbage substitutions. This is how Melliès [1995] constructed a non-terminating reduction sequence for $\lambda\sigma$.

Because of this failure our strong normalisation proof for $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$ is again rather involved. An important fact in this proof is that $\xrightarrow{x}$ is confluent, in contrast to $\xrightarrow{loc}$, which clearly is not.

Lemma 2.6.4: The reduction $\xrightarrow{x}$ is

(i) strongly normalising and

(ii) confluent.

Proof. The measure given on Page 151 implies $[\![M]\!] > [\![N]\!]$ whenever $M \xrightarrow{x} N$. Therefore $\xrightarrow{x}$ must be strongly normalising. Confluence of $\xrightarrow{x}$ follows from local confluence, which can easily be established by inspection of the reduction rules, and strong normalisation of $\xrightarrow{x}$. $\square$

As a consequence of this lemma, we may define the unique x-normal form of a term belonging to $\mathcal{T}^{\leftrightarrow}$.

Definition 2.6.5: The unique x-normal form of a term $M \in \mathcal{T}^{\leftrightarrow}$ is denoted by $|M|_x$.

Before we continue, two quick observations about x-normal forms.

Lemma 2.6.6: If and only if M is an element of $\mathcal{T}$, then $|M|_x \equiv M$.

Proof. If $M \in \mathcal{T}$, then M cannot contain any instance of $\mathsf{C}\overleftarrow{\mathsf{ut}}$ or $\mathsf{C}\overrightarrow{\mathsf{ut}}$, and *vice versa*. □

Lemma 2.6.7: For all unlabelled terms, for example $\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)$, we have

$$|\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)|_x \equiv \mathsf{And}_R(\langle a\rangle |M|_x, \langle b\rangle |N|_x, c)\ .$$

Similarly for the other unlabelled terms.

Proof. By inspection of the $\xrightarrow{x}$-reductions. □

Next, we shall prove that $\xrightarrow{x}$ correctly simulates the substitution operation.

Lemma 2.6.8: For all $M, N \in \mathcal{T}$ we have

(i) $\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle M, (x)N) \xrightarrow{x}{}^{+} M\{a := (x)N\}$

(ii) $\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle N, (x)M) \xrightarrow{x}{}^{+} M\{x := \langle a\rangle N\}$

Proof. Both cases by induction on the structure of M; note we assumed $M, N \in \mathcal{T}$. □

If we apply $|\_|_x$ to the lemma above, we have the following corollary.

Corollary 2.6.9: For all $M, N \in \mathcal{T}$ we thus have

(i) $|\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle M, (x)N)|_x \equiv M\{a := (x)N\}$

(ii) $|\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle M, (x)N)|_x \equiv N\{x := \langle a\rangle M\}$

This corollary concerns subterms belonging to $\mathcal{T}$, but we can now easily generalise it to all terms of $\mathcal{T}^{\leftrightarrow}$.

Lemma 2.6.10: For all $M, N \in \mathcal{T}^{\leftrightarrow}$ we have

(i) $|\mathsf{C}\overleftarrow{\mathsf{ut}}(\langle a\rangle M, (x)N)|_x \ \equiv\ |M|_x\{a := (x)\,|N|_x\}$

(ii) $|\mathsf{C}\overrightarrow{\mathsf{ut}}(\langle a\rangle M, (x)N)|_x \ \equiv\ |N|_x\{x := \langle a\rangle\,|M|_x\}$

Proof. By calculation using Lemma 2.6.6 and Corollary 2.6.9 (see Page 151). □

Now we are in a position to show that the $\xrightarrow{loc}$-reductions project onto $\xrightarrow{aux}$-reductions.

Lemma 2.6.11: For all $M, N \in \mathfrak{T}^{\leftrightarrow}$ if $M \xrightarrow{loc} N$, then $|M|_x \xrightarrow{aux}{}^{*} |N|_x$.

Proof. By induction on the definition of $\xrightarrow{loc}$ (see Page 151). $\square$

As explained earlier, this lemma is not strong enough to prove strong normalisation for $(\mathfrak{T}^{\leftrightarrow}, \xrightarrow{loc})$. In the proof, which we shall give next, we make use of the recursive path ordering theorem by Dershowitz.

Definition 2.6.12 (Recursive Path Ordering, Dershowitz, 1982)**:**[7]
Let $s \equiv f(s_1, \ldots, s_m)$ and $t \equiv g(t_1, \ldots, t_n)$ be terms, then $s >^{rpo} t$,

if and only if

(i)	$s_i \geq^{rpo} t$ for some $i = 1, \ldots, m$	(subterm)

or

(ii)	$f \gg g$ and $s >^{rpo} t_j$ for all $j = 1, \ldots, n$	(decreasing heads)

or

(iii)	$f = g$ and $\{\!\{s_1, \ldots, s_m\}\!\} >^{rpo}_{mult} \{\!\{t_1, \ldots, t_n\}\!\}$	(equal heads)

where $\gg$ is a precedence defined over term constructors, $>^{rpo}_{mult}$ is the extension of $>^{rpo}$ to finite multisets and $\geq^{rpo}$ means $>^{rpo}$ or equivalent up to permutation of subterms.

Theorem 2.6.13 (Dershowitz, 1982)**:** The recursive path ordering $>^{rpo}$ is well-founded, if and only if the precedence $\gg$ is well-founded.

Unfortunately, two problems preclude a direct application of Dershowitz's theorem.

- First, the theorem requires a well-founded precedence for our term constructors. Unfortunately, our reduction rules include the two reductions (written schematically)

$$\begin{array}{rcl} \mathsf{Cut}(\_,\_) & \xrightarrow{c''} & \mathsf{Cu}\overleftarrow{\mathsf{t}}(\_,\_) \\ \mathsf{Cu}\overleftarrow{\mathsf{t}}(\mathsf{Cut}(\_,\_),\_) & \xrightarrow{x} & \mathsf{Cut}(\mathsf{Cu}\overleftarrow{\mathsf{t}}(\_,\_), \mathsf{Cu}\overleftarrow{\mathsf{t}}(\_,\_)) \end{array}$$

  and consequently we have the cycle

$$\begin{tikzcd} \mathsf{Cut} \arrow[r, bend left, "c''"] & \mathsf{Cu}\overleftarrow{\mathsf{t}} \arrow[l, bend left, "x"] \end{tikzcd}$$

  for the outermost term constructors. Bloo and Geuvers [1999] have presented a clever solution for an analogous problem in $\lambda\mathsf{x}$, which we shall adapt for our rewrite system. The essence of this solution is that we take into account that $(\mathfrak{T}, \xrightarrow{aux})$ is strongly normalising. This allows us to order the term constructors according to the maximal number of $\xrightarrow{aux}$-reductions their x-normal form can perform.

[7] $\{\!\{\ldots\}\!\}$ stands for a multiset of terms.

- The second problem arises from the fact that the recursive path ordering theorem applies only to first-order rewrite systems, i.e., no binding operations are allowed. In our term calculus however we have two binding operations: one for names and one for co-names. We solve this problem by introducing another term calculus, denoted by $\mathcal{H}$, for which we can apply this theorem, and then prove strong normalisation for $(\mathcal{T}, \xrightarrow{loc})$ by translation.

The term calculus $\mathcal{H}$ is defined as follows.

Definition 2.6.14: Let $\mathcal{H}$ denote the set of all terms generated by the grammar

$$M, N \quad ::= \quad \star \quad | \quad M \cdot_n N \quad | \quad M\langle N\rangle_n \quad | \quad \langle M\rangle_n N \quad | \quad (\!|M, N|\!) \quad | \quad (\!|M|\!)$$

where n is a natural number. Furthermore, we have the well-founded precedence

$$\_ \cdot_{n+1} \_ \;\gg\; \langle\_\rangle_n\_\,,\; \_\langle\_\rangle_n \;\gg\; \_ \cdot_n \_ \;\gg\; \star,\; (\!|\_|\!),\; (\!|\_,\_|\!)\ .$$

In order to define a translation from terms of $\mathcal{T}^{\leftrightarrow}$ to terms of $\mathcal{H}$, we shall use, as it turns out later, an alternative definition of the set of $\mathcal{T}^{\leftrightarrow}$. This alternative definition is required to strengthen an induction hypothesis.

Definition 2.6.15:

$$\mathcal{T}^{\leftrightarrow}_{<\infty} \quad \stackrel{\text{def}}{=} \quad \left\{ M \;\middle|\; M \in \mathcal{T}^{\leftrightarrow} \text{ and for every subterm } N \text{ of } M,\, |N|_x \in SN_{aux} \right\}$$

Clearly, we have $\mathcal{T}^{\leftrightarrow}_{<\infty} \subseteq \mathcal{T}^{\leftrightarrow}$. The first property we need to show for $\mathcal{T}^{\leftrightarrow}_{<\infty}$ is that it is closed under $\xrightarrow{loc}$-reductions.

Lemma 2.6.16: $\mathcal{T}^{\leftrightarrow}_{<\infty}$ is closed under $\xrightarrow{loc}$-reductions.

Proof. By induction on the definition of $\xrightarrow{loc}$ (see Page 152). □

Next, we shall define a translation from terms of $\mathcal{T}^{\leftrightarrow}_{<\infty}$ to terms of $\mathcal{H}$.

Definition 2.6.17: The translation $\underline{\;\;}: \mathcal{T}^{\leftrightarrow}_{<\infty} \to \mathcal{H}$ is inductively defined by the following clauses (l, m, n are natural numbers).

$$\begin{array}{lclrcl}
\underline{\mathsf{Cut}(\langle a\rangle S, (x)T)} & \stackrel{\text{def}}{=} & \underline{S} \cdot_l \underline{T} & l & \stackrel{\text{def}}{=} & \text{MAXRED}_{aux}(|\mathsf{Cut}(\langle a\rangle S, (x)T)|_x) \\
\underline{\overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)} & \stackrel{\text{def}}{=} & \underline{S}\langle\underline{T}\rangle_m & m & \stackrel{\text{def}}{=} & \text{MAXRED}_{aux}(|\overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)|_x) \\
\underline{\overrightarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)} & \stackrel{\text{def}}{=} & \langle\underline{S}\rangle_n\underline{T} & n & \stackrel{\text{def}}{=} & \text{MAXRED}_{aux}(|\overrightarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)|_x)
\end{array}$$

Otherwise

$$\begin{array}{rcl}
\underline{\mathsf{Ax}(x, a)} & \stackrel{\text{def}}{=} & \star \\
\underline{\mathsf{Not}_R((x)M, a)} & \stackrel{\text{def}}{=} & (\!|\underline{M}|\!) \\
\underline{\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)} & \stackrel{\text{def}}{=} & (\!|\underline{M}, \underline{N}|\!) \\
\underline{\mathsf{Or}^i_R(\langle a\rangle M, b)} & \stackrel{\text{def}}{=} & (\!|\underline{M}|\!) \\
\underline{\mathsf{Imp}_R((x)\langle a\rangle M, b)} & \stackrel{\text{def}}{=} & (\!|\underline{M}|\!)
\end{array}
\qquad
\begin{array}{rcl}
\underline{\mathsf{Not}_L(\langle a\rangle M, x)} & \stackrel{\text{def}}{=} & (\!|\underline{M}|\!) \\
\underline{\mathsf{And}^i_L((x)M, y)} & \stackrel{\text{def}}{=} & (\!|\underline{M}|\!) \\
\underline{\mathsf{Or}_L((x)M, (y)N, z)} & \stackrel{\text{def}}{=} & (\!|\underline{M}, \underline{N}|\!) \\
\underline{\mathsf{Imp}_L(\langle a\rangle M, (x)N, y)} & \stackrel{\text{def}}{=} & (\!|\underline{M}, \underline{N}|\!)
\end{array}$$

Because of the restriction put on $\mathcal{T}^{\leftrightarrow}_{<\infty}$, it is not difficult to prove that this translation is well-defined.

Lemma 2.6.18: The translation $\underline{\ \ }: \mathcal{T}^{\leftrightarrow}_{<\infty} \to \mathcal{H}$ is well-defined.

Proof. Assume $M \in \mathcal{T}^{\leftrightarrow}_{<\infty}$. We have by definition of $\mathcal{T}^{\leftrightarrow}_{<\infty}$ that the x-normal form of every subterm, N, of M is an element in SN_{aux}. Consequently, $\textsc{maxred}_{aux}(|N|_x)$ is a natural number. $\square$

The next lemma is useful for comparing labels of terms belonging to $\mathcal{H}$; it relates two terms of $\mathcal{T}^{\leftrightarrow}_{<\infty}$ according to the number of $\xrightarrow{aux}$-reductions their x-normal form can perform.

Lemma 2.6.19: For all terms $M, N \in \mathcal{T}^{\leftrightarrow}_{<\infty}$ we have

(i) $\textsc{maxred}_{aux}(|M|_x) \geq \textsc{maxred}_{aux}(|N|_x)$, provided $M \xrightarrow{loc} N$.

(ii) $\textsc{maxred}_{aux}(|M|_x) \geq \textsc{maxred}_{aux}(|N|_x)$, provided N is an immediate subterm of M, and M is unlabelled.

(iii) $\textsc{maxred}_{aux}(|M|_x) > \textsc{maxred}_{aux}(|N|_x)$, provided $M \xrightarrow{l'} N$ or $M \xrightarrow{c''} N$ on the outermost level.

Proof. (i) follows from Lemma 2.6.11; for (ii) note that all reductions which $|N|_x$ can perform can be performed by $|M|_x$; (iii) is by simple calculation using Lemma 2.6.7 and the fact that the side conditions put on $\xrightarrow{c''}$ ensures that $|M|_x \xrightarrow{aux} |N|_x$. $\square$

Now we are in the position to prove the (main) lemma that relates every $\xrightarrow{loc}$-reduction to a pair of terms that belong to $\mathcal{H}$ and which are ordered decreasingly according to $>^{rpo}$.

Lemma 2.6.20: For all $M, N \in \mathcal{T}^{\leftrightarrow}_{<\infty}$, if $M \xrightarrow{loc} N$, then $\underline{M} >^{rpo} \underline{N}$.

Proof. By induction on the definition of $\xrightarrow{loc}$ (see Pages 152–155). $\square$

Using this lemma, we can show that every $\xrightarrow{loc}$-reduction sequence starting with a term of $\mathcal{T}$ is terminating.

Lemma 2.6.21: Every $\xrightarrow{loc}$-reduction sequence starting with a term belonging to $\mathcal{T}$ is terminating.

Proof. Suppose for the sake of deriving a contradiction that from M the following infinite reduction sequence starts.

$$M \equiv M_1 \xrightarrow{loc} M_2 \xrightarrow{loc} M_3 \xrightarrow{loc} M_4 \xrightarrow{loc} \dots$$

By Lemma 2.6.6 we have for all subterms N of M that $|N|_x \equiv N$ (M does not contain any labelled cut), and by Theorem 2.3.19 we know that each of them is strongly normalising under $\xrightarrow{aux}$. Consequently, every $\textsc{maxred}_{aux}(|N|_x)$ is finite, and thus M belongs to $\mathfrak{T}^{\leftrightarrow}_{<\infty}$. By Lemmas 2.6.16 and 2.6.20 we have that the infinite reduction sequence starting from M can be mapped onto the decreasing chain

$$\underline{M_1} >^{rpo} \underline{M_2} >^{rpo} \underline{M_3} >^{rpo} \underline{M_4} >^{rpo} \ldots$$

which however contradicts the well-foundedness of $>^{rpo}$. Thus all $\xrightarrow{loc}$-reduction sequences starting with a term that is an element in $\mathfrak{T}$ must terminate. □

Next, we extend this lemma to all terms of $\mathfrak{T}^{\leftrightarrow}$. To do so, we shall first show that for every $M \in \mathfrak{T}^{\leftrightarrow}$ there is a term $N \in \mathfrak{T}$, such that $N \xrightarrow{loc}{}^{*} M$. Because N is an element in $\mathfrak{T}$, we have that N is strongly normalising by the lemma just given, and so M, too, must be strongly normalising.

Lemma 2.6.22: For every term $M \in \mathfrak{T}^{\leftrightarrow}$ with the typing judgement $\Gamma \triangleright M \triangleright \Delta$, there is a term $N \in \mathfrak{T}$ with the typing judgement $\Gamma', \Gamma \triangleright N \triangleright \Delta, \Delta'$ such that $N \xrightarrow{loc}{}^{*} M$.

Proof. We construct N by inductively replacing in M all occurrences of $\overleftarrow{\mathsf{Cut}}$ and $\overrightarrow{\mathsf{Cut}}$ by some instances of Cut (see Page 155). □

Now the strong normalisation theorem is by a simple contradiction argument.

Theorem 2.6.23: The reduction system $(\mathfrak{T}^{\leftrightarrow}, \xrightarrow{loc})$ is strongly normalising.

Proof. Suppose $M \in \mathfrak{T}^{\leftrightarrow}$ is not strongly normalising. Then by the lemma just given there is a term $N \in \mathfrak{T}$ such that $N \xrightarrow{loc}{}^{*} M$. Clearly, if M is not strongly normalising, then so is N, which however contradicts Lemma 2.6.21. Consequently, M must be strongly normalising. □

While Gentzen-like local reductions tend to break strong normalisation, in this section we have shown that a strongly normalising cut-elimination procedure can be obtained by replacing our global operation of proof substitution with appropriate local reductions.

2.7 Notes

Several cut-elimination procedures have been introduced in this chapter, and we have focused on showing, in each case, the property of strong normalisation. In this section we shall take a step back and consider a more global picture. We shall first motivate our use of two slightly different notions of proof substitution. Next, we shall compare our work to the cut-elimination procedure by Dragalin and to the symmetric lambda calculus by Barbanera and Berardi.

2.7.1 Why Two Notions of Proof Substitution?

The proof substitution denoted by [_] was introduced in order to avoid infinite reduction sequences arising from the too liberal definition of Gentzen's reduction rules for commuting cuts. We modified this proof substitution in Section 2.3 so that independent substitutions can commute. The modified variant was referred to as auxiliary proof substitution and denoted by $\{\_\}$. Both substitution operations are applied when instances of commuting cuts are reduced: $[\_]$ is used in the cut-elimination procedure $(\mathcal{T}, \xrightarrow{cut})$ and $\{\_\}$ in $(\mathcal{T}, \xrightarrow{aux})$. In Section 2.6 we derived from $(\mathcal{T}, \xrightarrow{aux})$ the local cut-elimination procedure $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$, where in place of the substitution operation $\{\_\}$, local reduction rules are applied. This cut-elimination procedure could also be derived from $(\mathcal{T}, \xrightarrow{cut})$, in which case we would have a slightly different local procedure. All procedures are complete in the sense of eliminating all cuts from a classical proof. Thus the question arises, why did we introduce two cut-elimination procedures for $\mathcal{T}$, or more specifically why did we introduce two proof substitutions? In the following we will discuss the trade-offs that led us to give the two versions of substitution and cut-elimination.

Whilst our sequent calculus given in Section 2.2 is rather different from Gentzen's sequent calculus LK (all structural rules are implicit), the cut-elimination rules of $(\mathcal{T}, \xrightarrow{cut})$ are very reminiscent of his rules, except that we allow cuts to pass over other cuts. In contrast, the rules of $(\mathcal{T}, \xrightarrow{aux})$ differ somewhat from Gentzen's rules. The differences arise from the two clauses

$$\begin{aligned}\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)M)\{x := \langle b\rangle P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle b\rangle P,(y)M\{x := \langle b\rangle P\})\\ \mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Ax}(x,b))\{b := (y)P\} &\stackrel{\text{def}}{=} \mathsf{Cut}(\langle a\rangle M\{b := (y)P\},(y)P)\end{aligned}$$

which allow us, for example, to reduce the lower cut-instance in the proof

$$\dfrac{\Gamma \triangleright N \triangleright b : B \qquad \dfrac{\dfrac{}{x : B \triangleright \mathsf{Ax}(x,a) \triangleright a : B} \quad y : B \triangleright M \triangleright \Delta}{x : B \triangleright \mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)M) \triangleright \Delta}\,\text{Cut}}{\Gamma \triangleright \mathsf{Cut}(\langle b\rangle N,(x)\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)M)) \triangleright \Delta}\,\text{Cut}$$

in a single step to

$$\dfrac{\Gamma \triangleright N \triangleright b : B \quad y : B \triangleright M \triangleright \Delta}{\Gamma \triangleright \mathsf{Cut}(\langle b\rangle N,(y)M) \triangleright \Delta}\,\text{Cut}\;.$$

This kind of cut-reduction, which "merges" two cuts into a single cut has, as far as we are aware, not been considered elsewhere. Therefore comparing the cut-elimination procedure $(\mathcal{T}, \xrightarrow{aux})$ with traditional cut-elimination procedures, for example, according to which normal forms can be reached from a proof, is rather difficult. Much simpler is the comparison with $(\mathcal{T}, \xrightarrow{cut})$ because its reduction rules are easier to relate to standard reduction rules.

Another, more important reason for presenting $(\mathcal{T}, \xrightarrow{cut})$ is the close correspondence with normalisation in natural deduction. As we shall see in Chapter 3, we have the following correspondences in the $(\supset,\wedge)$-fragment of intuitionistic logic

$$\begin{array}{ccc} M & \xrightarrow{int} & N \\ {\scriptstyle\mathsf{n}}\downarrow & & \downarrow{\scriptstyle\mathsf{n}} \\ M' & \xrightarrow{\beta}{}^{*} & N' \end{array} \qquad\qquad \begin{array}{ccc} S & \xrightarrow{\beta} & T \\ {\scriptstyle\mathsf{s}}\downarrow & & \downarrow{\scriptstyle\mathsf{s}} \\ S' & \xrightarrow{int}{}^{+} & T' \end{array}$$

where $\xrightarrow{int}$ is the intuitionistic version of $\xrightarrow{cut}$ and $|\_|^{\mathsf{n}}$, $|\_|^{\mathsf{s}}$ are slight variants of the standard translations between natural deduction proofs and sequent proofs. Thus $\xrightarrow{int}$ can simulate beta-reduction, which (using $|\_|^{\mathsf{s}}$) the intuitionistic version of $\xrightarrow{aux}$ cannot.

Whilst the proof substitution $[\_]$ leads to a more traditional cut-elimination procedure, the auxiliary proof substitution $\{\_\}$, on the other hand, plays a crucial rôle in our strong normalisation proof: it allows independent substitutions to commute. Indeed, it is difficult to imagine a version of the proof of Lemma 2.3.18 that does not require the commutation of independent substitutions.

Another important point in favour of $\xrightarrow{aux}$ is the fact that if we are interested in the collection of normal forms which can be reached by cut-elimination, then $\xrightarrow{aux}$ has several advantages over $\xrightarrow{cut}$. To calculate a certain normal form, it is possible to apply the rules of $\xrightarrow{aux}$ in a leftmost-outermost strategy. This can be exploited in an implementation, the details of which will be given in Chapter 4. If we restrict $\xrightarrow{cut}$ in the same way, we "lose" some normal forms. By this we mean some normal forms reachable by $\xrightarrow{cut}$ cannot be reached by the corresponding leftmost-outermost strategy. Unfortunately, without such a restriction an implementation of $\xrightarrow{cut}$ is no picnic!

2.7.2 Dragalin's Strong Normalisation Proof for Cut-Elimination

As far as we are aware, Dragalin [1988] gave the first strong normalisation proof for a cut-elimination procedure for classical logic. His procedure is based on Gentzen-like local reduction rules, and thus he had to overcome several difficulties in order to prove strong normalisation. One difficulty arises in situations where the cut-formula is contracted just above the cut-rule, as illustrated in (2.2) on Page 14. Reducing this kind of cut-instance using a Gentzen-like reduction rule yields a proof with two new cuts. As a result, the proof size increases without a decrease of the height of the subproofs above the lowest cut. This is very annoying in a strong normalisation proof, because simple inductions on the proof size or on the proof height are not applicable. Following Gentzen, Dragalin overcame this difficulty by using the multicut-rule in place of the cut-rule. Thus, instead of reducing the cut in (2.2) to a proof with two new cuts, one just removes the contraction and increases the corresponding index of the multicut, as shown below.

$$\dfrac{\dfrac{\Gamma_1 \vdash \Delta_1, C^{n+1}}{\Gamma_1 \vdash \Delta_1, C^{n}}\,\mathrm{Contr}_R \quad C^m, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,\text{Multicut} \quad\longrightarrow\quad \dfrac{\Gamma_1 \vdash \Delta_1, C^{n+1} \quad C^m, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,\text{Multicut}$$

Another difficulty arises from reduction rules that allow (multi)cuts to be permuted with other (multi)cuts: they cause loops, unless some restrictions are imposed. Unfortunately, Dragalin did not consider any of these restricted rules, and without them the correpondence between normalisation and cut-elimination fails. We feel this is a serious shortcoming of his cut-elimination procedure, not just because we "lose" the correspondence with normalisation (see next chapter), but also they are required to interpret Lafont's example as an instance of a non-deterministic choice. Although Lafont's example (translated into Dragalin's system)

$$\dfrac{\dfrac{\vdash B}{\vdash B, C^1}\,\text{Weak}_R \qquad \dfrac{\vdash B}{C^1 \vdash B}\,\text{Weak}_L}{\dfrac{\vdash B, B}{\vdash B}\,\text{Contr}_R}\,\text{Multicut}$$

can reduce non-deterministically to either subproof, the lower cut in the "nested" version of Lafont's example

$$\dfrac{\dfrac{\dfrac{\vdash B}{\vdash B, C^1}\,\text{Weak}_R \qquad \dfrac{\dfrac{\vdash B}{\vdash B, D^1}\,\text{Weak}_R}{C^1 \vdash B, D^1}\,\text{Weak}_L}{\vdash B, B, D^1}\,\text{Multicut} \qquad \dfrac{\vdash B}{D^1 \vdash B}\,\text{Weak}_L}{\dfrac{\vdash B, B, B}{\vdash B}\,\text{Contr}_R}\,\text{Multicut}$$

cannot. Using Dragalin's reduction rules, the lower multicut can reduce to its right subproof, but not immediately to its left one. For this the upper multicut has to reduce first. Consequently, the lower multicut is not a genuine non-deterministic choice as it depends on another multicut. Clearly, to remedy this problem we need reduction rules that permute the lower multicut with the upper one. As mentioned earlier, the naïve formulation of these reduction rules, e.g.

$$\dfrac{\dfrac{\ldots \vdash \ldots \qquad \ldots \vdash \ldots}{\ldots \vdash \ldots}\,\text{Multicut}_2 \qquad \ldots \vdash \ldots}{\ldots \vdash \ldots}\,\text{Multicut}_1$$

$$\longrightarrow \qquad \dfrac{\dfrac{\ldots \vdash \ldots \qquad \ldots \vdash \ldots}{\ldots \vdash \ldots}\,\text{Multicut}_1 \qquad \dfrac{\ldots \vdash \ldots \qquad \ldots \vdash \ldots}{\ldots \vdash \ldots}\,\text{Multicut}_1}{\ldots \vdash \ldots}\,\text{Multicut}_2$$

breaks strong normalisation.

As far as I have been able to ascertain, the cut-elimination procedure given in Section 2.6 is the only strongly normalising cut-elimination procedure that is both local and allows cut-instances to be permuted with other cut-instances.[8] Of course we can easily reformulate Dragalin's cut-elimination procedure, along the lines of $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$, to include reduction rules that permute multicuts with other multicuts so that strong normalisation still holds. But a strong normalisation proof is then much harder: these rules do not decrease the proof size or proof height. It is not obvious how Dragalin's proof method can be adapted to prove strong normalisation in this case.

[8]Zucker [1974] obtained a strongly normalising cut-elimination procedure for intuitionistic logic with local reduction rules by considering *proper* reductions, i.e., non-repeating and termination reductions, but he did not restrict his reduction rules accordingly.

2.7.3 Comparison with the Symmetric Lambda Calculus

In this section we compare our work with the symmetric lambda calculus, λ^{Sym} for short, developed by Barbanera and Berardi [1994]. This comparison is worthwhile because we adapted their proof technique based on the notion of symmetric reducibility candidates, and moreover they also take the view that cut-elimination in classical logic corresponds via the Curry-Howard correspondence to non-deterministic computation [Barbanera et al., 1997].

The formulae of λ^{Sym} are given by the grammar

$$B, C ::= A \;\mid\; A^{\perp} \;\mid\; B{\wedge}C \;\mid\; B{\vee}C$$

where A stands for atomic formulae. The negation in λ^{Sym} is not a primitive connective, as in our sequent calculus, instead it is defined by the clauses:

$$\begin{array}{rclcrcl} (A)^{\perp} & \stackrel{\text{def}}{=} & A^{\perp} & \qquad & (B{\wedge}C)^{\perp} & \stackrel{\text{def}}{=} & B^{\perp} \vee C^{\perp} \\ (A^{\perp})^{\perp} & \stackrel{\text{def}}{=} & A & \qquad & (B{\vee}C)^{\perp} & \stackrel{\text{def}}{=} & B^{\perp} \wedge C^{\perp} \end{array}$$

The terms of λ^{Sym} are given by the grammar

$$M, N ::= x \;\mid\; \langle M, N\rangle \;\mid\; \sigma_i(M) \;\mid\; \lambda x.M \;\mid\; M \star N$$

where x is taken from a set of variables. There are two kinds of typing judgements in λ^{Sym}, *viz.*

$$\Gamma \triangleright M : B \qquad \text{and} \qquad \Gamma \triangleright M : \perp \tag{2.6}$$

where in the latter $\perp$ is a distinguished atomic formula. In the sequel we are only interested in well-typed terms, that means those for which there is a context Γ and a formula B such that either $\Gamma \triangleright M : B$ or $\Gamma \triangleright M : \perp$ is derivable given the inference rules shown in Figure 2.12. Because negation is not a primitive connective, the inference rules of λ^{Sym} are quite hard to read, in particular the inference rule $\star$. An example may clarify this inference rule. Consider the following λ^{Sym}-proof.

$$\dfrac{\dfrac{\overline{x : B \triangleright x : B} \quad \overline{y : C \triangleright y : C}}{x : B, y : C \triangleright \langle x, y\rangle : B{\wedge}C}\,\langle -, -\rangle \qquad \dfrac{\overline{z : B^{\perp} \triangleright z : B^{\perp}}}{z : B^{\perp} \triangleright \sigma_1(z) : B^{\perp} \vee C^{\perp}}\,\sigma_1}{x : B, y : C, z : B^{\perp} \triangleright \langle x, y\rangle \star \sigma_1(z) : \perp}\,\star \tag{2.7}$$

The last step in this proof is a valid instance of $\star$, since $(B{\wedge}C)^{\perp}$ is defined as $B^{\perp} \vee C^{\perp}$.

Barbanera and Berardi call a term of the form $M \star N$ *symmetric application*, because if $M \star N$ is well-typed, then so is $N \star M$. However, one can also regard $\star$ as an instance of the cut-rule. To illustrate this, let us reformulate the proof just given in our sequent

$$\overline{x:B,\Gamma \triangleright x:B}$$

$$\frac{\Gamma \triangleright M:B \quad \Gamma \triangleright N:C}{\Gamma \triangleright \langle M,N\rangle : B\wedge C}\ \langle -,-\rangle \qquad \frac{\Gamma \triangleright M:B_i}{\Gamma \triangleright \sigma_i(M):B_1\vee B_2}\ \sigma_i$$

$$\frac{x:B,\Gamma \triangleright M:\bot}{\Gamma \triangleright \lambda x.M:B^{\bot}}\ \lambda \qquad \frac{\Gamma_1 \triangleright M:B \quad \Gamma_2 \triangleright N:B^{\bot}}{\Gamma_1,\Gamma_2 \triangleright M\star N:\bot}\ \star$$

Figure 2.12: Term assignment for the symmetric lambda calculus.

calculus. Here is the sequent proof that corresponds to (2.7) (for brevity we omit all labels).

$$\dfrac{\dfrac{\overline{B\vdash B} \quad \overline{C\vdash C}}{B,C\vdash B\wedge C}\ \wedge_R \qquad \dfrac{\dfrac{\overline{B\vdash B}}{B,\neg B\vdash}\ \neg_L}{B\wedge C,\neg B\vdash}\ \wedge_{L_1}}{B,C,\neg B\vdash}\ \text{Cut}$$

As the reader can see, even if the σ_1-rule introduces an $\vee$, it corresponds to an instance of the $\wedge_{L_1}$-rule, and the $\star$-rule is clearly a cut.

There are a number of reduction rules associated with λ^{Sym}, some of which are given below.

$$\begin{array}{rcl} \lambda x.M\star N & \xrightarrow{sym} & M[x:=N] \\ M\star \lambda x.N & \xrightarrow{sym} & N[x:=M] \\ \langle M_1,M_2\rangle \star \sigma_i(N) & \xrightarrow{sym} & M_i\star N \\ \sigma_i(N)\star\langle M_1,M_2\rangle & \xrightarrow{sym} & N\star M_i \end{array} \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\}\quad i=1,2$$

As usual, the expression $M[x:=N]$ stands for the substitution of the variable x. According to the reduction rules of λ^{sym}, the term $\lambda x.M\star\lambda y.N$ reduces to either M or N assuming that x and y are not free variables in M and N, respectively. So the reduction system of λ^{Sym} is non-deterministic, analogous to our cut-elimination procedure for classical logic.

Because of the close correspondence with our sequent calculus, one may reasonably ask whether our strong normalisation results for $(\mathcal{T},\xrightarrow{cut})$ and $(\mathcal{T},\xrightarrow{aux})$ follow from Barbanera and Berardi's strong normalisation result, for example, by a simple translation. However, it does not appear that such a simple translation exists. Primarily this is because our term constructors include multiple binders; for example, the term constructor $\mathsf{Imp}_R((x)\langle a\rangle M,b)$ and $\mathsf{And}'_L((x)(y)M,z)$. Owing to the specific form of the typing judgements shown in (2.6), where only a single formula on the right-hand side is permitted, it is difficult to imagine a translation that encodes the typing judgements of these term constructors into λ^{Sym}. Another obstacle to a proof by translation are our cut-reductions that replace a single cut-instance with two nested cut-instances. In fact, we had to extend the proof technique of Barbanera and Berardi to deal with such cut-reductions. It seems highly unlikely that this dynamic behaviour of our cut-elimination procedure could be captured by a simple translation.

Even when restricting our calculus to just the $(\wedge, \vee)$-fragment, i.e., to the fragment where term constructors bind only a single name or co-name and where reduction rules do not create nested cut-instances, the strong normalisation proof by translation fails using the natural candidate, $|\_|^{\mathsf{Sym}}$, as translation. The clauses of $|\_|^{\mathsf{Sym}}$ are given below.

$$\begin{array}{rcl}
|\mathsf{Ax}(x,a)|^{\mathsf{Sym}} & \stackrel{\text{def}}{=} & x \star a \\
|\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)|^{\mathsf{Sym}} & \stackrel{\text{def}}{=} & \langle \lambda a.|M|^{\mathsf{Sym}}, \lambda b.|N|^{\mathsf{Sym}}\rangle \star c \\
|\mathsf{And}_L^i((x)M, y)|^{\mathsf{Sym}} & \stackrel{\text{def}}{=} & \sigma_i(\lambda x.|M|^{\mathsf{Sym}}) \star y \\
|\mathsf{Or}_R^i(\langle a\rangle M, b)|^{\mathsf{Sym}} & \stackrel{\text{def}}{=} & \sigma_i(\lambda a.|M|^{\mathsf{Sym}}) \star b \\
|\mathsf{Or}_L((x)M, (y)N, z)|^{\mathsf{Sym}} & \stackrel{\text{def}}{=} & \langle \lambda x.|M|^{\mathsf{Sym}}, \lambda y.|N|^{\mathsf{Sym}}\rangle \star z \\
|\mathsf{Cut}(\langle a\rangle M, (x)N)|^{\mathsf{Sym}} & \stackrel{\text{def}}{=} & \lambda a.|M|^{\mathsf{Sym}} \star \lambda x.|N|^{\mathsf{Sym}}
\end{array}$$

The translation respects the typing judgements of $\mathcal{T}^{(\wedge,\vee)}$, as stated in the following proposition.

Proposition 2.7.1: If $M \in \mathcal{T}^{(\wedge,\vee)}$ with the typing judgement $\Gamma \triangleright M \triangleright \Delta$, then $|M|^{\mathsf{Sym}}$ is well-typed in λ^{Sym} with the typing judgement $\Gamma, \Delta^{\perp} \triangleright |M|^{\mathsf{Sym}} : \perp$, where $\Delta^{\perp}$ is taken to mean that all formulae of Δ are negated.

Proof. By induction on the structure of M. □

But $|\_|^{\mathsf{Sym}}$ does *not* respect the reductions in the system $(\mathcal{T}^{(\wedge,\vee)}, \xrightarrow{cut})$. To give a counter-example, consider the term $\mathsf{Cut}(\langle a\rangle M, (x)N)$ and assume M introduces a, but not freshly (i.e., there is an implicit contraction). According to our reduction rules this term can reduce as follows.

$$\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{cut} M[a := \langle x\rangle N]$$

Since M introduces a, the reduct $M[a := \langle x\rangle N]$ is of the form $\mathsf{Cut}(\langle a\rangle \ldots, (x)N)$, meaning that the outermost cut-instance does not change. Given the translation $|\_|^{\mathsf{Sym}}$, no reduction of λ^{Sym} can "mirror" this behaviour. The existence of a more complicated translation of our sequent proofs into λ^{Sym}, which gives a strong normalisation proof for $\mathcal{T}^{(\wedge,\vee)}$, is possible, but we conjecture unlikely.

Chapter 3

Natural Deduction Calculi

> I intended first to set up a formal system which comes as close as possible to actual reasoning. The result was a "calculus of natural deduction".
>
> —G. Gentzen
> in Investigations into Logical Deductions, 1935.

Natural deduction calculi, like sequent calculi, were invented by Gentzen [1935]. Whilst sequent calculi offered a convenient way to state his *Hauptsatz*, natural deduction calculi offered a convenient approach to reasoning by mimicking the reasoning employed by mathematicians. Prawitz [1965] proved later that a version of the cut-elimination theorem also holds for NJ and NK, Gentzen's natural deduction calculi for intuitionistic and classical logic, respectively.

In this chapter we shall study the striking similarities between natural deduction calculi and sequent calculi. In particular, we shall show a close correspondence between cut-elimination and normalisation using slight variants of the standard translations between natural deduction proofs and sequent proofs. This gives, for example, in the classical case the correspondences illustrated with the diagrams

$$\begin{array}{ccc} S & \xrightarrow{\kappa} & T \\ {\scriptstyle \mathsf{S}}\downarrow & & \downarrow{\scriptstyle \mathsf{S}} \\ S' & \xrightarrow{cut}{}^{+} & T' \end{array} \qquad\qquad \begin{array}{ccc} M & \xrightarrow{cut} & N \\ {\scriptstyle \mathsf{N}}\downarrow & & \downarrow{\scriptstyle \mathsf{N}} \\ M' & \xrightarrow{\kappa}{}^{*} & N' \end{array}$$

where $|\_|^{\mathsf{S}}$ is a translation from natural deduction proofs to sequent proofs, $|\_|^{\mathsf{N}}$ is a translation from sequent proofs to natural deduction proofs, and $\xrightarrow{\kappa}$ is a normalisation procedure for our natural deduction calculus. In contrast to earlier work, we obtain such tight correspondences because of the fewer constraints in our cut-elimination procedures.

We shall begin with showing the correspondence between our intuitionistic cut-elimination procedure and the standard normalisation procedure of NJ. This part will closely follow existing work. The next step will be to extend the correspondence to classical logic. Unfortunately, it is very impractical to do the extension using Gentzen's natural deduction calculus NK, because this calculus requires double negation translations for encoding classical proofs. So instead of using NK, we shall introduce a variant of the natural deduction calculus developed by Boričić [1985] and give normalisation rules inspired by the rules Parigot [1992] defined for his $\lambda\mu$-calculus. No claim is made about superiority of our natural deduction calculus over other natural deduction calculi for classical logic.

3.1 Intuitionistic Natural Deduction

Like sequent proofs, natural deduction proofs are written in a tree-like fashion, and two styles of writing them can be identified in the literature.

- One, used for example in [Gentzen, 1935] and in the classic work on natural deduction [Prawitz, 1965], is to write trees of the form

$$\begin{matrix}\Gamma\\ \vdots\\ B\end{matrix}$$

  where the formula B is the conclusion and Γ stands for a finite number of assumption packets. The assumption packets are multisets containing formulae all of which are annotated with a unique (natural number) label. The inference rules that discharge, or "close", assumption packets are annotated with the labels of the packets they discharge.

- The other, used for example in [Gentzen, 1936], is to write natural deduction proofs in "sequent-style" where in every stage of the proof the undischarged assumption packets are recorded explicitly. Thus the proofs are of the form

$$\begin{matrix}\vdots\\ \Gamma \vdash B\end{matrix}$$

  Throughout this chapter we shall use the sequent-style for writing natural deduction proofs. We shall take Γ to be a set of labelled formulae analogous to the left-contexts defined in connection with sequent calculi. However in this section and in the next, the standard terminology for natural deduction will be adopted, and thus the labels are referred to as *variables* (these two sections deal with the propositional fragment of intuitionistic logic, only; so no confusion can occur with the notion of variables introduced in Section 2.4 for first-order classical logic).

Figure 3.1 gives the inference rules of Gentzen's intuitionistic natural deduction calculus, called NJ.[1] The inference rules come in two flavours, those for *introducing* a connective and those for *eliminating* a connective. Writing the proofs in sequent-style means that the inference rules of natural deduction are solely concerned with manipulating formulae in the succedent, unlike in sequent calculi where inference rules manipulate formulae in both the antecedent and the succedent. This is why it is often said that natural deduction calculi are asymmetric.

There are well-known translations going back to Gentzen [1935] between NJ-proofs and intuitionistic sequent proofs. Below, we shall define the translation $|\_|^{\mathsf{s}}$, which maps NJ-proofs to sequent proofs, and the translation $|\_|^{\mathsf{n}}$, which maps sequent proofs to NJ-proofs. It will be convenient to define both over terms, rather than proofs. One slight complication arises from the fact that we have chosen intuitionistic sequents to be of the form $\Gamma \vdash a : B$ (see Page 46). This choice spared us the work of having to define specific cut-elimination rules for intuitionistic sequent proofs, but it causes the translation

[1] We are not concerned here with the quantifiers and negation.

$$\overline{x\!:\!B,\Gamma \vdash B}$$

$$\frac{x\!:\!B,\Gamma \vdash C}{\Gamma \vdash B\supset C}\supset_I \qquad \frac{\Gamma \vdash B\supset C \quad \Gamma \vdash B}{\Gamma \vdash C}\supset_E$$

$$\frac{\Gamma \vdash B \quad \Gamma \vdash C}{\Gamma \vdash B\wedge C}\wedge_I \qquad \frac{\Gamma \vdash B_1\wedge B_2}{\Gamma \vdash B_i}\wedge_{E_i}$$

$$\frac{\Gamma \vdash B_i}{\Gamma \vdash B_1\vee B_2}\vee_{I_i} \qquad \frac{\Gamma \vdash B\vee C \quad x\!:\!B,\Gamma \vdash D \quad y\!:\!C,\Gamma \vdash D}{\Gamma \vdash D}\vee_E$$

Figure 3.1: Natural deduction calculus for intuitionistic logic.

$|\_|^{\mathsf{s}}$ to be slightly messy: we have to explicitly record the co-name of the formula in the succedent. Before giving the definition of $|\_|^{\mathsf{s}}$ and $|\_|^{\mathsf{n}}$, we shall first illustrate the translations on the level of proofs.

Axioms: The sequent proof

$$\overline{x\!:\!B,\Gamma \vdash a\!:\!B}$$

is translated to

$$\overline{x\!:\!B,\Gamma \vdash B}\ .$$

The reverse translation is analogous, but we have to record, as mentioned above, the co-name a.

Right Rules: A sequent proof ending with an instance of the $\wedge_R$-rule

$$\frac{\begin{array}{c}\pi_1\\ \Gamma \vdash a\!:\!B\end{array} \quad \begin{array}{c}\pi_2\\ \Gamma \vdash b\!:\!C\end{array}}{\Gamma \vdash c\!:\!B\wedge C}\wedge_R$$

is translated to

$$\frac{\begin{array}{c}|\pi_1|^{\mathsf{n}}\\ \Gamma \vdash B\end{array} \quad \begin{array}{c}|\pi_2|^{\mathsf{n}}\\ \Gamma \vdash C\end{array}}{\Gamma \vdash B\wedge C}\wedge_I\ .$$

Left Rules: A sequent proof ending with an instance of the $\vee_L$-rule

$$\frac{\begin{array}{c}\pi_1\\ x\!:\!B,\Gamma \vdash c\!:\!D\end{array} \quad \begin{array}{c}\pi_2\\ y\!:\!C,\Gamma \vdash c\!:\!D\end{array}}{z\!:\!B\vee C,\Gamma \vdash c\!:\!D}\vee_L$$

is translated to

$$\dfrac{\dfrac{}{z:B\vee C,\Gamma\vdash B\vee C} \quad \begin{matrix}|\pi_1|^{\mathsf{n}}\\ x:B,z:B\vee C,\Gamma\vdash D\end{matrix} \quad \begin{matrix}|\pi_2|^{\mathsf{n}}\\ y:C,z:B\vee C,\Gamma\vdash D\end{matrix}}{z:B\vee C,\Gamma\vdash D}\,\vee_E$$

where $z:B\vee C$ might need to be added to $|\pi_1|^{\mathsf{n}}$ or to $|\pi_2|^{\mathsf{n}}$.

The Cut-Rule: A sequent proof ending with an instance of the cut-rule

$$\dfrac{\begin{matrix}\pi_1\\ \Gamma_1\vdash a:B\end{matrix} \quad \begin{matrix}\pi_2\\ x:B,\Gamma_2\vdash b:C\end{matrix}}{\Gamma_1,\Gamma_2\vdash b:C}\,\text{Cut}$$

is translated to

$$\dfrac{\begin{matrix}|\pi_1|^{\mathsf{n}}\\ \Gamma_1\vdash a:B\end{matrix} \quad \begin{matrix}|\pi_2|^{\mathsf{n}}\\ x:B,\Gamma_2\vdash b:C\end{matrix}}{\Gamma_1,\Gamma_2\vdash b:C}\,\text{Subst}\ .$$

There are two possible ways of dealing with the substitution: either it is performed immediately, or it is "suspended" and performed later. We shall employ the first method and mention some problems with the second.

Introduction Rules: An NJ-proof ending with an instance of the $\wedge_I$-rule

$$\dfrac{\begin{matrix}\mathcal{D}_1\\ \Gamma\vdash B\end{matrix} \quad \begin{matrix}\mathcal{D}_2\\ \Gamma\vdash C\end{matrix}}{\Gamma\vdash B\wedge C}\,\wedge_I$$

is translated to

$$\dfrac{\begin{matrix}|\mathcal{D}_1|^{\mathsf{s}}_a\\ \Gamma\vdash a:B\end{matrix} \quad \begin{matrix}|\mathcal{D}_2|^{\mathsf{s}}_b\\ \Gamma\vdash b:C\end{matrix}}{\Gamma\vdash c:B\wedge C}\,\wedge_R$$

where c is a fresh co-name.

Elimination Rules: An NJ-proof ending with an instance of the $\vee_E$-rule

$$\dfrac{\begin{matrix}\mathcal{D}_1\\ \Gamma\vdash B\vee C\end{matrix} \quad \begin{matrix}\mathcal{D}_2\\ x:B,\Gamma\vdash D\end{matrix} \quad \begin{matrix}\mathcal{D}_3\\ y:C,\Gamma\vdash D\end{matrix}}{\Gamma\vdash D}\,\vee_E$$

is translated to

$$\dfrac{\begin{matrix}|\mathcal{D}_1|^{\mathsf{s}}_a\\ \Gamma\vdash a:B\vee C\end{matrix} \quad \dfrac{\begin{matrix}|\mathcal{D}_2|^{\mathsf{s}}_c\\ x:B,\Gamma\vdash c:D\end{matrix} \quad \begin{matrix}|\mathcal{D}_3|^{\mathsf{s}}_c\\ y:C,\Gamma\vdash c:D\end{matrix}}{z:B\vee C,\Gamma\vdash c:D}\,\vee_L}{\Gamma\vdash c:D}\,\text{Cut}$$

where z is a fresh name. This is the standard translation for the $\vee_E$-rule; however, we shall slightly modify this translation by using our proof substitution instead of the cut-rule.

As mentioned earlier, it will be convenient to define the translations $|\_|^{\mathsf{s}}$ and $|\_|^{\mathsf{n}}$ over terms. To do so, NJ-proofs will be annotated with lambda terms.

Definition 3.1.1 (Raw Lambda Terms)**:** The set of raw lambda terms is defined by the grammar

M, N, P	$::=$	x	variable
	$\mid$	$\lambda x\!:\!B.M$	implication-introduction
	$\mid$	$M\,N$	implication-elimination
	$\mid$	$\langle M, N\rangle$	and-introduction
	$\mid$	$\mathtt{fst}(M)$	and^1-elimination
	$\mid$	$\mathtt{snd}(M)$	and^2-elimination
	$\mid$	$\mathtt{inl}(M)$	or^1-introduction
	$\mid$	$\mathtt{inr}(M)$	or^2-introduction
	$\mid$	$\mathtt{case}(P, \lambda x\!:\!B.M, \lambda y\!:\!C.N)$	or-elimination

Let us briefly describe the notions and conventions we shall use for lambda terms: the notions of bound and free variables will be as usual; for brevity we shall often omit the typings on binders and simply write $\lambda x.M$; a Barendregt-style naming convention for variables will always be observed; and lambda terms will be regarded as equivalent up to renaming of bound variables. A pleasing consequence of these conventions is that the notion of variable substitution can be defined without worrying about clashes and capture of variables. Since it will always be clear from the context whether we deal with a proof substitution or a variable substitution, no confusion will occur when we employ the same shorthand notation for both substitutions; that is $[\sigma]$.

To annotate natural deduction proofs with lambda terms, we shall replace every sequent of the form $\Gamma \vdash B$ with a typing judgement of the form $\Gamma \triangleright M : B$ in which Γ is a context, M is a lambda term and B is a formula. In the sequel we are interested in only simply-typed lambda terms, i.e., those for which there is a context Γ and a formula B such that $\Gamma \triangleright M : B$ is derivable given the inferences shown in Figure 3.2. The following set contains all simply-typed lambda terms.

$$\Lambda \stackrel{\text{def}}{=} \left\{ M \;\middle|\; \Gamma \triangleright M : B \text{ is derivable using the rules given in Figure 3.2} \right\}$$

Now we return to the translations $|\_|^{\mathsf{s}}$ and $|\_|^{\mathsf{n}}$ whose clauses are given in Figures 3.3 and 3.4, respectively. Clearly, we expect that both translations respect typing judgements.

Proposition 3.1.2: (i) If $M \in \Lambda$ with the typing judgement $\Gamma \triangleright M : B$, then for all co-names a we have $|M|^{\mathsf{s}}_a \in \mathfrak{I}$ with the typing judgement $\Gamma \triangleright |M|^{\mathsf{s}}_a \triangleright a : B$.

(ii) If $M \in \mathfrak{I}$ with the typing judgement $\Gamma \triangleright M \triangleright a : B$, then we have $|M|^{\mathsf{n}} \in \Lambda$ with the typing judgement $\Gamma \triangleright |M|^{\mathsf{n}} : B$.

$$\frac{}{x:B,\Gamma \triangleright x:B}$$

$$\frac{x:B,\Gamma\triangleright M:C}{\Gamma\triangleright \lambda x.M:B\supset C}\supset_I \qquad\qquad \frac{\Gamma\triangleright M:B\supset C \quad \Gamma\triangleright N:B}{\Gamma\triangleright M\,N:C}\supset_E$$

$$\frac{\Gamma\triangleright M:B \quad \Gamma\triangleright N:C}{\Gamma\triangleright \langle M,N\rangle:B\wedge C}\wedge_I \qquad \frac{\Gamma\triangleright M:B_1\wedge B_2}{\Gamma\triangleright \mathtt{fst}(M):B_1}\wedge_{E_1} \quad \frac{\Gamma\triangleright M:B_1\wedge B_2}{\Gamma\triangleright \mathtt{snd}(M):B_2}\wedge_{E_2}$$

$$\frac{\Gamma\triangleright M:B_1}{\Gamma\triangleright \mathtt{inl}(M):B_1\vee B_2}\vee_{I_1} \qquad \frac{\Gamma\triangleright M:B_2}{\Gamma\triangleright \mathtt{inr}(M):B_1\vee B_2}\vee_{I_2}$$

$$\frac{\Gamma\triangleright P:B\vee C \quad x:B,\Gamma\triangleright M:D \quad y:C,\Gamma\triangleright N:D}{\Gamma\triangleright \mathtt{case}(P,\lambda x.M,\lambda y.N):D}\vee_E$$

Figure 3.2: Term assignment for intuitionistic natural deduction proofs.

Proof. Both by routine induction on the structure of M. ☐

Remark 3.1.3: It is well-known that $|\_|^{\mathsf{n}}$, the translation from sequent proofs to NJ-proofs, is many-to-one. Take for example the sequent proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}}{A\wedge B\vdash A}\wedge_{L_1} \qquad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\,\text{Cut}}{\vdash (A\wedge B)\supset(A\wedge A)}\supset_R$$

and the two normal forms

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\wedge_{L_1}}{\vdash (A\wedge B)\supset(A\wedge A)}\supset_R \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}}{A\wedge B\vdash A}\wedge_{L_1} \quad \dfrac{\dfrac{}{A\vdash A}}{A\wedge B\vdash A}\wedge_{L_1}}{A\wedge B\vdash A\wedge A}\wedge_R}{\vdash (A\wedge B)\supset(A\wedge A)}\supset_R$$

reachable from it (for brevity we omitted all labels). All three sequent proofs are mapped by $|\_|^{\mathsf{n}}$ to the same lambda term, namely

$$\lambda x.\langle \mathtt{fst}(x),\mathtt{snd}(x)\rangle\ .$$

Next we shall define the reduction rules for NJ-proofs. Recall that one of the motivations of cut-elimination are to transform a proof containing cuts to one in normal form satisfying the subformula property (see Definition 2.1.1). Whereas in sequent calculi one can check by simple inspection of a proof whether the subformula property holds, this is

$$
\begin{array}{lrcl}
(1) & |x|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & \mathsf{Ax}(x,c) \\
(2) & |\lambda x.M|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & \mathsf{Imp}_R((x)\langle a\rangle\,|M|^{\mathsf{s}}_a, c) \\
(3) & |\langle M,N\rangle|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & \mathsf{And}_R(\langle a\rangle\,|M|^{\mathsf{s}}_a, \langle b\rangle\,|N|^{\mathsf{s}}_b, c) \\
(4) & |\mathtt{inl}(M)|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & \mathsf{Or}^1_R(\langle a\rangle\,|M|^{\mathsf{s}}_a, c) \\
(5) & |\mathtt{inr}(M)|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & \mathsf{Or}^2_R(\langle a\rangle\,|M|^{\mathsf{s}}_a, c) \\
\\
(6) & |M\ N|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & |M|^{\mathsf{s}}_a[a := (y)\mathsf{Imp}_L(\langle b\rangle|N|^{\mathsf{s}}_b, (z)\mathsf{Ax}(z,c), y)] \\
(7) & |\mathtt{fst}(M)|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & |M|^{\mathsf{s}}_a[a := (y)\mathsf{And}^1_L((x)\mathsf{Ax}(x,c), y)] \\
(8) & |\mathtt{snd}(M)|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & |M|^{\mathsf{s}}_a[a := (y)\mathsf{And}^2_L((x)\mathsf{Ax}(x,c), y)] \\
(9) & |\mathtt{case}(P,\lambda x.M,\lambda y.N)|^{\mathsf{s}}_c & \stackrel{\text{def}}{=} & |P|^{\mathsf{s}}_a[a := (z)\mathsf{Or}_L((x)|M|^{\mathsf{s}}_c, (y)|N|^{\mathsf{s}}_c, z)]
\end{array}
$$

where in: (6) $y \notin FN(\langle b\rangle|N|^{\mathsf{s}}_b)$
(9) $z \notin FN((x)|M|^{\mathsf{s}}_c, (y)|N|^{\mathsf{s}}_c)$

Figure 3.3: Translation from NJ-proofs to intuitionistic sequent proofs.

$$
\begin{array}{lrcl}
(1) & |\mathsf{Ax}(x,a)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & x \\
(2) & |\mathsf{Cut}(\langle a\rangle M, (x)N)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & |N|^{\mathsf{n}}[x := |M|^{\mathsf{n}}] \\
\\
(3) & |\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & \langle |M|^{\mathsf{n}}, |N|^{\mathsf{n}}\rangle \\
(4) & |\mathsf{Or}^1_R(\langle a\rangle M, b)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & \mathtt{inl}(|M|^{\mathsf{n}}) \\
(5) & |\mathsf{Or}^2_R(\langle a\rangle M, b)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & \mathtt{inr}(|M|^{\mathsf{n}}) \\
(6) & |\mathsf{Imp}_R((x)\langle a\rangle M, c)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & \lambda x.|M|^{\mathsf{n}} \\
\\
(7) & |\mathsf{And}^1_L((x)M, y)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & |M|^{\mathsf{n}}[x := \mathtt{fst}(y)] \\
(8) & |\mathsf{And}^2_L((x)M, y)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & |M|^{\mathsf{n}}[x := \mathtt{snd}(y)] \\
(9) & |\mathsf{Or}_L((x)M, (y)N, z)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & \mathtt{case}(z, \lambda x.|M|^{\mathsf{n}}, \lambda y.|N|^{\mathsf{n}}) \\
(10) & |\mathsf{Imp}_L(\langle a\rangle M, (x)N, y)|^{\mathsf{n}} & \stackrel{\text{def}}{=} & |N|^{\mathsf{n}}[x := (y\ |M|^{\mathsf{n}})]
\end{array}
$$

Figure 3.4: Translation from intuitionistic sequent proofs to NJ-proofs.

more difficult in NJ. Consider the proof fragment where an implication is introduced and eliminated immediately afterwards.

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ x:B,\Gamma\vdash C\end{matrix}}{\Gamma\vdash B\supset C}\supset_I \quad \begin{matrix}\mathcal{D}_2\\ \Gamma\vdash B\end{matrix}}{\Gamma\vdash C}\supset_E \tag{3.1}$$

Here the formula $B\supset C$ violates the subformula property. However, we expect that $\Gamma\vdash C$ is provable without introducing this formula. Indeed, we can construct a proof roughly as follows: some leaves in $\mathcal{D}_1$ are of the form $\overline{x:B,\Gamma,\Delta\vdash B}$—replace all those instances with the natural deduction proof of $\Delta,\Gamma\vdash B$ that is obtained by adding Δ to all sequents of $\mathcal{D}_2$ and by possibly renaming some variables; finally remove all assumptions $x:B$ in $\mathcal{D}_1$. Below, we shall introduce the reduction rules, commonly referred to as *beta-reductions*, realising this kind of proof transformations.

Definition 3.1.4 (Beta-Reductions)**:**

$$\begin{array}{rcl}
(\lambda x.M)\,N & \xrightarrow{\beta} & M[x:=N]\\
\mathtt{fst}(\langle M,N\rangle) & \xrightarrow{\beta} & M\\
\mathtt{snd}(\langle M,N\rangle) & \xrightarrow{\beta} & N\\
\mathtt{case}(\mathtt{inl}(P),\lambda x.M,\lambda y.N) & \xrightarrow{\beta} & M[x:=P]\\
\mathtt{case}(\mathtt{inr}(P),\lambda x.M,\lambda y.N) & \xrightarrow{\beta} & N[y:=P]
\end{array}$$

The first reduction rule deals with proof instances as shown in (3.1). Since all rules are rather standard, we do not explain them further.

Unfortunately, to ensure that the subformula property holds for normal NJ-proofs, the beta-reductions are not sufficient. For this property to hold, further reduction rules, referred to as *commuting reductions*, are required. These rules are often criticised as being inelegant, if not catastrophic [Girard et al., 1989, Page 74]. Certainly, they are more complicated than beta-reductions and seem to be necessary just because of certain defects of the syntax based on terms, but we shall show that they correspond to the (quite natural) commuting reductions of sequent calculi.

Definition 3.1.5 (Commuting Reductions)**:**

$$\begin{array}{rcl}
\mathtt{case}(P,\lambda x.M,\lambda y.N)\,Q & \xrightarrow{\gamma} & \mathtt{case}(P,\lambda x.(M\,Q),\lambda y.(N\,Q))\\
\mathtt{fst}(\mathtt{case}(P,\lambda x.M,\lambda y.N)) & \xrightarrow{\gamma} & \mathtt{case}(P,\lambda x.\mathtt{fst}(M),\lambda y.\mathtt{fst}(N))\\
\mathtt{snd}(\mathtt{case}(P,\lambda x.M,\lambda y.N)) & \xrightarrow{\gamma} & \mathtt{case}(P,\lambda x.\mathtt{snd}(M),\lambda y.\mathtt{snd}(N))\\
\mathtt{case}(\mathtt{case}(P,\lambda x.M,\lambda y.N),\lambda u.S,\lambda v.T) & \xrightarrow{\gamma} & \mathtt{case}(P,\lambda x.\mathtt{case}(M,\lambda u.S,\lambda v.T),\\
 & & \qquad\lambda y.\mathtt{case}(N,\lambda u.S,\lambda v.T))
\end{array}$$

We shall, as usual, assume that $\xrightarrow{\beta}$ and $\xrightarrow{\gamma}$ are closed under context formation. Thus we can define the normalisation procedure of NJ-proofs as follows.

Definition 3.1.6 (Normalisation Procedure of NJ-Proofs)**:**
The normalisation procedure for NJ-proofs, $(\Lambda, \xrightarrow{\iota})$, is a reduction system where:

- Λ is the set of simply-typed lambda terms, and
- $\xrightarrow{\iota}$ consists of beta-reductions and commuting reductions; that is

$$\xrightarrow{\iota} \stackrel{\text{def}}{=} \xrightarrow{\beta} \cup \xrightarrow{\gamma} \quad .$$

It is simple to show that $\xrightarrow{\iota}$ satisfies the subject reduction property.

Proposition 3.1.7 (Subject Reduction, Prawitz, 1965)**:** Suppose M is a simply-typed lambda term with the typing judgement $\Gamma \triangleright M : B$ and $M \xrightarrow{\iota} N$, then N is a simply-typed lambda term with the typing judgement $\Gamma \triangleright M : B$.

Proof. Analogous to proof of Proposition 2.2.10. □

Furthermore, the reduction $\xrightarrow{\iota}$ is complete in the sense that the normal NJ-proofs satisfy the subformula property. Whilst in the setting of sequent calculi the proof of this property is by trivial inspection of the inference rules, the proof for NJ is more difficult. For example, a simple inspection of the inference rules is not sufficient; for the details of the proof we refer to [Prawitz, 1965]. One interesting feature of the natural deduction calculus we shall introduce in Section 3.3 for classical logic is that the subformula property for normal proofs can be proved, like in sequent calculi, by a simple inspection of the inference rules.

3.2 The Correspondence between Cut-Elimination and Normalisation I

In this section, we address the question whether there is a correspondence between cut-elimination and normalisation, or more precisely whether the translations defined in the preceding section respect reduction. This "correspondence question" was tackled previously by Zucker [1974] and Pottinger [1977], who wrote the classic works on the subject. Zucker mentioned that the correspondence question goes back to the following remark by Kreisel [1971, Page 113], which both Zucker and Pottinger disprove.

> Consider…a calculus of sequents and a system of natural deduction. Inspection shows that…the normalization procedure for systems of natural deduction does not correspond to a particularly natural cut elimination procedure.

However, there are some subtle points in their works, which make their answers not so clear-cut. Let us analyse in turn these subtleties.

Zucker studied the correspondence question between NJ and a variant of LJ. Since all reduction sequences of NJ are terminating, one proviso for the correspondence is that all reduction sequences in the sequent calculus are terminating, too. Indeed, a non-terminating reduction sequence in the sequent calculus would hamper the correspondence with NJ. Another proviso is that the cut-elimination procedure has to include reduction rules which allow cuts to pass over other cuts. Gentzen [1935] did not consider such rules, as they are usually the source for loops and therefore for non-terminating reduction sequences; however, they are crucial for the correspondence question.

One (rather *ad hoc*) way round the problem of loops is to consider only *proper reduction sequences* [Zucker, 1974, Page 93], which do not include repetitions. This restriction is sufficient to ensure termination of Zucker's cut-elimination procedure in the $(\wedge, \supset, \forall)$-fragment of intuitionistic logic, and so he could answer the correspondence question positively for this fragment. Using the $\vee_L$-rule, however, he constructed a non-repeating, non-terminating reduction sequence, and thus he failed to ensure the first proviso. A simplified version of this reduction sequence is shown in Figure 3.5. Consequently, he gave a negative answer to the correspondence question for the fragment that includes $\vee$ and $\exists$. Zucker concluded that the lack of termination of cut-elimination was responsible for this failure, but we shall show that even if one does have a strongly normalising cut-elimination procedure (which is very similar to the one used by Zucker), the correspondence between cut-elimination and normalisation fails for the standard formulation of NJ: certain commuting reductions of the sequent calculus cannot be encoded in NJ. In Section 3.3 we shall make appropriate changes to the standard formulation of natural deduction and improve Zucker's result by showing the correspondence in classical logic for all connectives.

In some sense Pottinger solved Zucker's problem with non-terminating reduction sequences, and thus he could show the correspondence for all connectives. Unfortunately, his method had a subtle defect: he annotated sequent proofs with lambda-terms and then formalised cut-elimination as normalisation. As already noted by himself, the notion of normality for lambda terms does *not* coincide with the notion of cut-freedom [Pottinger, 1977, Page 323]. Consider for example the following sequent proof.

$$\dfrac{\dfrac{\overline{B \vdash B} \quad \overline{B \vdash B}}{B \vdash B}\,\text{Cut} \qquad \dfrac{\overline{B, C \vdash C}}{B \vdash C \supset C}\,\supset_R}{B \vdash C \supset C}\,\text{Cut}$$

Using Pottinger's term assignment, this proof is annotated with the lambda term $\lambda x.x$. Clearly, this term is normal, but the sequent proof contains two cuts. Because the cut-elimination reductions that start from the proof above are not "mirrored" by any reduction of the lambda term, the question whether these cut-reductions break the correspondence remains open. We shall avoid this problem by using our term assignment, which encodes precisely the structure of sequent proofs.

Now we shall begin by showing that normalisation in NJ can be simulated by our intuitionistic cut-elimination procedure. To do so, we shall show that beta-reductions map onto a series of cut-reductions and that commuting reductions map onto identities. For the first part we have to prove three lemmas. The first shows that a form of the substitution lemma holds for the proof substitution $[\_]$ (the "usual" substitution lemma does, in general, not hold for this substitution; cf. Lemma 2.3.1).

$$\dfrac{\dfrac{\overset{\pi_1}{C\vdash F}\quad \overset{\pi_2}{D\vdash F}}{C\lor D\vdash F}\lor_L \qquad \dfrac{\dfrac{\overset{\pi_3}{A\vdash F}\quad\overset{\pi_4}{B\vdash F}}{A\lor B\vdash F}\lor_L \quad \overset{\pi_5}{F,F\vdash G}}{A\lor B,F\vdash G}\,\mathbf{Cut}^\star}{C\lor D, A\lor B\vdash G}\,\mathbf{Cut}$$

$$\downarrow$$

$$\dfrac{\dfrac{\overset{\pi_1}{C\vdash F}\quad \overset{\pi_2}{D\vdash F}}{C\lor D\vdash F}\lor_L \qquad \dfrac{\dfrac{\dfrac{\overset{\pi_3}{A\vdash F}\quad \overset{\pi_5}{F,F\vdash G}}{A,F\vdash G}\,\text{Cut}\quad \dfrac{\overset{\pi_4}{B\vdash F}\quad \overset{\pi_5}{F,F\vdash G}}{B,F\vdash G}\,\text{Cut}}{A\lor B,F,F\vdash G}\lor_L}{A\lor B,F\vdash G}\,\text{Contr}_L}{C\lor D,A\lor B\vdash G}\,\text{Cut}^\bullet$$

$$\downarrow$$

$$\dfrac{\dfrac{\dfrac{\overset{\pi_1}{C\vdash F}\quad \overset{\pi_2}{D\vdash F}}{C\lor D\vdash F}\lor_L \qquad \dfrac{\dfrac{\overset{\pi_1}{C\vdash F}\quad \overset{\pi_2}{D\vdash F}}{C\lor D\vdash F}\lor_L\quad \dfrac{\dfrac{\overset{\pi_3}{A\vdash F}\quad \overset{\pi_5}{F,F\vdash G}}{A,F\vdash G}\,\text{Cut}\quad \dfrac{\overset{\pi_4}{B\vdash F}\quad \overset{\pi_5}{F,F\vdash G}}{B,F\vdash G}\,\text{Cut}}{A\lor B,F,F\vdash G}\lor_L}{C\lor D,A\lor B,F\vdash G}\,\mathbf{Cut}}{C\lor D,C\lor D,A\lor B\vdash G}\,\mathbf{Cut}}{C\lor D,A\lor B\vdash G}\,\text{Contr}_L$$

Figure 3.5: A non-terminating reduction sequence in Zucker's sequent calculus (see [Ungar, 1992]). In the first step, the cut marked with a star is reduced creating two copies of π_5 as well as introducing a contraction. In the second step, the cut marked with a disk is permuted with the contraction rule creating two copies of π_1 and π_2. Now the cuts written in bold face have the same shape, and we can repeat the reduction applied in the first step and so forth *ad infinitum.*

Lemma 3.2.1: For all $M \in \mathcal{J}$ and two substitutions having the form $[a := (y)S]$ and $[x := \langle b \rangle T]$, we have

$$M[a := (y)S][x := \langle b \rangle T] \xrightarrow{int}{}^{*} M[x := \langle b \rangle T][a := (y)S[x := \langle b \rangle T]]$$

provided a is not free in $\langle b \rangle T$, and S freshly introduces y, but is not an axiom.

Proof. By induction on the structure of M (see Page 156). ☐

The next lemma concerns the freshness of a co-name, say c, when forming the translation $|\_|^{\mathsf{s}}_{c}$; it will be applied to determine whether a logical cut-reduction can be performed.

Lemma 3.2.2: For all $M \in \Lambda$ and for all co-names c, if M is of the form

$$x \mid \lambda x.S \mid \langle S, T \rangle \mid \mathtt{inl}(S) \mid \mathtt{inr}(S)$$

then $|M|^{\mathsf{s}}_{c}$ freshly introduces c.

Proof. By inspection of the inference rules shown in Figure 3.2 and the translations given in Figure 3.3. ☐

The variable substitution and proof substitution are related as follows.

Lemma 3.2.3: For all $M, N \in \Lambda$ and for all co-names a, b, we have

$$|M|^{\mathsf{s}}_{a}[x := \langle b \rangle |N|^{\mathsf{s}}_{b}] \xrightarrow{int}{}^{*} |M[x := N]|^{\mathsf{s}}_{a} \; .$$

Proof. By induction on the structure of M (see Page 157). ☐

Now we are in a position to show that beta-reductions map onto a series of cut-reductions. To make the induction go through, we have to prove this in conjunction with a slightly more general statement.

Proposition 3.2.4: For all $M, M' \in \Lambda$, for all co-names c, and for any substitution of the form $[c := (y)N]$ where N freshly introduces y and is not an axiom, we have if $M \xrightarrow{\beta} M'$ then

(i) $|M|^{\mathsf{s}}_{c} \xrightarrow{int}{}^{+} |M'|^{\mathsf{s}}_{c}$

(ii) $|M|^{\mathsf{s}}_{c}[c := (y)N] \xrightarrow{int}{}^{+} |M'|^{\mathsf{s}}_{c}[c := (y)N]$

Proof. By induction on the definition of $\xrightarrow{\beta}$ (see Page 157). ☐

It remains to check whether commuting reductions translate onto identities.

Proposition 3.2.5: For all $M, N \in \Lambda$ and for all c's, if $M \xrightarrow{\gamma} N$, then $|M|^{\mathsf{s}}_{c} \equiv |N|^{\mathsf{s}}_{c}$.

Proof. By induction on the definition of $\xrightarrow{\gamma}$ (see Page 158). ☐

So we have shown that normalisation can be simulated by cut-elimination using the translation $|\_|^{\mathsf{s}}$. To sum up this result, we should like to show that $(\Lambda, \xrightarrow{\iota})$ is strongly normalising by appealing to the strong normalisation result of $(\mathcal{J}, \xrightarrow{int})$. This is a very roundabout way for proving this property for $(\Lambda, \xrightarrow{\iota})$, but the point here is to support the claim made in Section 2.5 stating that our reduction system for cut-elimination is very useful for strong normalisation proofs by translation.

Theorem 3.2.6 (Prawitz, 1971)**:** The reduction system $(\Lambda, \xrightarrow{\iota})$ is strongly normalising.

Proof. Every $\xrightarrow{\iota}$-reduction sequence is of the form

$$\begin{tikzcd}
M_1 \arrow[r, "\gamma"] \arrow[d, "\mathsf{s}"'] & {}^{*}\, M_2 \arrow[r, "\beta"] \arrow[d, "\mathsf{s}"'] & M_3 \arrow[r, "\gamma"] \arrow[d, "\mathsf{s}"'] & {}^{*}\, M_4 \arrow[d, "\mathsf{s}"'] & \ldots \\
N_1 \arrow[r, equal] & N_2 \arrow[r, "int"] & {}^{+}\, N_3 \arrow[r, equal] & N_4 & \ldots
\end{tikzcd}$$

which as shown can be mapped onto a reduction sequence in our intuitionistic sequent calculus (first and third square are by Proposition 3.2.5 and second square by Proposition 3.2.4). Given the measure shown on Page 159, we can easily infer that all $\xrightarrow{\gamma}$-reduction sequences must terminate. Hence by Lemma 2.6.3 and Theorem 2.5.2, we find that every $\xrightarrow{\iota}$-reduction sequence is terminating. Thus we are done. □

Next we shall show the correspondence in the other direction, meaning that cut-elimination can be simulated by normalisation. As we shall see, the difficulties here arise from the commuting reductions in our sequent calculus, and therefore we shall first focus on the $(\wedge, \supset)$-fragment of intuitionistic logic where the problems can be avoided. We shall write $\mathfrak{I}^{(\wedge,\supset)}$ for the corresponding set of well-typed terms of our sequent calculus. Before we can prove the proposition for the correspondence, we have to show that the proof substitution of our sequent calculus corresponds to the variable substitution in NJ.

Lemma 3.2.7: For all $M, N \in \mathfrak{I}^{(\wedge,\supset)}$ we have

(i) $|M[x := \langle a\rangle N]|^{\mathsf{n}} \equiv |M|^{\mathsf{n}}[x := |N|^{\mathsf{n}}]$

(ii) $|M[a := (x)N]|^{\mathsf{n}} \equiv |N|^{\mathsf{n}}[x := |M|^{\mathsf{n}}]$, provided $a \in FC(M)$.

Proof. By induction on the structure of M (see Page 159). □

The proposition which shows that cut-elimination can be simulated by normalisation is as follows.

Proposition 3.2.8: For all $M, N \in \mathfrak{I}^{(\wedge,\supset)}$, if $M \xrightarrow{int} N$, then $|M|^{\mathsf{n}} \xrightarrow{\iota}{}^{*} |N|^{\mathsf{n}}$.

Proof. By induction on the definition of $\xrightarrow{int}$ (see Page 160). □

Let us now take a step back and analyse the result obtained in Proposition 3.2.8. What is shown is that cut-elimination steps can be simulated by beta-reductions—but simulated in a weak sense. Whenever

$$\mathsf{Cut}(\langle a\rangle M,(x)N) \xrightarrow{int} \mathsf{Cut}(\langle a\rangle M',(x)N)$$

with x is not free in N, then

$$|\mathsf{Cut}(\langle a\rangle M,(x)N)|^{\mathsf{n}} \equiv |\mathsf{Cut}(\langle a\rangle M',(x)N)|^{\mathsf{n}}\ .$$

We achieve a somewhat closer correspondence between cut-elimination and normalisation, if we translate the sequent calculus into a simply-typed lambda calculus extended with an explicit substitution operator. This was proposed, for example, by Herbelin [1994], and Barendregt and Ghilezan [2000]. On one hand, the explicit substitution operator ensures that more cut-elimination reductions match with normalisation reductions; on the other, we are not aware of any work that points out that there is a discrepancy preventing a perfect match. We shall demonstrate the discrepancy using $\lambda\mathtt{x}$ [Rose, 1996].

Remark 3.2.9: First let us remind the reader of the explicit substitution calculus λx briefly described on Page 48. The terms are given by the grammar

$$x \ \mid\ \lambda x.M \ \mid\ M\,N \ \mid\ M\langle x := N\rangle\ .$$

In contrast to the simply-typed lambda calculus, in $\lambda\mathtt{x}$ beta-reductions are split into more atomic steps: one step is the reduction $(\lambda x.M)N \xrightarrow{b} M\langle x := N\rangle$ introducing the explicit substitution operator, and the others involve reductions that permute this operator inside the term M. As a consequence, the translation from sequent proofs to natural deduction proofs can be modified, so as to incorporate the new term constructor.

$$\begin{array}{rcl}
|\mathsf{Cut}(\langle a\rangle M,(x)N)|^{\mathsf{n}'} & \stackrel{\mathrm{def}}{=} & |N|^{\mathsf{n}'}\langle x := |M|^{\mathsf{n}'}\rangle \\
|\mathsf{Imp}_R((x)\langle a\rangle M,c)|^{\mathsf{n}'} & \stackrel{\mathrm{def}}{=} & \lambda x.|M|^{\mathsf{n}'} \\
|\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)|^{\mathsf{n}'} & \stackrel{\mathrm{def}}{=} & |N|^{\mathsf{n}'}\langle x := (y\ |M|^{\mathsf{n}'})\rangle
\end{array}$$

To show the discrepancy, we shall now consider the logical cut-reduction for cuts with a cut-formula having an implication as top-most connective. On Page 29 we defined this cut-reduction as

$$\begin{array}{l}
\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M,b),(z)\mathsf{Imp}_L(\langle c\rangle N,(y)P,z)) \\
\quad \xrightarrow{l} \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle N,(x)M),(y)P) \ \text{ or} \\
\quad \xrightarrow{l} \mathsf{Cut}(\langle c\rangle N,(x)\mathsf{Cut}(\langle a\rangle M,(y)P))
\end{array} \tag{3.2}$$

including a choice of how the cuts are arranged in the reduct. There is no reason to prefer one choice over the other (they even lead to different normal forms as we shall show in Chapter 4). Unfortunately, using the translation $|\_|^{\mathsf{n}'}$ given above only the first reduction is matched by reductions in $\lambda\mathtt{x}$; the other is *not*. The translations of the reducts shown in (3.2) are:

$$|\mathsf{Cut}(\langle a\rangle\mathsf{Cut}(\langle c\rangle N,(x)M),(y)P)|^{\mathsf{n}'} = |P|^{\mathsf{n}'}\langle y := |M|^{\mathsf{n}'}\langle x := |N|^{\mathsf{n}'}\rangle\rangle$$
$$|\mathsf{Cut}(\langle c\rangle N,(x)\mathsf{Cut}(\langle a\rangle M,(y)P))|^{\mathsf{n}'} = |P|^{\mathsf{n}'}\langle y := |M|^{\mathsf{n}'}\rangle\langle x := |N|^{\mathsf{n}'}\rangle$$

With the (standard) reduction rules of $\lambda\mathtt{x}$, only the first reduct can be reached.

$$\begin{array}{rl} & |\mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle M,b),(z)\mathsf{Imp}_L(\langle c\rangle N,(y)P,z))|^{\mathsf{n}'} \\ = & |P|^{\mathsf{n}'}\langle y := z\,|N|^{\mathsf{n}'}\rangle\langle z := \lambda x.|M|^{\mathsf{n}'}\rangle \\ \longrightarrow & |P|^{\mathsf{n}'}\langle y := z\,|N|^{\mathsf{n}'}\langle z := \lambda x.|M|^{\mathsf{n}'}\rangle\rangle \\ \longrightarrow^* & |P|^{\mathsf{n}'}\langle y := (\lambda x.|M|^{\mathsf{n}'})\,|N|^{\mathsf{n}'}\rangle \\ \longrightarrow & |P|^{\mathsf{n}'}\langle y := |M|^{\mathsf{n}'}\langle x := |N|^{\mathsf{n}'}\rangle\rangle \end{array}$$

The discrepancy appears if we try to reach the second reduct. Here we would need a reduction of the form ($x \notin FV(P)$)

$$P\langle y := M\langle x := N\rangle\rangle \longrightarrow P\langle y := M\rangle\langle x := N\rangle$$

which, as far as we know, has not been considered for any explicit substitution calculus and indeed would be rather strange.

What is shown by this example is that the obvious candidate for a simply-typed lambda calculus with an explicit substitution operator and a natural translation from sequent proofs to this lambda calculus gives a closer correspondence between cut-elimination and normalisation, but not all cut-reductions are "mirrored". In the next section we shall develop a natural deduction calculus with a symmetric explicit substitution operator. For this calculus we shall give translations so that *all* cut-reductions map onto normalisation reductions.

In the rest of this section we shall analyse whether we can extend Proposition 3.2.8 to all connectives of intuitionistic logic. As mentioned earlier, Zucker concluded that this cannot be done for the fragment including $\vee$ or $\exists$, because (in his setting) cut-elimination fails to be strongly normalising. Our cut-elimination procedure $(\mathcal{I}, \xrightarrow{int})$ *is* strongly normalising, and thus it is interesting to re-address the question of the correspondence. Unfortunately, it turns out that the $\vee$-connective is still problematic with respect to the simulation of cut-elimination.

Clearly, to eliminate all cut-instances from sequent proofs we need the commuting reduction of the form

$$\mathsf{Cut}(\langle a\rangle M,(x)N) \xrightarrow{c} M[a := (x)N] \tag{3.3}$$

which may be applied if M does not freshly introduce a. In the $(\wedge,\supset)$-fragment considered above, this reduction is mapped by $|\_|^{\mathsf{n}}$ onto an identity (see Lemma 3.2.7). But in case M is an $\vee_L$-rule, then $|\_|^{\mathsf{n}}$ does not translate (3.3) onto an identity. In consequence, the correspondence fails. The issues are best explained with an example. Consider the following sequent proof (where for brevity we have omitted all labels).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_R\quad\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_R}{A\vee B,B,A\vdash A\wedge B}\vee_L\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\wedge_{L_1}}{A\vee B,B,A\vdash A\wedge A}\text{Cut} \tag{3.4}$$

To eliminate the cut, we need to apply the rule shown in (3.3). This gives us

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_R\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\wedge_{L_1}}{A,B\vdash A\wedge A}\text{Cut}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_R\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\wedge_{L_1}}{A,B\vdash A\wedge A}\text{Cut}}{A\vee B,B,A\vdash A\wedge B}\vee_L \tag{3.5}$$

where two copies of the cut have been created. Let us give the corresponding natural deduction proofs. Forming $|(3.4)|^{\mathsf{n}}$ we have the natural deduction proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I\quad\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A\vee B,A,B\vdash A\wedge B}\vee_E}{A\vee B,A,B\vdash A}\wedge_{E_1}\quad\dfrac{\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I\quad\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A\vee B,A,B\vdash A\wedge B}\vee_E}{A\vee B,A,B\vdash A}\wedge_{E_1}}{A\vee B,A,B\vdash A\wedge A}\wedge_I$$

in which the $\wedge_{E_1}$-rules can be permuted with the $\vee_E$-rules. This gives the following proof.

$$\dfrac{\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A\vee B,A,B\vdash A}\vee_E\quad\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A\vee B,A,B\vdash A}\vee_E}{A\vee B,A,B\vdash A\wedge A}\wedge_I$$

Here we are stuck! There is no reduction that would give $|(3.5)|^{\mathsf{n}}$—the proof below.

$$\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A,B\vdash A\wedge A}\wedge_I\quad\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A,B\vdash A\wedge A}\wedge_I}{A\vee B,B,A\vdash A\wedge A}\vee_E$$

What is needed is a reduction rule which permutes $\wedge_I$ with $\vee_E$. On the level of lambda terms the corresponding reduction rule is as follows.

$$\begin{aligned}&\langle\mathtt{case}(P,\lambda x.M,\lambda y.N),\mathtt{case}(P,\lambda z.S,\lambda w.T)\rangle\\ &\qquad\longrightarrow\quad\mathtt{case}(P,\lambda x.\langle M,S[z:=x]\rangle,\lambda y.\langle N,T[w:=y]\rangle)\end{aligned}$$

Since the reduction (3.3) cannot be dispensed with in the sequent calculus, we have to extend the natural deduction calculus with commuting reductions of the kind shown above.[2] As can be easily shown, these commuting reductions are harmless with respect to the subformula property. We shall do the extension in the next section and establish the correspondence for classical logic.

3.3 Classical Natural Deduction

Gentzen [1935] defined the classical natural deduction calculus NK by extending NJ with an additional axiom, known as the excluded middle axiom, shown below.

$$B \vee \neg B$$

In order to obtain a classical natural deduction calculus, one can, alternatively, add to NJ the stability axiom, $\neg\neg B \supset B$, an instance of Peirce's law, $((B \supset \bot) \supset B) \supset B$, or an additional inference rule called the absurdity rule. All these calculi are adequate for classical logic in terms of provability; however, from a proof theoretic point of view they are not very elegant. Let us first mention two shortcomings remarked by Gentzen [1935, Pages 189–190] and Prawitz [1971, Page 251]: the inference rules for negation do not fall into the general pattern of introduction and elimination rules, and there are problems concerning the subformula property. For analysing the correspondence question a third shortcoming is particularly annoying: classical sequent proofs can only be encoded into NK via a double negation translation.

Better suited for handling classical logic and much more convenient for showing the correspondence is the sequence-conclusion natural deduction calculus[3] introduced by Boričić [1985]. We shall reformulate this calculus into "sequent-style". For this, we shall take sequents of the form $\Gamma \vdash \Delta$ where Γ and Δ are sets of (label,formula) pairs, analogous to the left- and right-contexts introduced for sequent calculi. Thus the labels in Γ are referred to as names and in Δ as co-names. The inference rules of Boričić's calculus, as usual for natural deduction, come in pairs introducing a connective and eliminating a connective. For instance the rules introducing and eliminating $\wedge$ are:

$$\frac{\Gamma \vdash \Delta, a : B \quad \Gamma \vdash \Delta, b : C}{\Gamma \vdash \Delta, c : B \wedge C} \wedge_I \qquad\qquad \frac{\Gamma \vdash \Delta, a : B_1 \wedge B_2}{\Gamma \vdash \Delta, b : B_i} \wedge_{E_i}$$

Again, redexes arise when a connective is introduced and eliminated afterwards. An example is

$$\cfrac{\cfrac{\overset{\pi}{\Gamma \vdash \Delta, a : B} \quad \overset{\pi'}{\Gamma \vdash \Delta, b : C}}{\Gamma \vdash \Delta, c : B \wedge C} \wedge_I}{\Gamma \vdash \Delta, d : B} \wedge_{E_1}$$

which reduces to the proof

$$\begin{array}{c} \vdots \\ \Gamma \vdash \Delta, d : B \end{array}$$

[2] While this rule has not been considered for a natural deduction calculus, unpublished work by Dybjer and Ong suggests that this rule comes into play when considering bicartesian closed categories.

[3] To be consistent with our terminology for sequents, we should refer to this natural deduction calculus as sequence-succedent natural deduction calculus, but we shall follow Boričić's original terminology.

obtained from π by renaming a to d. However, notice that in case of more than one formula in the succedent there might be several inferences in "between" the rule that introduces a connective and the rule that eliminates it, as indicated below.

$$\dfrac{\dfrac{\Gamma' \vdash \Delta', a:B \quad \Gamma' \vdash \Delta', b:C}{\Gamma' \vdash \Delta', c:B\wedge C}\wedge_I}{\begin{array}{c}\vdots\\ \dfrac{\Gamma \vdash \Delta, c:B\wedge C}{\Gamma \vdash \Delta, d:B}\wedge_{E_1}\end{array}}$$

In order to apply the reduction rule eliminating the redex, we have to move the elimination rule next to the introduction rule. This means already in the fragment containing only the $\wedge$-connective we need commuting reductions. Cellucci [1992] gave the first set of reduction rules for Boričić's calculus. These reductions are Gentzen-like, in the sense of rewriting neighbouring inference rules only. As with sequent calculi, a set of Gentzen-like reduction rules that is strongly normalising is rather difficult to find (we illustrated some of the difficulties with an LK-proof in Example 2.1.3). Unlike his rules, which are *not* strongly normalising, our reduction rules will be similar to the "global" reduction rules introduced for $(\mathcal{T}, \xrightarrow{cut})$ and in consequence will be strongly normalising. Restricted versions of our rules were studied previously in connection with $\lambda\mu$ [Parigot, 1992]. For space reasons we shall give the calculus for only the propositional fragment of classical logic. Note however that all results given below can be easily extended to more expressive fragments.

We alluded in Section 2.5 to the many variations of defining sequent calculi; similarly there are many variants of sequence-conclusion natural deduction calculi. To show the correspondence we shall use the calculus given in Figure 3.6. The notion of implicit contractions and the context convention can be adapted to this calculus in the obvious way (see Page 19). Whilst our introduction rules are rather standard, our elimination rules are somewhat non-standard. They are implicit in the work by Boričić [1985], and some of them appear in *free deduction* introduced by Parigot [1991]. However, we should like to stress the point that the reasons for choosing this particular set of inference rules are entirely pragmatic: they simplify considerably the translations between natural deduction proofs and sequent proofs. Other sets of inference rules can also be used to study the correspondence, although the machinery required is more complicated.

Recall the translation $|\_|^{\mathsf{n}}$ from intuitionistic sequent proofs to intuitionistic natural deduction proofs (Figure 3.4). There, as illustrated below, we applied a substitution operation to translate cuts.

$$|\mathsf{Cut}(\langle a\rangle M, (x)N)|^{\mathsf{n}} \;=\; |N|^{\mathsf{n}}[x := |M|^{\mathsf{n}}]$$

In consequence, we obtained a rather weak correspondence: whenever x is not free in N and therefore in $|N|^{\mathsf{n}}$, the cut-reductions that M can perform are mapped onto identities. Consider an analogous translation for classical sequent proofs (denoted by $|\_|^{\mathsf{N}}$), and suppose we are given the term corresponding to Lafont's example; that is

$$\mathsf{Cut}(\langle a\rangle M, (x)N) \tag{3.6}$$

where a is not free in M and x not free in N. Clearly, this term can reduce to either M

$$\overline{x:B,\Gamma\vdash\Delta,a:B}$$

$$\frac{x:B,\Gamma\vdash\Delta,a:C}{\Gamma\vdash\Delta,b:B\supset C}\supset_I \qquad \frac{\Gamma\vdash\Delta,a:B\supset C \quad \Gamma\vdash\Delta,b:B \quad x:C,\Gamma\vdash\Delta}{\Gamma\vdash\Delta}\supset_E$$

$$\frac{\Gamma\vdash\Delta,a:B \qquad \Gamma\vdash\Delta,b:C}{\Gamma\vdash\Delta,c:B\wedge C}\wedge_I \qquad \frac{\Gamma\vdash\Delta,a:B_1\wedge B_2 \quad x:B_i,\Gamma\vdash\Delta}{\Gamma\vdash\Delta}\wedge_{E_i}$$

$$\frac{\Gamma\vdash\Delta,a:B_i}{\Gamma\vdash\Delta,b:B_1\vee B_2}\vee_{I_i} \qquad \frac{\Gamma\vdash\Delta,a:B\vee C \quad x:B,\Gamma\vdash\Delta \quad y:C,\Gamma\vdash\Delta}{\Gamma\vdash\Delta}\vee_E$$

$$\frac{x:B,\Gamma\vdash\Delta}{\Gamma\vdash\Delta,a:\neg B}\neg_I \qquad \frac{\Gamma\vdash\Delta,a:\neg B \quad \Gamma\vdash\Delta,b:B}{\Gamma\vdash\Delta}\neg_E$$

Figure 3.6: Natural deduction calculus for classical logic.

or N. Translating this cut-instance to a natural deduction proof using any kind of substitution operation would be disastrous: it would result into a failure of the correspondence between normalisation and cut-elimination. Given $|\_|^{\mathsf{N}}$ we would translate this term as follows

$$|\mathsf{Cut}(\langle a\rangle M,(x)N)|^{\mathsf{N}} \;=\; |N|^{\mathsf{N}}[x:=\langle a\rangle|M|^{\mathsf{N}}]$$

where, because x not free in $|N|^{\mathsf{N}}$, the substitution vanishes; that is the term on the right-hand side is equivalent to $|N|^{\mathsf{N}}$. In consequence, we would "lose" the cut-reductions which reduce (3.6) to M. This means we cannot define $|\_|^{\mathsf{N}}$ using a substitution operation. We shall remedy this problem by introducing the following (symmetric) explicit substitution rule

$$\frac{\Gamma_1\vdash\Delta_1,a:B \quad x:B,\Gamma_2\vdash\Delta_2}{\Gamma_1,\Gamma_2\vdash\Delta_1,\Delta_2}\ \text{Subst}$$

which allows us to define a translation $|\_|^{\mathsf{N}}$ and reduction rules so that $|(3.6)|^{\mathsf{N}}$ can match *both* reductions of (3.6). Thus, we add this inference rule to our natural deduction calculus.

In the following we shall develop reduction rules for our classical natural deduction calculus. As with our cut-elimination rules, these reduction rules will be defined using terms annotated to proofs. The raw terms are defined below.

Definition 3.3.1 (Raw Terms)**:** The set of raw terms, R, is defined by the grammar

$$
\begin{array}{rcll}
M, N, P & ::= & \mathsf{Id}(x, a) & \text{identity} \\
& | & \mathsf{And}_I(\langle a{:}B\rangle M, \langle b{:}C\rangle N, c) & \text{and-introduction} \\
& | & \mathsf{And}^i_E(\langle a{:}B_1{\wedge}B_2\rangle M, (x:B_i)N) & \text{and}^i\text{-elimination} \quad i = 1,2 \\
& | & \mathsf{Or}^i_I(\langle a_i{:}B_i\rangle M, b) & \text{or}^i\text{-introduction} \quad i = 1,2 \\
& | & \mathsf{Or}_E(\langle a{:}B{\vee}C\rangle M, (x{:}B)N, (y{:}C)P) & \text{or-elimination} \\
& | & \mathsf{Imp}_I((x{:}B)\langle a{:}C\rangle M, b) & \text{implication-introduction} \\
& | & \mathsf{Imp}_E(\langle a{:}B{\supset}C\rangle M, \langle b{:}C\rangle N, (x{:}B)P) & \text{implication-elimination} \\
& | & \mathsf{Not}_I((x{:}B)M, b) & \text{negation-introduction} \\
& | & \mathsf{Not}_E(\langle a{:}\neg B\rangle M, \langle b{:}B\rangle N) & \text{negation-elimination} \\
& | & \mathsf{Subst}(\langle a{:}B\rangle M, (x{:}B)N) & \text{symmetric substitution}
\end{array}
$$

Henceforth we shall omit the types on the binders, provided they are clear from the context. Convention 2.2.4 concerning the Barendregt-style naming convention extends to the terms defined above. Given the notions of free names and co-names for terms of $\mathcal{R}$, it is routine to define analogous notions for the terms of R.

To annotate terms to our natural deduction proofs we shall replace every sequent of the form $\Gamma \vdash \Delta$ with a typing judgement of the form $\Gamma \triangleright M \triangleright \Delta$ in which Γ and Δ are contexts and M is a term. As usual, we are interested in only well-typed terms, i.e., those for which there is a context Γ and a context Δ such that $\Gamma \triangleright M \triangleright \Delta$ is derivable given the rules shown in Figure 3.7. The set of well-typed terms is defined as follows.

$$\mathrm{K} \stackrel{\text{def}}{=} \left\{ M \;\middle|\; M \in \mathrm{R} \text{ and well-typed by the inference rules given in Figure 3.7} \right\}$$

Our inference rules are defined such that only axioms and introduction rules are concerned with introducing a co-name; elimination rules, on the other hand, are concerned with only eliminating a co-name. This is expressed in the following definition.

Definition 3.3.2:

- A term M *introduces* the co-name c, if and only if M is of the form:

$$\mathsf{Id}(x, c) \;\Big|\; \mathsf{And}_I(\langle a\rangle S, \langle b\rangle T, c) \;\Big|\; \mathsf{Or}^i_I(\langle a\rangle S, c) \;\Big|\; \mathsf{Imp}_I((x)\langle a\rangle S, c) \;\Big|\; \mathsf{Not}_I((x)S, c)$$

- A term M *freshly* introduces a co-name, if and only if none of its proper subterms introduces this co-name.

- A term M *eliminates* the co-name c, if and only if M is of the form:

$$\mathsf{And}^i_E(\langle c\rangle S, (x)T) \;\Big|\; \mathsf{Or}_E(\langle c\rangle S, (x)T, (y)P) \;\Big|\; \mathsf{Imp}_E(\langle c\rangle S, \langle b\rangle T, (x)P) \;\Big|\; \mathsf{Not}_E(\langle c\rangle S, \langle b\rangle T)$$

Next we shall give the translation $|\_|^{\mathsf{N}}$, which maps terms of $\mathcal{T}$ to terms of K, and the translation $|\_|^{\mathsf{S}}$ for the other way around, that is from terms of K to terms of $\mathcal{T}$. They are given in Figures 3.8 and 3.9, respectively. An important property of these translations is that they respect typing judgements. The proof of this property coincides with showing the soundness and completeness of our natural deduction calculus with respect to provability in classical logic.

$$\frac{}{x:B,\Gamma \triangleright \mathsf{Id}(x,a) \triangleright \Delta, a:B}$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B \qquad \Gamma \triangleright N \triangleright \Delta, b:C}{\Gamma \triangleright \mathsf{And}_I(\langle a\rangle M, \langle b\rangle N, c) \triangleright \Delta, c:B\wedge C}\ \wedge_I \qquad \frac{\Gamma \triangleright M \triangleright \Delta, a:B_i}{\Gamma \triangleright \mathsf{Or}^i_I(\langle a\rangle M, b) \triangleright \Delta, b:B_1\vee B_2}\ \vee_{I_i}$$

$$\frac{x:B,\Gamma \triangleright M \triangleright \Delta, a:C}{\Gamma \triangleright \mathsf{Imp}_I((x)\langle a\rangle M, b) \triangleright \Delta, b:B\supset C}\ \supset_I \qquad \frac{x:B,\Gamma \triangleright M \triangleright \Delta}{\Gamma \triangleright \mathsf{Not}_I((x)M, a) \triangleright \Delta, a:\neg B}\ \neg_I$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B_1\wedge B_2 \qquad x:B_i,\Gamma \triangleright N \triangleright \Delta}{\Gamma \triangleright \mathsf{And}^i_E(\langle a\rangle M, (x)N) \triangleright \Delta}\ \wedge_{E_i}$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B\vee C \qquad x:B,\Gamma \triangleright N \triangleright \Delta \qquad y:C,\Gamma \triangleright P \triangleright \Delta}{\Gamma \triangleright \mathsf{Or}_E(\langle a\rangle M, (x)N, (y)P) \triangleright \Delta}\ \vee_E$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B\supset C \qquad \Gamma \triangleright N \triangleright \Delta, b:B \qquad x:C,\Gamma \triangleright P \triangleright \Delta}{\Gamma \triangleright \mathsf{Imp}_E(\langle a\rangle M, \langle b\rangle N, (x)P) \triangleright \Delta}\ \supset_E$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:\neg B \qquad \Gamma \triangleright N \triangleright \Delta, b:B}{\Gamma \triangleright \mathsf{Not}_E(\langle a\rangle M, \langle b\rangle N) \triangleright \Delta}\ \neg_E$$

$$\frac{\Gamma_1 \triangleright M \triangleright \Delta_1, a:B \qquad x:B,\Gamma_2 \triangleright N \triangleright \Delta_2}{\Gamma_1,\Gamma_2 \triangleright \mathsf{Subst}(\langle a\rangle M, (x)N) \triangleright \Delta_1,\Delta_2}\ \text{Subst}$$

Figure 3.7: Term assignment for classical natural deduction proofs.

$$
\begin{array}{lrcl}
(1) & |\mathsf{Id}(x,a)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Ax}(x,a)\\
(2) & |\mathsf{Subst}(\langle a\rangle M,(x)N)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle\,|M|^{\mathsf{S}},(x)\,|N|^{\mathsf{S}})\\[4pt]
(3) & |\mathsf{And}_I(\langle a\rangle M,\langle b\rangle N,c)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{And}_R(\langle a\rangle\,|M|^{\mathsf{S}},\langle b\rangle\,|N|^{\mathsf{S}},c)\\
(4) & |\mathsf{Or}^i_I(\langle a\rangle M,b)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Or}^i_R(\langle a\rangle\,|M|^{\mathsf{S}},b)\\
(5) & |\mathsf{Imp}_I((x)\langle a\rangle M,b)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Imp}_R((x)\langle a\rangle\,|M|^{\mathsf{S}},b)\\
(6) & |\mathsf{Not}_I((x)M,a)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Not}_R((x)\,|M|^{\mathsf{S}},a)\\[4pt]
(7) & |\mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(z,a),(x)N)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{And}^i_L((x)\,|N|^{\mathsf{S}},z)\\
(8) & |\mathsf{Or}_E(\langle a\rangle\mathsf{Id}(z,a),(x)N,(y)P)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Or}_L((x)\,|N|^{\mathsf{S}},(y)\,|P|^{\mathsf{S}},z)\\
(9) & |\mathsf{Imp}_E(\langle a\rangle\mathsf{Id}(z,a),\langle b\rangle N,(x)P)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Imp}_L(\langle b\rangle\,|N|^{\mathsf{S}},(x)\,|P|^{\mathsf{S}},z)\\
(10) & |\mathsf{Not}_E(\langle a\rangle\mathsf{Id}(z,a),\langle b\rangle N)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Not}_L(\langle b\rangle\,|N|^{\mathsf{S}},z)
\end{array}
$$

Otherwise:

$$
\begin{array}{lrcl}
(11) & |\mathsf{And}^i_E(\langle a\rangle M,(x)N)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle\,|M|^{\mathsf{S}},(y)\mathsf{And}^i_L((x)\,|N|^{\mathsf{S}},y))\\
(12) & |\mathsf{Or}_E(\langle a\rangle M,(x)N,(y)P)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle\,|M|^{\mathsf{S}},(z)\mathsf{Or}_L((x)\,|N|^{\mathsf{S}},(y)\,|P|^{\mathsf{S}},z))\\
(13) & |\mathsf{Imp}_E(\langle a\rangle M,\langle b\rangle N,(x)P)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle\,|M|^{\mathsf{S}},(y)\mathsf{Imp}_L(\langle b\rangle\,|N|^{\mathsf{S}},(x)\,|P|^{\mathsf{S}},y))\\
(14) & |\mathsf{Not}_E(\langle a\rangle M,\langle b\rangle N)|^{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle\,|M|^{\mathsf{S}},(y)\mathsf{Not}_L(\langle b\rangle\,|N|^{\mathsf{S}},y))
\end{array}
$$

where in (11) $y \notin FN((x)|N|^{\mathsf{S}})$ (12) $z \notin FN((x)|N|^{\mathsf{S}},(y)|P|^{\mathsf{S}})$

(13) $y \notin FN(\langle b\rangle|N|^{\mathsf{S}},(x)|P|^{\mathsf{S}})$ (14) $y \notin FN(\langle b\rangle|N|^{\mathsf{S}})$

Figure 3.8: Translation from natural deduction proofs to sequent proofs.

$$
\begin{array}{lrcl}
(1) & |\mathsf{Ax}(x,a)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Id}(x,a) \\
(2) & |\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{And}_I(\langle a\rangle\,|M|^{\mathsf{N}},\langle b\rangle\,|N|^{\mathsf{N}},c) \\
(3) & |\mathsf{Or}^i_R(\langle a\rangle M,b)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Or}^i_I(\langle a\rangle\,|M|^{\mathsf{N}},b) \\
(4) & |\mathsf{Imp}_R((x)\langle a\rangle M,b)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Imp}_I((x)\langle a\rangle\,|M|^{\mathsf{N}},b) \\
(5) & |\mathsf{Not}_R((x)M,a)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Not}_I((x)\,|M|^{\mathsf{N}},a) \\
(6) & |\mathsf{And}^i_L((x)M,y)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(y,a),(x)\,|M|^{\mathsf{N}}) \\
(7) & |\mathsf{Or}_L((x)M,(y)N,z)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Or}_E(\langle a\rangle\mathsf{Id}(z,a),(x)\,|M|^{\mathsf{N}},(y)\,|N|^{\mathsf{N}}) \\
(8) & |\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Imp}_E(\langle b\rangle\mathsf{Id}(y,b),\langle a\rangle\,|M|^{\mathsf{N}},(x)\,|N|^{\mathsf{N}}) \\
(9) & |\mathsf{Not}_L(\langle a\rangle M,x)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Not}_E(\langle b\rangle\mathsf{Id}(x,b),\langle a\rangle\,|M|^{\mathsf{N}}) \\
(10) & |\mathsf{Cut}(\langle a\rangle M,(x)\mathsf{And}^i_L((y)N,x))|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{And}^i_E(\langle a\rangle|M|^{\mathsf{N}},(y)|N|^{\mathsf{N}}) \\
(11) & |\mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Or}_L((y)N,(z)P,x))|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Or}_E(\langle a\rangle|M|^{\mathsf{N}},(y)|N|^{\mathsf{N}},(z)|P|^{\mathsf{N}}) \\
(12) & |\mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Imp}_L(\langle b\rangle Q,(y)N,x))|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Imp}_E(\langle a\rangle|M|^{\mathsf{N}},\langle b\rangle|Q|^{\mathsf{N}},(y)|N|^{\mathsf{N}}) \\
(13) & |\mathsf{Cut}(\langle a\rangle M,(x)\mathsf{Not}_L(\langle b\rangle Q,x))|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Not}_E(\langle a\rangle|M|^{\mathsf{N}},\langle b\rangle|Q|^{\mathsf{N}})
\end{array}
$$

Otherwise:

$$
\begin{array}{lrcl}
(14) & |\mathsf{Cut}(\langle a\rangle M,(x)N)|^{\mathsf{N}} & \stackrel{\text{def}}{=} & \mathsf{Subst}(\langle a\rangle|M|^{\mathsf{N}},(x)|N|^{\mathsf{N}})
\end{array}
$$

where in (10) $x \notin FN((y)N)$ (11) $x \notin FN((y)N,(z)P)$
(12) $x \notin FN((y)N,\langle b\rangle Q)$ (13) $x \notin FN(\langle b\rangle Q)$

Figure 3.9: Translation from sequent proofs to natural deduction proofs.

Theorem 3.3.3 (Soundness and Completeness)**:**

- If $M \in \mathrm{K}$ with the typing judgement $\Gamma \triangleright M \triangleright \Delta$, then $|M|^{\mathsf{S}} \in \mathfrak{T}$ with the typing judgement $\Gamma \triangleright |M|^{\mathsf{S}} \triangleright \Delta$.
- If $M \in \mathfrak{T}$ with the typing judgement $\Gamma \triangleright M \triangleright \Delta$, then $|M|^{\mathsf{N}} \in \mathrm{K}$ with the typing judgement $\Gamma \triangleright |M|^{\mathsf{N}} \triangleright \Delta$.

Proof. Both by routine induction on the structure of M. □

Before we can attack the correspondence question with our cut-elimination procedure, we have to define a normalisation procedure for the natural deduction calculus. Although we shall define this procedure using terms, the corresponding notion of a normal term is best explained on the level of proofs: unlike in our sequent calculi where it is immediately clear what is meant by a normal term (absence of the cut-term constructor), things are more delicate in natural deduction calculi, as already seen in NJ. To formally define the notion of a normal natural deduction proof, the following definition will be useful.

Definition 3.3.4 (Maximal Segment)**:** A *maximal segment* in a natural deduction proof is a sequence of occurrences of a labelled formula $a : B$, say $(a : B)_1, \ldots, (a : B)_n$, such that

- the first occurrence, $(a:B)_1$, is a labelled formula which is eliminated and not introduced by an axiom.
- $(a : B)_{i+1}$ follows $(a : B)_i$, if $a : B$ occurs in a conclusion and premise of an inference rule.
- the last occurrence of $a : B$ either appears in an axiom or is freshly introduced by an introduction rule.

Now the notion of a normal natural deduction proof is as follows.

Definition 3.3.5 (Normal Natural Deduction Proof)**:** A *normal* natural deduction proof contains neither a maximal segment nor an instance of Subst.

As mentioned earlier, we can easily prove that normal natural deduction proofs respect the subformula property. For this notice that all elimination rules must have as leftmost premise an axiom which introduces the formula that is eliminated. For example the $\vee_E$-rule in a normal natural deduction proof must be of the form

$$\dfrac{\overline{z:B\vee C,\Gamma\vdash\Delta,a:B\vee C} \quad x:B,z:B\vee C,\Gamma\vdash\Delta \quad y:C,z:B\vee C,\Gamma\vdash\Delta}{z:B\vee C,\Gamma\vdash\Delta}\vee_E\ .$$

Clearly, this instance of $\vee_E$ satisfies the subformula property. Let us analyse the three other types of instances of this rule and show in each case that they cannot occur in a normal natural deduction proof. First, assume the leftmost premise is an axiom which does not introduce a. Then we have a maximal segment of length 1 (first and third clause of Definition 3.3.4). Second, assume the leftmost premise is an introduction rule. Then we either have a maximal segment of length 1, provided the introduction rule introduces freshly the formula that is eliminated, or a maximal segment of length greater than 1 (second and third clause of Definition 3.3.4). Third, if the leftmost premise is an elimination rule, then we have a maximal segment of length greater than 1 (second clause of Definition 3.3.4). So in all three cases the proof would be non-normal. Analogous remarks apply to the other elimination rules. Because in normal natural deduction proofs all instances of introduction rules satisfy the subformula property (obvious by inspection) and, as shown above, so do all instances of elimination rules, we thus have shown that the subformula property holds for all normal natural deduction proofs.

Next, we shall introduce reduction rules that transform stepwise a non-normal natural deduction proof into a normal one. We begin with the reduction rules that eliminate a maximal segment of length 1. Consider the proof

$$\dfrac{\dfrac{\Gamma \vdash \Delta, a : B}{\Gamma \vdash \Delta, b : B \vee C}\,\vee_{I_1} \quad x : B, \Gamma \vdash \Delta \quad y : C, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\,\vee_E$$

where $b : B \vee C$ forms a maximal segment of length 1, provided $b : B \vee C$ does not occur in the premise of $\vee_{I_1}$. This proof will be reduced to

$$\dfrac{\Gamma \vdash \Delta, a : B \quad x : B, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\,\text{Subst}\ .$$

The reduction rules, denoted by $\xrightarrow{\beta}$, eliminating this type of maximal segments are defined below using terms.

Definition 3.3.6 (Beta-Reductions)**:**

$$\begin{array}{rl}
\mathsf{And}^i_E(\langle c\rangle\mathsf{And}_I(\langle a_1\rangle M_1, \langle a_2\rangle M_2, c), (x)N) & \xrightarrow{\beta} \mathsf{Subst}(\langle a_i\rangle M_i, (x)N) \\
& \text{if } \mathsf{And}_I(\langle a_1\rangle M_1, \langle a_2\rangle M_2, c) \text{ freshly introduces } c \\[1ex]
\mathsf{Or}_E(\langle b\rangle\mathsf{Or}^i_I(\langle a\rangle M, b), (x_1)N_1, (x_2)N_2) & \xrightarrow{\beta} \mathsf{Subst}(\langle a\rangle M, (x_i)N_i) \\
& \text{if } \mathsf{Or}^i_I(\langle a\rangle M, b) \text{ freshly introduces } b \\[1ex]
\mathsf{Imp}_E(\langle b\rangle\mathsf{Imp}_I((x)\langle a\rangle M, b), \langle c\rangle N, (y)P) & \xrightarrow{\beta} \mathsf{Subst}(\langle c\rangle N, (x)\mathsf{Subst}(\langle a\rangle M, (y)P)) \ \text{ or} \\
& \xrightarrow{\beta} \mathsf{Subst}(\langle a\rangle\mathsf{Subst}(\langle c\rangle N, (x)M), (y)P) \\
& \text{if } \mathsf{Imp}_I((x)\langle a\rangle M, b) \text{ freshly introduces } b \\[1ex]
\mathsf{Not}_E(\langle a\rangle\mathsf{Not}_I((x)M, a), \langle b\rangle N) & \xrightarrow{\beta} \mathsf{Subst}(\langle b\rangle N, (x)M) \\
& \text{if } \mathsf{Not}_I((x)M, a) \text{ freshly introduces } a
\end{array}$$

Remark 3.3.7: There are a few subtleties in the third rule. As with the cut-reduction $\supset_R/\supset_L$, there are two ways to eliminate an introduction-elimination pair $\supset_E/\supset_I$. Consequently, we have two reduction rules. We need, however, to observe the side-conditions x not free in $FN((y)P)$ (first rule) and a not free in $FC(\langle c\rangle N)$ (second rule), so that no name or co-name capture occurs. This can always be achieved by renaming some binders.

To eliminate all other maximal segments, we introduce commuting reductions. Let us first give an example. Consider the proof

$$
\begin{array}{c}
\dfrac{\Gamma' \vdash \Delta', a:B \qquad \Gamma' \vdash \Delta', b:C}{\Gamma' \vdash \Delta', c:B\wedge C}\,\wedge_I \\
\vdots \\
\dfrac{\Gamma \vdash \Delta, c:B\wedge C \qquad\qquad x:B, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\,\wedge_{E_1}
\end{array}
$$

where the $\wedge_{E_1}$-rule needs to be permuted so as to yield the proof

$$
\begin{array}{c}
\dfrac{\dfrac{\Gamma^\bullet \vdash \Delta^\bullet, a:B \qquad \Gamma^\bullet \vdash \Delta^\bullet, b:C}{\Gamma^\bullet \vdash \Delta^\bullet, c:B\wedge C}\,\wedge_I \qquad x:B, \Gamma^\bullet \vdash \Delta^\bullet}{\Gamma^\bullet \vdash \Delta^\bullet}\,\wedge_{E_1} \\
\vdots \\
\Gamma \vdash \Delta
\end{array}
$$

Here $\Gamma^\bullet$ stands for the context $\Gamma \cup \Gamma'$ (similarly $\Delta^\bullet$). We realise this kind of operation using a proof substitution of the form:

$$
\begin{array}{ll}
M \,\langle\!\langle \mathsf{And}^i_E(\langle a\rangle\_, (x)N)\rangle\!\rangle & \qquad M \,\langle\!\langle \mathsf{Imp}_E(\langle a\rangle\_, \langle b\rangle N, (x)P)\rangle\!\rangle \\
M \,\langle\!\langle \mathsf{Or}_E(\langle a\rangle\_, (x)N, (y)P)\rangle\!\rangle & \qquad M \,\langle\!\langle \mathsf{Not}_E(\langle a\rangle\_, \langle b\rangle N)\rangle\!\rangle
\end{array}
$$

The idea behind this proof substitution is to move the elimination rule inside the term M until either a corresponding introduction rule or a corresponding axiom is reached. In these two cases the term M is "plugged" into the place of the bar. If there is no corresponding introduction rule nor a corresponding axiom, then the proof substitution just vanishes. Figure 3.10 gives the complete definition of $M\langle\!\langle\sigma\rangle\!\rangle$. Our commuting reductions, denoted by $\xrightarrow{\gamma}$, are then defined as follows.

Definition 3.3.8 (Commuting Reductions)**:**

$$
\left.
\begin{array}{rcl}
\mathsf{And}^i_E(\langle a\rangle M, (x)N) & \xrightarrow{\gamma} & M\langle\!\langle \mathsf{And}^i_E(\langle a\rangle\_, (x)N)\rangle\!\rangle \\
\mathsf{Or}_E(\langle a\rangle M, (x)N, (y)P) & \xrightarrow{\gamma} & M\langle\!\langle \mathsf{Or}_E(\langle a\rangle\_, (x)N, (y)P)\rangle\!\rangle \\
\mathsf{Imp}_E(\langle a\rangle M, \langle b\rangle N, (x)P) & \xrightarrow{\gamma} & M\langle\!\langle \mathsf{Imp}_E(\langle a\rangle\_, \langle b\rangle N, (x)P)\rangle\!\rangle \\
\mathsf{Not}_E(\langle a\rangle M, \langle b\rangle N) & \xrightarrow{\gamma} & M\langle\!\langle \mathsf{Not}_E(\langle a\rangle\_, \langle b\rangle N)\rangle\!\rangle
\end{array}
\right\}
\begin{array}{l}
\text{if } M \text{ does not} \\
\text{freshly introduce } a
\end{array}
$$

What remains is to define reduction rules eliminating instances of Subst. This term constructor was introduced in order to match all reductions of commuting cuts, in particular the ones in Lafont's example. Therefore, we shall define the reduction rules for Subst such that we can capture the non-deterministic behaviour of commuting cuts. To do so, we introduce another proof substitution, written as

$$
M[\![a := (x)N]\!] \quad \text{or} \quad N[\![x := \langle a\rangle M]\!].
$$

Figure 3.11 gives its complete definition, which is formulated so that, given the translation $|\_|^{\mathsf{N}}$, there is a perfect match with proof substitutions of the form $[\sigma]$. Consequently, we have the following reduction rule for Subst.

Let $\langle\!\langle\sigma\rangle\!\rangle$ be of the form $\langle\!\langle \mathsf{And}^i_E(\langle c\rangle\_, (x)P)\rangle\!\rangle$, then:

$$\mathsf{Id}(y,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{And}^i_E(\langle c\rangle \mathsf{Id}(y,c),(x)P) \tag{1}$$

$$\mathsf{And}_I(\langle a\rangle M,\langle b\rangle N,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{And}^i_E(\langle c\rangle \mathsf{And}_I(\langle a\rangle M\langle\!\langle\sigma\rangle\!\rangle,\langle b\rangle N\langle\!\langle\sigma\rangle\!\rangle,c),(x)P) \tag{2}$$

Let $\langle\!\langle\sigma\rangle\!\rangle$ be of the form $\langle\!\langle \mathsf{Or}_E(\langle c\rangle\_, (x)P, (y)Q)\rangle\!\rangle$, then:

$$\mathsf{Id}(y,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{Or}_E(\langle c\rangle \mathsf{Id}(y,c),(x)P,(y)Q) \tag{3}$$

$$\mathsf{Or}^i_I(\langle a\rangle M,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{Or}_E(\langle c\rangle \mathsf{Or}^i_I(\langle a\rangle M\langle\!\langle\sigma\rangle\!\rangle,c),(x)P,(y)Q) \tag{4}$$

Let $\langle\!\langle\sigma\rangle\!\rangle$ be of the form $\langle\!\langle \mathsf{Imp}_E(\langle c\rangle\_, \langle b\rangle P, (x)Q)\rangle\!\rangle$, then:

$$\mathsf{Id}(y,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{Imp}_E(\langle c\rangle \mathsf{Id}(y,c),\langle b\rangle P,(x)Q) \tag{5}$$

$$\mathsf{Imp}_I((x)\langle a\rangle M,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{Imp}_E(\langle c\rangle \mathsf{Imp}_I((x)\langle a\rangle M\langle\!\langle\sigma\rangle\!\rangle,c),\langle b\rangle P,(x)Q) \tag{6}$$

Let $\langle\!\langle\sigma\rangle\!\rangle$ be of the form $\langle\!\langle \mathsf{Not}_E(\langle c\rangle\_, \langle b\rangle P)\rangle\!\rangle$, then:

$$\mathsf{Id}(y,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{Not}_E(\langle c\rangle \mathsf{Id}(y,c),\langle b\rangle P) \tag{7}$$

$$\mathsf{Not}_I((x)M,c)\langle\!\langle\sigma\rangle\!\rangle \stackrel{\text{def}}{=} \mathsf{Not}_E(\langle c\rangle \mathsf{Not}_I((x)M\langle\!\langle\sigma\rangle\!\rangle,c),\langle b\rangle P) \tag{8}$$

Otherwise:

$$\begin{aligned}
\mathsf{Id}(y,a)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Id}(x,a)\\
\mathsf{And}_I(\langle a\rangle M,\langle b\rangle N,c)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{And}_I(\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,\langle b\rangle\, N\langle\!\langle\sigma\rangle\!\rangle,c)\\
\mathsf{And}^i_E(\langle a\rangle M,(x)N)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{And}^i_E(\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,(x)\, N\langle\!\langle\sigma\rangle\!\rangle)\\
\mathsf{Or}^i_I(\langle a\rangle M,b)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Or}^i_I(\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,b)\\
\mathsf{Or}_E(\langle a\rangle M,(x)N,(y)P)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Or}_E(\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,(x)\, N\langle\!\langle\sigma\rangle\!\rangle,(y)\, P\langle\!\langle\sigma\rangle\!\rangle)\\
\mathsf{Imp}_I((x)\langle a\rangle M,b)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Imp}_I((x)\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,b)\\
\mathsf{Imp}_E(\langle a\rangle M,\langle b\rangle N,(x)P)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Imp}_E(\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,\langle b\rangle\, N\langle\!\langle\sigma\rangle\!\rangle,(x)\, P\langle\!\langle\sigma\rangle\!\rangle)\\
\mathsf{Not}_I((x)M,b)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Not}_I((x)\, M\langle\!\langle\sigma\rangle\!\rangle,b)\\
\mathsf{Not}_E(\langle a\rangle M,\langle b\rangle N)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Not}_E(\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,\langle b\rangle\, N\langle\!\langle\sigma\rangle\!\rangle)\\
\mathsf{Subst}(\langle a\rangle M,(x)N)\langle\!\langle\sigma\rangle\!\rangle &\stackrel{\text{def}}{=} \mathsf{Subst}(\langle a\rangle\, M\langle\!\langle\sigma\rangle\!\rangle,(x)\, N\langle\!\langle\sigma\rangle\!\rangle)
\end{aligned}$$

Figure 3.10: Proof substitution in natural deduction I.

Let $\llbracket\sigma\rrbracket$ be of the form $\llbracket x := \langle c\rangle P\rrbracket$, then:

$$\mathsf{Id}(x,a)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} P[c\mapsto a] \tag{1}$$

Let $\llbracket\sigma\rrbracket$ be of the form $\llbracket c := (x)P\rrbracket$, then:

$$\mathsf{Id}(x,c)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} P[y\mapsto x] \tag{2}$$

$$\mathsf{And}_I(\langle a\rangle M,\langle b\rangle N,c)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Subst}(\langle c\rangle \mathsf{And}_I(\langle a\rangle M\llbracket\sigma\rrbracket,\langle b\rangle N\llbracket\sigma\rrbracket,c),(y)P) \tag{3}$$

$$\mathsf{Or}^i_I(\langle a\rangle M,c)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Subst}(\langle c\rangle \mathsf{Or}^i_I(\langle a\rangle M\llbracket\sigma\rrbracket,c),(y)P) \tag{4}$$

$$\mathsf{Imp}_I((x)\langle a\rangle M,c)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Subst}(\langle c\rangle \mathsf{Imp}_I((x)\langle a\rangle M\llbracket\sigma\rrbracket,c),(y)P) \tag{5}$$

$$\mathsf{Not}_I((x)M,c)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Subst}(\langle c\rangle \mathsf{Not}_I((x)M\llbracket\sigma\rrbracket,c),(y)P) \tag{6}$$

Otherwise:

$$\mathsf{Id}(y,a)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Id}(y,a)$$

$$\mathsf{And}_I(\langle a\rangle M,\langle b\rangle N,c)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{And}_I(\langle a\rangle\, M\llbracket\sigma\rrbracket,\langle b\rangle\, N\llbracket\sigma\rrbracket,c)$$

$$\mathsf{And}^i_E(\langle a\rangle M,(x)N)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{And}^i_E(\langle a\rangle\, M\llbracket\sigma\rrbracket,(x)\, N\llbracket\sigma\rrbracket)$$

$$\mathsf{Or}^i_I(\langle a\rangle M,b)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Or}^i_I(\langle a\rangle\, M\llbracket\sigma\rrbracket,b)$$

$$\mathsf{Or}_E(\langle a\rangle M,(x)N,(y)P)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Or}_E(\langle a\rangle\, M\llbracket\sigma\rrbracket,(x)\, N\llbracket\sigma\rrbracket,(y)\, P\llbracket\sigma\rrbracket)$$

$$\mathsf{Imp}_I((x)\langle a\rangle M,b)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Imp}_I((x)\langle a\rangle\, M\llbracket\sigma\rrbracket,b)$$

$$\mathsf{Imp}_E(\langle a\rangle M,\langle b\rangle N,(x)P)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Imp}_E(\langle a\rangle\, M\llbracket\sigma\rrbracket,\langle b\rangle\, N\llbracket\sigma\rrbracket,(x)\, P\llbracket\sigma\rrbracket)$$

$$\mathsf{Not}_I((x)M,b)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Not}_I((x)\, M\llbracket\sigma\rrbracket,b)$$

$$\mathsf{Not}_E(\langle a\rangle M,\langle b\rangle N)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Not}_E(\langle a\rangle\, M\llbracket\sigma\rrbracket,\langle b\rangle\, N\llbracket\sigma\rrbracket)$$

$$\mathsf{Subst}(\langle a\rangle M,(x)N)\llbracket\sigma\rrbracket \stackrel{\text{def}}{=} \mathsf{Subst}(\langle a\rangle\, M\llbracket\sigma\rrbracket,(x)\, N\llbracket\sigma\rrbracket)$$

Figure 3.11: Proof substitution in natural deduction II.

Definition 3.3.9 (Substitution Elimination)**:**

$$\begin{array}{lll}\mathsf{Subst}(\langle a\rangle M,(x)N) & \xrightarrow{\sigma} & M\{\!\![a := (x)M]\!\!\} \text{ if } M \text{ does not freshly introduce } a\text{, or}\\ & \xrightarrow{\sigma} & N\{\!\![x := \langle a\rangle M]\!\!\}\end{array}$$

To sum up the discussion about the reduction rules let us combine them in the definition of a normalisation procedure for classical natural deduction proofs.

Definition 3.3.10 (Normalisation for Classical Natural Deduction Proofs)**:**
The normalisation procedure for classical natural deduction proofs, $(\mathrm{K}, \xrightarrow{\kappa})$, is a reduction system where:

- K is the set of well-typed terms, and
- $\xrightarrow{\kappa}$ consists of beta-reductions, commuting reductions and reductions to eliminate substitutions; that is
$$\xrightarrow{\kappa} \stackrel{\text{def}}{=} \xrightarrow{\beta} \cup \xrightarrow{\gamma} \cup \xrightarrow{\sigma} \ .$$

We should like to state that $\xrightarrow{\kappa}$ respects the subject reduction property.

Proposition 3.3.11 (Subject Reduction)**:** Suppose $M \in \mathrm{K}$ with the typing judgement $\Gamma \triangleright M \triangleright \Delta$ and $M \xrightarrow{\kappa} N$, then $N \in \mathrm{K}$ with the typing judgement $\Gamma \triangleright N \triangleright \Delta$.

Proof. Analogous to proof of Proposition 2.2.10. □

As mentioned earlier, the reduction $\xrightarrow{\kappa}$ is complete in the sense that to every non-normal natural deduction proof a reduction rule can be applied. To show that every non-normal natural deduction proof can be reduced to a normal one, we have to show that $\xrightarrow{\kappa}$ is terminating. This is what we shall do next by showing the correspondence between $\xrightarrow{\kappa}$ and $\xrightarrow{cut}$.

3.4 The Correspondence between Cut-Elimination and Normalisation II

In this section we shall show the correspondence between normalisation and cut-elimination in classical logic. A byproduct, we shall show that $\xrightarrow{\kappa}$ is strongly normalising. The lengthy part in this proof is to establish the correspondence between the proof substitutions $\langle\!\langle\_\rangle\!\rangle$, $\{\!\![\_]\!\!\}$ and $[\_]$. This is what we shall do first.

Lemma 3.4.1: For all $M, N \in \mathrm{K}$, $|M|^{\mathsf{S}}[a := (x)|N|^{\mathsf{S}}] \equiv |M\{\!\![a := (x)N]\!\!\}|^{\mathsf{S}}$.

Proof. By induction on the structure of M (see Page 161). □

Lemma 3.4.2: For all $M, N \in \mathrm{K}$, $|M|^{\mathsf{S}}[x := \langle a\rangle|N|^{\mathsf{S}}] \xrightarrow{cut}{}^{*} |M\{\!\![x := \langle a\rangle N]\!\!\}|^{\mathsf{S}}$.

Proof. By induction on the structure of M (see Page 161). □

Lemma 3.4.3: For all $M, N, P, Q \in \mathrm{K}$ with $z \notin FN((x)|N|^{\mathsf{S}}, (y)|P|^{\mathsf{S}}, \langle b\rangle|Q|^{\mathsf{S}})$ we have

$$\begin{aligned}
|M|^{\mathsf{S}}[a := (z)\mathsf{And}^i_L((x)\,|N|^{\mathsf{S}}, z)] \;&\equiv\; |M\langle\!\langle \mathsf{And}^i_E(\langle a\rangle_, (x)N)\rangle\!\rangle|^{\mathsf{S}} \\
|M|^{\mathsf{S}}[a := (z)\mathsf{Or}_L((x)\,|N|^{\mathsf{S}}, (y)\,|P|^{\mathsf{S}}, z)] \;&\equiv\; |M\langle\!\langle \mathsf{Or}_E(\langle a\rangle_, (x)N, (y)P)\rangle\!\rangle|^{\mathsf{S}} \\
|M|^{\mathsf{S}}[a := (z)\mathsf{Imp}_L(\langle b\rangle\,|Q|^{\mathsf{S}}, (y)\,|P|^{\mathsf{S}}, z)] \;&\equiv\; |M\langle\!\langle \mathsf{Imp}_E(\langle a\rangle_, \langle b\rangle Q, (y)P)\rangle\!\rangle|^{\mathsf{S}} \\
|M|^{\mathsf{S}}[a := (z)\mathsf{Not}_L(\langle b\rangle\,|Q|^{\mathsf{S}}, z)] \;&\equiv\; |M\langle\!\langle \mathsf{Not}_E(\langle a\rangle_, \langle b\rangle Q)\rangle\!\rangle|^{\mathsf{S}}
\end{aligned}$$

Proof. By induction on the structure of M (see Page 162). ☐

Now we are in a position to show that under $|\_|^{\mathsf{S}}$ every normalisation reduction maps onto a series of cut-reductions.

Theorem 3.4.4: For all $M, N \in \mathrm{K}$, if $M \xrightarrow{\kappa} N$, then $|M|^{\mathsf{S}} \xrightarrow{cut}{}^{+} |N|^{\mathsf{S}}$.

Proof. By induction on the definition of $\xrightarrow{\kappa}$ (see Page 163). ☐

Since we have shown that $\xrightarrow{\kappa}$-reductions can be simulated by $\xrightarrow{cut}$, we expect that every $\xrightarrow{\kappa}$-reduction sequence terminates. This is indeed the case.

Theorem 3.4.5: The reduction system $(\mathrm{K}, \xrightarrow{\kappa})$ is strongly normalising.

Proof. By Theorem 3.4.4 we know that every reduction sequence of $\xrightarrow{\kappa}$ maps to a reduction sequence of $\xrightarrow{cut}$. Because $(\mathcal{T}, \xrightarrow{cut})$ is strongly normalising, $(\mathrm{K}, \xrightarrow{\kappa})$ must be, too. ☐

We proceed with showing the correspondence in the opposite direction. First we show the correspondence between the proof substitutions—again this involves rather lengthy calculations.

Lemma 3.4.6: For all $M, N \in \mathcal{T}$, $|M|^{\mathsf{N}}[\![x := \langle a\rangle|N|^{\mathsf{N}}]\!] \equiv |M[x := \langle a\rangle N]|^{\mathsf{N}}$.

Proof. By induction on the structure of M (see Page 163). ☐

Lemma 3.4.7: For all $M, N \in \mathcal{T}$, if N freshly introduces x and is not an axiom, then

$$|M|^{\mathsf{N}}\langle\!\langle N_{\mathsf{N}}\rangle\!\rangle \quad \xrightarrow{\kappa}{}^{*} \quad |M[a := (x)N]|^{\mathsf{N}}$$

where $N_{\mathsf{N}} \equiv \begin{cases} \mathsf{And}^i_E(\langle a\rangle\_, (y)|P|^{\mathsf{N}}) \\ \mathsf{Or}_E(\langle a\rangle\_, (y)|P|^{\mathsf{N}}, (z)|Q|^{\mathsf{N}}) \\ \mathsf{Imp}_E(\langle a\rangle\_, \langle b\rangle|P|^{\mathsf{N}}, (y)|Q|^{\mathsf{N}}) \\ \mathsf{Not}_E(\langle a\rangle\_, \langle b\rangle|P|^{\mathsf{N}}) \end{cases}$ provided $N \equiv \begin{cases} \mathsf{And}^i_L((y)P, x) \\ \mathsf{Or}_L((y)P, (z)Q, x) \\ \mathsf{Imp}_L(\langle b\rangle P, (y)Q, x) \\ \mathsf{Not}_L(\langle b\rangle P, x) \end{cases}$

Proof. By induction on the structure of M (see Page 164). ☐

Lemma 3.4.8: For all $M, N \in \mathcal{T}$, provided N does not freshly introduce x, except where N is an axiom, then

$$|M[a := (x)N]|^{\mathsf{N}} \equiv |M|^{\mathsf{N}} ⦇a := (x)|N|^{\mathsf{N}}⦈ \; .$$

Proof. By induction on the structure of M (see Page 165). ☐

The correspondence between the reduction rules is now a matter of simple calculations.

Theorem 3.4.9: For all $M, N \in \mathcal{T}$, if $M \xrightarrow{cut} N$, then $|M|^{\mathsf{N}} \xrightarrow{\kappa}{}^{*} |N|^{\mathsf{N}}$.

Proof. By induction on the definition of $\xrightarrow{cut}$ (see Page 165). ☐

To sum up, we have shown in this section the close correspondence between normalisation and cut-elimination in propositional classical logic; by Theorems 3.4.4 and 3.4.9 we have the correspondences listed below.

$$\begin{tikzcd} S \arrow[r, "\kappa"] \arrow[d, "\mathsf{S}"'] & T \arrow[d, "\mathsf{S}"] \\ S' \arrow[r, "cut"', "+"] & T' \end{tikzcd} \qquad \begin{tikzcd} M \arrow[r, "cut"] \arrow[d, "\mathsf{N}"'] & N \arrow[d, "\mathsf{N}"] \\ M' \arrow[r, "\kappa"', "*"] & N' \end{tikzcd}$$

These correspondences can be extended to first-order classical logic and to other variants of sequence-conclusion natural deduction calculi, but the details are omitted.

3.5 Notes

In this section, we shall first compare our work with earlier work by Ungar [1992], which also studies the correspondence between cut-elimination and normalisation, and second describe briefly the $\lambda\mu$-calculus developed by Parigot [1992]. We shall show that strong normalisation for (propositional) $\lambda\mu$ can be inferred by a simple translation from our strong normalisation result, but not *vice versa*.

3.5.1 Comparison with Ungar's Approach to the Correspondence Question

While Zucker [1974] argued that in the $(\wedge, \supset, \vee)$-fragment of intuitionistic logic the lack of strong normalisation of cut-elimination is responsible for the failure of the correspondence, we have shown that in fact the lack of certain reduction rules in NJ is responsible (we fixed the problem with strong normalisation, but could not establish the correspondence using NJ). Ungar made the same observation[4] and already extended the correspondence to all connectives. He even established the correspondence in classical logic. However, his methods, and the details of his results, are quite different from ours. For example, instead of allaying the problem Zucker had with strong normalisation, he showed

[4] "The problem arises with the conversion steps which allow a cut rule to be permuted upwards past an application of $\exists$- or $\vee$-left in the derivation of its left-hand premise...[T]hese do not in general correspond to permutative reductions of the sort prescribed by Prawitz for NJ." [Ungar, 1992, Page 41]

the correspondence between a cut-elimination procedure and a normalisation procedure that are both *not* strongly normalising. Also, Ungar's results build upon the rather complicated natural deduction calculus introduced by Shoesmith and Smiley [1978]. Therefore it seemed prudent for us to reconsider the correspondence between cut-elimination and normalisation.

Let us briefly study Shoesmith and Smiley's natural deduction calculus, which was used by Ungar. In this calculus, the inference rules for $\wedge$ and $\supset$ are the same as in Gentzen's NJ-calculus, but the $\vee$-elimination rule is the form

$$\frac{B\vee C}{B \quad C}\,\vee_E \,.$$

Here B and C are separate conclusions. Having inferences with two separate conclusions has the profound effect that, in general, proofs are not trees, but graphs. Here is a proof (written schematically) in their natural deduction calculus.

$$\begin{array}{ccc} \multicolumn{2}{c}{B} & \\ \cline{1-2} C & D & \\ \cline{1-1}\cline{2-3} E & F & G \\ \cline{1-2}\cline{3-3} \multicolumn{2}{c}{H} & I \end{array}$$

Since proofs are now graphs, the calculus of Shoesmith and Smiley has several rather unpleasant features. For example, if one applies an inference rule, one has to take into account the shape of the entire proof. By this we mean that from a proof of B and a proof of C, one cannot immediately construct a proof of $B\wedge C$, because the proofs of B and C must not have any subproof in common. For example the derivation

$$\cfrac{\cfrac{B\vee C}{B \quad C}\,\vee_E}{B\wedge C}\,\wedge_I$$

is not allowed. This restriction applies to all inference rules with two premises. Another unpleasant feature of this natural deduction calculus concerns the operation of proof substitution. Consider the proof

$$\begin{array}{cc} B & C \\ \vdots & \vdots \end{array}$$

with two assumptions, B and C, and the proof

$$\frac{\begin{array}{c}\vdots\\ B\vee C\end{array}}{B \quad C}\,\vee_E$$

ending with two conclusions, B and C. Substituting the latter proof for the assumptions

B and C in the former one gives the proof

$$\dfrac{\begin{array}{c}\vdots\\ B\vee C\end{array}}{\begin{array}{cc}B & C\\ \vdots & \vdots\end{array}}$$

The graph-like structure of proofs, however, causes the definition of this substitution operation to be very complicated; for example a simple inductive definition is impossible (see [Ungar, 1992, Pages 55–76]). Moreover, it seems very difficult to define a sequent-style formalisation for this natural deduction calculus, and it is highly unlikely that terms can be annotated to its proofs. In the end, we found that, even if Shoesmith and Smiley's multiple-conclusion natural deduction calculus seems to be better suited for classical proofs (they do not have to be encoded via a double negation translation as in NK), it is a very complicated calculus.

Much more convenient is the sequence-conclusion natural deduction calculus introduced by Boričić [1985]. Our work presented in Section 3.3 builds upon a slightly modified version of this calculus. We presented for the corresponding proofs a term assignment, which is similar to the term assignment for our sequent proofs, i.e., the terms include two sorts of binders, $(\_)$ and $\langle\_\rangle$, and record precisely which formulae are discharged and introduced by inferences. A different type of term assignment for Boričić's sequence-conclusion natural deduction calculus was given by Parigot [1992]. The corresponding term calculus was named $\lambda\mu$-calculus, which we shall study briefly next.

3.5.2 Parigot's $\lambda\mu$-Calculus

The $\lambda\mu$-calculus can be seen as a simple extension of the simply-typed lambda calculus. The extension is such that via the Curry-Howard correspondence $\lambda\mu$-terms no longer correspond to proofs in Gentzen's NJ-calculus, but to proofs in Boričić's sequence-conclusion natural deduction calculus (see Page 81). In effect, Parigot gave an elegant solution to the problem of how to modify simply-typed lambda terms so that their single type is generalised to a sequence of types. The technical crux of Parigot's solution is the addition of two new term constructors to the simply-typed lambda calculus. The corresponding raw $\lambda\mu$-terms are given by the grammar

$$\begin{array}{rcll}
M, N & ::= & x & \text{variable}\\
& | & \lambda x.M & \text{abstraction}\\
& | & M\,N & \text{application}\\
& | & [a]M & \text{passivate}\\
& | & \mu a.M & \text{activate}
\end{array}$$

where x is taken from a set of variables and a from a set of μ-variables. As a consequence, he could consider the sequence of conclusions of proofs in Boričić's calculus as a whole, but distinguishing one of them as *active*—in what follows annotated with a disc—and the others as *passive*. The latter are signified by being labelled with μ-variables. Typing judgements in $\lambda\mu$ have thus the form

$$\Gamma \triangleright M : B^{\bullet}, \Delta$$

where Γ is a set of (variable,formula) pairs and Δ a set of (μ-variable,formula) pairs.

Since $\lambda\mu$ is a natural deduction calculus, its inference rules are concerned with introducing or eliminating formulae in the succedent. However, they are restricted so that they act on active formulae only, as can be seen in the term formation rules for abstraction and application.

$$\frac{x : B, \Gamma \triangleright M : C^{\bullet}, \Delta}{\Gamma \triangleright \lambda x.M : (B \supset C)^{\bullet}, \Delta} \supset_I \qquad \frac{\Gamma \triangleright M : (B \supset C)^{\bullet}, \Delta \quad \Gamma \triangleright N : B^{\bullet}, \Delta}{\Gamma \triangleright M\, N : C^{\bullet}, \Delta} \supset_E .$$

The formation rules for the new term constructors are called, using terminology introduced by Bierman [1998], *passivate* and *activate*. They are as follows.[5]

$$\frac{\Gamma \triangleright M : B^{\bullet}, \Delta}{\Gamma \triangleright [a]M : \bot^{\bullet}, a : B, \Delta} \qquad \frac{\Gamma \triangleright M : \bot^{\bullet}, a : B, \Delta}{\Gamma \triangleright \mu a.M : B^{\bullet}, \Delta}$$

The first rule changes the status of the active formula to become passive. The resulting term has then the active type $\bot$, a distinguished atomic formula. The other rule works similar, but in the reverse direction.

Before passing to the translations between $\lambda\mu$-terms and terms belonging to K, let us briefly comment on the syntactic conventions in both systems. It seems the differences are chiefly a matter of taste: whereas in $\lambda\mu$ a typing judgement has only a single active formula, and therefore it does not need to be recorded in the term, except of course in the cases where another formula becomes active; in our term calculus, on the other hand, all formulae in the succedent are active, so to speak, and therefore it has to be recorded explicitly which formulae are discharged and introduced. Clearly, both syntactic conventions can be "simulated" by the other. On the other hand, considering the amount of prolixity, the syntax of $\lambda\mu$ is more concise in the average case, but less so in the worst case. For example in $\lambda\mu$ we can form the term

$$\mu a.[a]\mu a.[a]x$$

which corresponds to the term $\mathsf{Id}(x, a)$ in our calculus.

Our natural deduction calculus K is a slightly modified variant of Boričić's sequence-conclusion natural deduction calculus (see Figure 3.7). We made the modifications in order to facilitate the translation with our sequent calculus $\mathcal{T}$. It may be therefore surprising, that there are also relatively simple translations for K to and from the $\lambda\mu$-calculus.

[5] We use the convention introduced by Bierman [1998] where all terms have an active type.

The translations for both directions are listed below.

$|\_|^{\mu} : \mathrm{K} \to \lambda\mu$

$$
\begin{array}{rcl}
|\mathsf{Id}(x,a)|^{\mu} & \stackrel{\text{def}}{=} & [a]x \\
|\mathsf{Imp}_I((x)\langle a\rangle M, b)|^{\mu} & \stackrel{\text{def}}{=} & [b](\lambda x.\mu a.|M|^{\mu}) \\
|\mathsf{Imp}_E(\langle a\rangle M, \langle b\rangle N, (x)P)|^{\mu} & \stackrel{\text{def}}{=} & |P|^{\mu}[x := (\mu a.|M|^{\mu})\ (\mu b.|N|^{\mu})] \\
|\mathsf{Subst}(\langle a\rangle M, (x)N)|^{\mu} & \stackrel{\text{def}}{=} & |N|^{\mu}[x := \mu a.|M|^{\mu}]
\end{array}
$$

$|\_|^{\kappa} : \lambda\mu \to \mathrm{K}$

$$
\begin{array}{rcl}
|x|^{\kappa}_{c} & \stackrel{\text{def}}{=} & \mathsf{Id}(x,c) \\
|\lambda x.M|^{\kappa}_{c} & \stackrel{\text{def}}{=} & \mathsf{Imp}_I((x)\langle a\rangle|M|^{\kappa}_{a}, c) \\
|MN|^{\kappa}_{c} & \stackrel{\text{def}}{=} & \mathsf{Imp}_E(\langle a\rangle|M|^{\kappa}_{a}, \langle b\rangle|N|^{\kappa}_{b}, (x)\mathsf{Id}(x,c)) \\
|[c]M|^{\kappa} & \stackrel{\text{def}}{=} & |M|^{\kappa}_{c} \\
|\mu a.M|^{\kappa}_{c} & \stackrel{\text{def}}{=} & |M[a \mapsto c]|^{\kappa}
\end{array}
$$

Note that in the case where the $\lambda\mu$-term has an active type different from $\perp$, the clauses of $|\_|^{\kappa}$ are indexed with a label. It is easy to verify that these translations respect typing judgements.

Proposition 3.5.1:

- If $M \in \mathrm{K}$ with the typing judgement $\Gamma \triangleright M \triangleright \Delta$, then $|M|^{\mu} \in \lambda\mu$ with the typing judgement $\Gamma \triangleright |M|^{\mu} \triangleright \perp^{\bullet}, \Delta$.

- Assume c is a unique label, not occurring anywhere in M. If $M \in \lambda\mu$ with the typing judgement $\Gamma \triangleright M \triangleright B^{\bullet}, \Delta$, then $|M|^{\kappa}_{c} \in \mathrm{K}$ with the typing judgement $\Gamma \triangleright |M|^{\kappa}_{c} \triangleright c : B, \Delta$. If $M \in \lambda\mu$ with the typing judgement $\Gamma \triangleright M \triangleright \perp^{\bullet}, \Delta$, then $|M|^{\kappa} \in \mathrm{K}$ with the typing judgement $\Gamma \triangleright |M|^{\kappa} \triangleright \Delta$.

Proof. Both cases by induction on the structure of M. $\square$

There are a number of reduction rules associated with the $\lambda\mu$-calculus, which we however omit and instead refer the reader to [Parigot, 1992] and [Bierman, 1998]. Parigot proved for the corresponding reduction system the following two properties.

Theorem 3.5.2 (Parigot)**:** The reduction system of the $\lambda\mu$-calculus is:

(i) strongly normalising, and
(ii) confluent.

Strong normalisation can, for example, be obtain using $|\_|^{\kappa}$ and appealing to our strong normalisation result for $(\mathrm{K}, \xrightarrow{\kappa})$. Unfortunately this method does not extend to the second-order version of $\lambda\mu$, for which Parigot showed strong normalisation using a candidates method.

To conclude the discussion about $\lambda\mu$, let us consider the confluence property of $\lambda\mu$. At first glance, it may be surprising that $\lambda\mu$ satisfies this property, but at a second it is less so, if we take into account that the explicit substitution operator in K was a prerequisite in order to obtain a correspondence with our (non-confluent) cut-elimination procedure. As remarked earlier, translating this substitution operator as

$$|\mathsf{Subst}(\langle a\rangle M,(x)N)|^{\mu} \stackrel{\text{def}}{=} |N|^{\mu}[x := \mu a.|M|^{\mu}]$$

then all reductions that M can perform are "lost" provided x is not free in N. Therefore the reduction system of $\lambda\mu$ is more restricted, and strong normalisation for $(\mathrm{K}, \xrightarrow{\kappa})$ cannot be inferred from Parigot's strong normalisation result (using the simple translation $|\_|^{\mu}$). Given the confluence property of $\lambda\mu$, more complicated translations that would establish strong normalisation for $(\mathrm{K}, \xrightarrow{\kappa})$ seem highly unlikely.

Chapter 4

Applications of Cut-Elimination

> It is reasonable to hope that the relationship between computation and mathematical logic will be as fruitful in the next century as that between analysis and physics in the last.
>
> —J. McCarthy
> in A Basis for a Mathematical Theory of Computation, 1963.

Typical treatments of applications of cut-elimination focus on proof-theoretic issues; e.g., subformula property, Herbrand interpretations, interpolation theorem, numerical bounds on the cut-elimination process. We, however, shall use cut-elimination for extracting data from a particular classical proof and show that computation in a simple, non-deterministic language can be simulated by cut-elimination in a fragment of classical logic. To study these applications, we found invaluable an implementation that computes all normal forms reachable from a classical proof. Since our cut-elimination procedures are non-deterministic, a naïve attempt to implement them is rather laborious: it would, for example, require backtracking. We shall therefore describe some modifications we can make to $\xrightarrow{aux}$ that lead to a simple implementation, which does not require backtracking.

Functional programming languages seem most suitable for implementing our reduction systems, since they have facilities for declaring algebraic datatypes and provide mechanisms for pattern matching. In effect, the operations defined on our terms can be implemented in these languages with great economy of expression. We have chosen OCaml in favour of other functional languages, because it allows pattern cases to have guards. For example to declare a function one can write

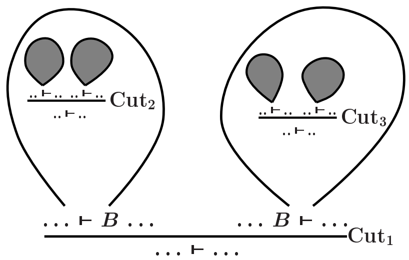
$$\begin{array}{ll}\texttt{function} & \mathit{pattern}_1\ \texttt{when}\ \mathit{cond}_1\texttt{->}\ \mathit{expr}_1\\ \quad\quad\quad\;\mid & \ldots\\ \quad\quad\quad\;\mid & \mathit{pattern}_n\ \texttt{when}\ \mathit{cond}_n\texttt{->}\ \mathit{expr}_n\end{array}$$

where for a pattern case, say expr_i, to be selected the pattern expression $\mathit{pattern}_i$ must match and the guard expression cond_i must evaluate to `true`. This is extremely helpful for structuring our implementation.

4.1 Implementation

In this section we shall give details of an implementation, which computes all normal forms reachable from a proof by $\xrightarrow{aux}$. Consider first the simply-typed lambda calculus where every term has only a single normal form. If we wish to compute this normal form, it is sufficient to implement a simple tail-recursive evaluation function. This can be done in a few lines of code. Strong normalisation and confluence of beta-reduction are the fundamental reasons why the code of this evaluation function is so simple. Confluence ensures that no matter which redex we choose we will find only one normal form, and the evaluation function will terminate because of the property of strong normalisation.

Our cut-elimination procedures are strongly normalising, but *not* confluent. Depending on which redex and which reduction rule one chooses, one may receive different normal forms. Thus an implementation which computes *all* normal forms reachable by $\xrightarrow{cut}$, for example, is non-trivial. Suppose in the proof



the two upper cuts (those with suffixes 2 and 3) are instances of Lafont's example (see Page 4). Each of them may reduce to one of its subproofs. Furthermore, assume the lower cut is a commuting cut, which may reduce to the left or to the right. Let the cut-formula, B, of this cut be introduced on both sides somewhere in one of the subproofs of the corresponding upper cut, then which normal form we receive depends on the reduction we apply for the two upper cuts. In one case the lower cut will eventually become a logical cut, and in another it will be eliminated in a single step because the subproofs where B was introduced vanished.

Similarly the normal forms we receive by reducing the upper cuts depend on how we reduce the lower cut. For example, if the lower cut is reduced to the left creating several copies of its right subproof, then we receive different normal forms depending on whether we reduce the upper cut (Suffix 3) before reducing the lower cut or after.

Since we do not yet know which choices lead to different normal forms, a naïve implementation would have to explore, or backtrack over, all possible choices. This could be done using techniques from logic programming, for example the proof search algorithms based on continuation passing (see for instance [Carlsson, 1984, Urban, 1998]). These techniques offer effective implementations for our cut-elimination procedures, but are substantially more complicated than the simple evaluation function of the simply-typed lambda calculus.

In the rest of this section we shall develop a simple implementation for the reduction system $(\mathcal{T}, \xrightarrow{aux})$. Using terminology from logic programming, we will achieve this by transforming the "don't know" non-determinism of $\xrightarrow{aux}$ into "don't care" non-determinism. In the end, somewhat surprisingly, we can implement a completely deterministic evaluation function, which, given a proof, computes all normal forms reachable by $\xrightarrow{aux}$ without the need of backtracking.

First, we observe that a leftmost-outermost reduction strategy is sufficient to compute all normal forms. This reduction strategy restricts the contexts in which an $\xrightarrow{aux}$-reduction may be performed such that not more than a single redex needs to be considered: there is either none (in the case of a normal form), or only one leftmost-outermost redex. Some clauses of the leftmost-outermost reduction strategy, denoted by $\xrightarrow{\overline{aux}}$, are given in Figure 4.1. To show that all normal forms reachable by $\xrightarrow{aux}$ are in fact reachable by the strategy $\xrightarrow{\overline{aux}}$, we shall first fix some terminology and prove a lemma showing that certain reductions can be rearranged in a reduction sequence.

Terminology 4.1.1: A reduction is said to be *bad*, denoted by $\xrightarrow{bad}$, if and only if it is of the form $\xrightarrow{aux}$, but not $\xrightarrow{\overline{aux}}$. This means the reduction violates the restrictions imposed on $\xrightarrow{\overline{aux}}$.

Lemma 4.1.2: For every reduction sequence of the form $M_1 \xrightarrow{bad} M_2 \xrightarrow{\overline{aux}} M_3$, there is a reduction sequence of the form $M_1 \xrightarrow{\overline{aux}} M' \xrightarrow{aux}{}^{*} M_3$.

Proof. By induction on the structure of M_1 (details on Page 167). □

Now we can prove that every normal form that can be reached by $\xrightarrow{aux}$ can also be reached by the strategy $\xrightarrow{\overline{aux}}$.

Theorem 4.1.3: For all $M, N \in \mathfrak{T}$ with N being a normal form,

$$\text{if } M \xrightarrow{aux}{}^{*} N \text{, then there is a reduction } M \xrightarrow{\overline{aux}}{}^{*} N.$$

Proof. By induction on the length of the reduction sequence of $\xrightarrow{aux}$, which by Theorem 2.3.19 is always finite (details on Page 167). □

We have shown that the leftmost-outermost reduction strategy, $\xrightarrow{\overline{aux}}$, does not restrict the collection of normal forms reachable from a classical proof. However, notice that, even though $\xrightarrow{\overline{aux}}$ is a strategy, it is non-deterministic. We can identify three types of terms as source for non-determinism.

(i) There are two cut-reductions that apply to $\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)\mathsf{Ax}(y,b))$. One yields $\mathsf{Ax}(x,a)[a\mapsto b]$ and the other $\mathsf{Ax}(y,b)[y\mapsto x]$.

(ii) The cut-reduction for commuting cuts may reduce the term $\mathsf{Cut}(\langle a\rangle M,(x)N)$ to either $M\{a := (x)N\}$ or $N\{x := \langle a\rangle M\}$ provided M and N do not freshly introduce a and x, respectively.

(iii) For logical cuts with a cut-formula of the form $B\supset C$ we have included two reductions (see Figure 2.5). Thus $\mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle M,b),(z)\mathsf{Imp}_L(\langle c\rangle N,(y)P,z))$ reduces to either

$$\mathsf{Cut}(\langle a\rangle\mathsf{Cut}(\langle c\rangle N,(x)M),(y)P) \quad \text{or} \quad \mathsf{Cut}(\langle c\rangle N,(x)\mathsf{Cut}(\langle a\rangle M,(y)P))\ .$$

$$\frac{M \xrightarrow{aux} N \quad \text{on the outermost level}}{M \xrightarrow{\overline{aux}} N}$$

$$\frac{M \xrightarrow{\overline{aux}} M'}{\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c) \xrightarrow{\overline{aux}} \mathsf{And}_R(\langle a\rangle M', \langle b\rangle N, c)} \qquad \frac{M \text{ is normal} \qquad N \xrightarrow{\overline{aux}} N'}{\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c) \xrightarrow{\overline{aux}} \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N', c)}$$

$$\frac{M \xrightarrow{\overline{aux}} M'}{\mathsf{And}^i_L((x)M, y) \xrightarrow{\overline{aux}} \mathsf{And}^i_L((x)M', y)}$$

Figure 4.1: Excerpt from the reduction system for the leftmost-outermost reduction strategy of $\xrightarrow{aux}$.

In the first case it is not difficult to see that the choice does not matter: both rules yield the normal form $\mathsf{Ax}(x, b)$. In the second we receive, in general, different normal forms depending on which choice is made. So both choices need to be explored in order to compute all normal forms reachable from a proof. Also in the third case the choice matters with respect to the reachable normal forms, as illustrated in Figure 4.2. If we do not explore both choices, we may "lose" some normal forms.

To avoid backtracking over the choices in Cases (ii) and (iii) we add the new term constructor

$$\mathsf{Mix}(M, N)$$

to the grammar of terms and, correspondingly, add the rule

$$\frac{\Gamma_1 \triangleright M \triangleright \Delta_1 \quad \Gamma_2 \triangleright N \triangleright \Delta_2}{\Gamma_1, \Gamma_2 \triangleright \mathsf{Mix}(M, N) \triangleright \Delta_1, \Delta_2}\ \text{Mix}$$

to the set of rules for forming typing judgements. We shall denote the corresponding set of well-typed terms by $\mathcal{T}^m$. There are no reduction rules that apply to Mix, but using this new term constructor we can reformulate the reductions that apply in Cases (ii) and (iii). The new reductions are listed below.

$\mathsf{Cut}(\langle a\rangle M, (x)N)$

$\xrightarrow{c'''}$ $M\{a := (x)N\}$ if N freshly introduces x, but M does not a, or

$\xrightarrow{c'''}$ $N\{x := \langle a\rangle M\}$ if M freshly introduces a, but N does not x, or

$\xrightarrow{c'''}$ $\mathsf{Mix}(M\{a := (x)N\}\,,\, N\{x := \langle a\rangle M\})$

if M and N do not, respectively, freshly introduce a and x

$\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle N, (y)P, z))$

$\xrightarrow{l''}$ $\mathsf{Mix}(\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle N, (x)M), (y)P)\,,\, \mathsf{Cut}(\langle c\rangle N, (x)\mathsf{Cut}(\langle a\rangle M, (y)P)))$

if $\mathsf{Imp}_R((x)\langle a\rangle M, b)$ freshly introduces b and
if $\mathsf{Imp}_L(\langle c\rangle N, (y)P, z)$ freshly introduces z

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash B}}{A\wedge A, B \vdash B}\,\wedge_{L_2}}{A\wedge A \vdash B\supset B}\,\supset_R \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}}{B\wedge B \vdash B}\,\wedge_{L_1} \quad \dfrac{\dfrac{}{B \vdash B}}{B \vdash B\vee B}\,\vee_{R_1}}{B\supset B, B\wedge B \vdash B\vee B}\,\supset_L}{A\wedge A, B\wedge B \vdash B\vee B}\,\text{Cut}$$

$$\swarrow \qquad\qquad\qquad\qquad \searrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}}{B\wedge B \vdash B}\,\wedge_{L_1} \quad \dfrac{\dfrac{}{A, B \vdash B}}{A\wedge A, B \vdash B}\,\wedge_{L_2}}{A\wedge A, B\wedge B \vdash B}\,\text{Cut} \quad \dfrac{\dfrac{}{B \vdash B}}{B \vdash B\vee B}\,\vee_{R_1}}{A\wedge A, B\wedge B \vdash B\vee B}\,\text{Cut} \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}}{B\wedge B \vdash B}\,\wedge_{L_1} \quad \dfrac{\dfrac{\dfrac{}{A, B \vdash B}}{A\wedge A, B \vdash B}\,\wedge_{L_2} \quad \dfrac{\dfrac{}{B \vdash B}}{B \vdash B\vee B}\,\vee_{R_1}}{B, A\wedge A \vdash B\vee B}\,\text{Cut}}{A\wedge A, B\wedge B \vdash B\vee B}\,\text{Cut}$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash B}}{A\wedge A, B \vdash B}\,\wedge_{L_2}}{A\wedge A, B\wedge B \vdash B}\,\wedge_{L_1}}{A\wedge A, B\wedge B \vdash B\vee B}\,\vee_{R_1} \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash B}}{A\wedge A, B \vdash B}\,\wedge_{L_2}}{A\wedge A, B \vdash B\vee B}\,\vee_{R_1}}{A\wedge A, B\wedge B \vdash B\vee B}\,\wedge_{L_1}$$

Figure 4.2: The cut in the upper proof is replaced by two cuts on smaller formulae. There are two possibilites of how to nest the resulting cuts, as is shown in the second line. From these proofs the normal forms given below can be reached. The normal form on the left, however, cannot be reached from the proof in the second line on the right, and the normal form on the right not from the proof in the second line on the left. Thus the choice of how to nest the cuts when applying a logical cut $\supset_R/\supset_L$ matters with respect to the normal forms that can be reached.

It is not hard to see that the reformulation does not affect the behaviour of cut-elimination with respect to strong normalisation. The corresponding new strategy is denoted by $\xrightarrow{\overline{aux}}{}'$; the new reduction system is denoted by $(\mathcal{T}^m, \xrightarrow{\overline{aux}}{}')$. To ensure that we do not have to explore any choices in $(\mathcal{T}^m, \xrightarrow{\overline{aux}}{}')$, we show that $\xrightarrow{\overline{aux}}{}'$ is confluent.

Proposition 4.1.4: The reduction $\xrightarrow{\overline{aux}}{}'$ is confluent.

Proof. By inspection of the reduction rules: the only critical pair is of the form

$$\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(x,a), (y)\mathsf{Ax}(y,b))$$

which in both cases reduces to $\mathsf{Ax}(x,b)$. □

Consequently, it does not matter which $\xrightarrow{\overline{aux}}{}'$-reduction we apply: we receive only a single normal form, which of course in general contains instances of Mix. This normal form can be seen as a compact encoding of the collection of all normal forms reachable by $\xrightarrow{\overline{aux}}$, and thus also by $\xrightarrow{aux}$ and $\xrightarrow{cut}$. In order to distinguish them from the normal forms reachable, for example, in $(\mathcal{T}, \xrightarrow{\overline{aux}})$, we shall henceforth refer to the normal forms with instances of Mix as *mix-normal forms*. An "unwinding" function will allow us to extract the collection of "real" normal forms from a mix-normal form.

Notice that the "trick" with the mix term constructor cannot be used in $(\mathcal{T}, \xrightarrow{aux})$: if we replace $\xrightarrow{c'}$ with $\xrightarrow{c'''}$ in $(\mathcal{T}, \xrightarrow{aux})$, we do *not* obtain a confluent reduction system, as demonstrated in the example below.

Example 4.1.5: Suppose M, N and P are cut-free terms and assume: (i) $a \notin FC(M)$, (ii) M freshly introduces b, (iii) $y \notin FN(P)$ and (iv) N freshly introduces x. Consider then the following term

$$\mathsf{Cut}(\langle b\rangle \mathsf{Cut}(\langle a\rangle M, (x)N), (y)P)\ .$$

Reducing first the outer cut, we receive the following reduction sequence.

$$\begin{array}{lll}
 & \mathsf{Cut}(\langle b\rangle \mathsf{Cut}(\langle a\rangle M, (x)N), (y)P) & \\
\xrightarrow{aux} & \mathsf{Mix}(\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle b\rangle M, (y)P), (x)N)\ ,\ P) & \text{by (ii) and (iii)} \\
\xrightarrow{aux} & \mathsf{Mix}(\mathsf{Cut}(\langle a\rangle P, (x)N)\ ,\ P) & \text{by (ii) and (iii)} \\
\xrightarrow{aux} & \mathsf{Mix}(P\ ,\ P) & \text{by (i) and (iv)}
\end{array}$$

Reducing the inner cut first, we receive a slightly different reduction sequence.

$$\begin{array}{lll}
 & \mathsf{Cut}(\langle b\rangle \mathsf{Cut}(\langle a\rangle M, (x)N), (y)P) & \\
\xrightarrow{aux} & \mathsf{Cut}(\langle b\rangle M, (y)P) & \text{by (i) and (iv)} \\
\xrightarrow{aux} & P & \text{by (ii) and (iii)}
\end{array}$$

So the restrictions imposed by the strategy are necessary for confluence.

4.1.1 The Code

Let us now give some details of the code. Figure 4.3 shows the main datatypes implementing the following design decisions:

- names, co-names and variables are implemented as strings,
- in every cut we record explicitly the cut-formula (a typing algorithm for terms is then rather simple, but beyond the scope of this thesis), and
- there are four syntactic categories of terms: terms, named terms, co-named terms and double-bounded terms.

The double-bounded terms implement the special binding operation in terms of the form $\mathsf{Imp}_R((x)\langle a\rangle M, b)$. Some examples of how these datatypes relate to our formal definitions are given below.

Example	Code	
expression $\mathsf{f}(\mathsf{g}(\mathsf{x}))$	`F("f",[F("g",[Var("x")])])`	
formula $\forall \mathsf{x}.(A \wedge B\mathsf{x})$	`Forall("x",And(Atm("A"),Pre("B",[Var("x")])))`	
term $\mathsf{And}_L^i((x)M, y)$	`AndL(t,m,"y")`	where `t` $\in$ {`I`,`II`} and `m` is the code of $(x)M$
term $\mathsf{Imp}_R((x)\langle a\rangle M, b)$	`ImpR(m,"b")`	where `m` is the code of $(x)\langle a\rangle M$
named term $(x)M$	`("x",m)`	where `m` is the code of M
co-named term $\langle a\rangle M$	`("a",m)`	where `m` is the code of M
double-bounded term $(x)\langle a\rangle M$	`("x","a",m)`	where `m` is the code of M

Next we give the details for the cut-reductions and the evaluation function. They call six subsidiary functions, *viz.*

- `rename`, which handles the renaming of names and co-names,
- `substL` and `substR`, which perform the proof substitution $\{\_\}$,
- `is_logical_cut`, which determines whether a term is a logical cut, and
- `is_fresh_nm` and `is_fresh_co`, which test whether a name or co-name is freshly introduced.

```
type name     = string;;
type coname   = string;;
type variable = string;;

type tag      = I | II;;

type expr =
    Var  of variable
  | F    of string * expr list
;;

type form =
    Atm       of string
  | Pre       of string * expr list
  | And       of form * form
  | Or        of form * form
  | Imp       of form * form
  | Neg       of form
  | Forall    of variable * form
  | Exists    of variable * form
;;

type term =
    Ax       of name   *  coname
  | Cut      of form   *  cterm  *  nterm
  | NotL     of cterm  *  name
  | NotR     of nterm  *  coname
  | AndR     of cterm  *  cterm  *  coname
  | AndL     of tag    *  nterm  *  name
  | OrR      of tag    *  cterm  *  coname
  | OrL      of nterm  *  nterm  *  name
  | ImpR     of ncterm *  name
  | ImpL     of cterm  *  nterm  *  name
  | ForallL  of nterm  *  expr      *  name
  | ForallR  of cterm  *  variable  *  coname
  | ExistsL  of nterm  *  variable  *  name
  | ExistsR  of cterm  *  expr      *  coname
  | Mix      of term   * term
and nterm   = name   * term
and cterm   = coname * term
and ncterm  = name   * coname * term
;;
```

Figure 4.3: Code for main datatypes.

Here is a fragment of the code that implements the cut-reductions.

```
let rec aux_redu term =
  match term with

    Cut(f,(b,Ax(x,a)),(z,m))
        when (b=a & is_logical_cut term)
        -> rename m z x

  | Cut(Or(f1,f2),(c,OrR(t,(a,m),b)),(z,OrL((x1,n1),(x2,n2),y)))
        when (c=b & z=y & is_logical_cut term) ->
        if t=I then Cut(f1,(a,m),(x1,n1))
               else Cut(f2,(a,m),(x2,n2))

  | Cut(f,(a,m),(x,n))
        when (not (is_fresh_co m a)) & (is_fresh_nm n x)
        -> substL m (a,f) (x,n)

  | Cut(f,(a,m),(x,n))
        when (not (is_fresh_co m a)) & (not (is_fresh_nm n x))
        -> Mix(substL m (a,f) (x,n),substR n (x,f) (a,m))

  | ...
;;
```

The first pattern case is for logical cuts with axioms, the second for logical cuts $\vee_R/\vee_L$, the third is for commuting cuts where only the right subproof introduces the cut-formula freshly, and the last is for commuting cuts where both subproofs do not freshly introduce the cut-formula.

As mentioned earlier, the reason for restricting $(\mathcal{T}, \xrightarrow{aux})$ was to obtain a simple evaluation function that computes all normal forms. It is the following tail-recursive function.

```
let rec eval term =
  match term with

      Cut(f,m,n)           -> eval (aux_redu (Cut(f,m,n)))
    | Ax(x,a)              -> Ax(x,a)
    | NotL((a,m),x)        -> NotL((a,eval m),x)
    | AndR((a,m),(b,n),c)  -> AndR((a,eval m),(b,eval n),c)
    | ...
    | ImpR((x,a,m),b)      -> ImpR((x,a,eval m),b)
    | Mix(m,n)             -> Mix(eval m,eval n)
;;
```

This evaluation function generates for every term a mix-normal form. From this mix-normal form the collection of the "real" normal forms can be extracted using the unwinding function given below, which returns the normal forms as a list of terms. This function calls two subsidiary function: `pair_up_lists`, which returns a list that contains all possible combinations of forming pairs from two lists, and `map`, which applies a function to all elements of a given list.

```
let rec unwind term  =
  match term with
      Ax(x,a)    -> [Ax(x,a)]
    | NotL((a,m),x)
                 -> map (fun m -> (NotL((a,m),x))) (unwind m)
    | AndR((a,m),(b,n),c)
                 -> let plist = pair_up_lists (unwind m) (unwind n) in
                    map (fun (m,n) -> (AndR((a,m),(b,n),c))) plist
    | ...
    | ImpR((x,a,m),b)
                 -> map (fun m -> (ImpR((x,a,m),b))) (unwind m)
    | Mix(m,n)   -> (unwind m) @ (unwind n)
;;
```

The next function calculates the number of normal forms encoded in a mix-normal form without actually extracting the normal forms. This function is useful for analysing examples.

```
let rec count term  =
  match term with
      Ax(x,a)              -> 1
    | NotL((a,m),x)        -> (count m)
    | AndR((a,m),(b,n),c)  -> (count m) * (count n)
    | ...
    | ImpR((x,a,m),b)      -> (count m)
    | Mix(m,n)             -> (count m) + (count n)
;;
```

We should now like to examine an example. Consider the proof in which for brevity all names and co-names are omitted.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\mathsf{a} \vdash A\mathsf{a}} \quad \dfrac{}{A\mathsf{b} \vdash A\mathsf{b}}}{A\mathsf{a}\lor A\mathsf{b} \vdash A\mathsf{a}, A\mathsf{b}}{\scriptstyle\lor_L}}{\forall \mathsf{y}.(A\mathsf{y}\lor A\mathsf{b}) \vdash A\mathsf{a}, A\mathsf{b}}{\scriptstyle\forall_L}}{\forall \mathsf{x}.\mathsf{y}.(A\mathsf{y}\lor A\mathsf{x}) \vdash A\mathsf{a}, A\mathsf{b}}{\scriptstyle\forall_L}}{\forall \mathsf{x}.\mathsf{y}.(A\mathsf{y}\lor A\mathsf{x}) \vdash A\mathsf{b}, \forall \mathsf{x}.A\mathsf{x}}{\scriptstyle\forall_R}}{\forall \mathsf{x}.\mathsf{y}.(A\mathsf{y}\lor A\mathsf{x}) \vdash \forall \mathsf{x}.A\mathsf{x}, \forall \mathsf{y}.A\mathsf{y}}{\scriptstyle\forall_R} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{A\mathsf{c} \vdash A\mathsf{c}} \quad \dfrac{}{A\mathsf{d} \vdash A\mathsf{d}}}{A\mathsf{c}, A\mathsf{d} \vdash A\mathsf{c}\land A\mathsf{d}}{\scriptstyle\land_R}}{A\mathsf{d}, \forall \mathsf{x}.A\mathsf{x} \vdash A\mathsf{c}\land A\mathsf{d}}{\scriptstyle\forall_L}}{\forall \mathsf{x}.A\mathsf{x} \vdash A\mathsf{c}\land A\mathsf{d}}{\scriptstyle\forall_L^\star}}{\forall \mathsf{x}.\mathsf{y}.(A\mathsf{y}\lor A\mathsf{x}) \vdash \forall \mathsf{y}.A\mathsf{y}, A\mathsf{c}\land A\mathsf{d}}{\scriptstyle\mathrm{Cut}} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{A\mathsf{e} \vdash A\mathsf{e}} \quad \dfrac{}{A\mathsf{f} \vdash A\mathsf{f}}}{A\mathsf{e}, A\mathsf{f} \vdash A\mathsf{e}\land A\mathsf{f}}{\scriptstyle\land_R}}{A\mathsf{f}, \forall \mathsf{y}.A\mathsf{y} \vdash A\mathsf{e}\land A\mathsf{f}}{\scriptstyle\forall_L}}{\forall \mathsf{y}.A\mathsf{y} \vdash A\mathsf{e}\land A\mathsf{f}}{\scriptstyle\forall_L^\star}}{\forall \mathsf{x}.\mathsf{y}.(A\mathsf{y}\lor A\mathsf{x}) \vdash A\mathsf{c}\land A\mathsf{d}, A\mathsf{e}\land A\mathsf{f}}{\scriptstyle\mathrm{Cut}} \tag{4.1}$$

One of its normal forms is given in Figure 4.4. However let us analyse the collection of all normal forms reachable from this proof. As can be seen, the proof contains two cuts and two implicit contractions (in the inference rules marked with a star). If we feed the corresponding term into our evaluation function and count all normal forms that can be extracted from the corresponding mix-normal form, we receive the number 1,618,912! Although the normal forms are all moderately small, we are confronted with a serious problem: the collection of normal forms does not fit into the memory of an average computer.[1] If we want to analyse the normal forms with regard to their computational content, we have to alleviate this problem. There are two reasons for this "explosion" in the number of normal forms:

- First, in case of commuting cuts whose cut-formula is not freshly introduced on either side, the reduction $\xrightarrow{c'}$ leads, in general, to different normal forms, and thus the reduct of $\xrightarrow{c'''}$ includes an instance of Mix. In some cases, however, this reduction does not generate different normal forms. Consider, for example, the following proof.

$$\dfrac{\dfrac{\overline{A \vdash A, C} \quad \overline{C, A \vdash A}}{A \vdash A}\,\text{Cut} \quad \dfrac{\overline{B \vdash B, D} \quad \overline{D, B \vdash B}}{B \vdash B}\,\text{Cut}}{A, B \vdash A \wedge B}\,\wedge_R$$

  Our evaluation function calculates a mix-normal form for this proof from which four normal forms can be extracted. But all of them stand for the proof

$$\dfrac{\overline{A \vdash A} \quad \overline{B \vdash B}}{A, B \vdash A \wedge B}\,\wedge_R\ .$$

  We could alleviate our problem by letting $\xrightarrow{c'''}$ only reduce to $\mathsf{Mix}(M, N)$ provided $M \not\equiv N$. However, in the examples we analysed this did not reduce substantially the number of normal forms.

- The second reason is more subtle. To explain the issues we give the reader an example. Consider the proof

$$\dfrac{\dfrac{\overline{A \vdash A}}{A \wedge B \vdash A}\,\wedge_L^1 \quad \dfrac{\overline{A \vdash A}}{A \vdash A \vee C}\,\vee_R^1}{A \wedge B \vdash A \vee C}\,\text{Cut}$$

  and let us calculate the normal forms reachable from this proof—we receive the following two proofs

$$\dfrac{\dfrac{\overline{A \vdash A}}{A \vdash A \vee C}\,\vee_R^1}{A \wedge B \vdash A \vee C}\,\wedge_L^1 \quad \text{and} \quad \dfrac{\dfrac{\overline{A \vdash A}}{A \wedge B \vdash A}\,\wedge_L^1}{A \wedge B \vdash A \vee C}\,\vee_R^1$$

  which differ only in the order of their inferences and can be identified by simple permutation rules introduced by Kleene [1952b]. In what follows, we shall restrict the reduction $\xrightarrow{c'''}$ such that in the examples of the kind above only one normal

[1] If we assume that every normal form occupies 100 Bytes of memory, which is an optimistic estimation for the smallest of these normal forms, we need a computer with more than 150 MByte of RAM.

$$\underbrace{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Ac \vdash Ac} \quad \dfrac{}{Ae \vdash Ae}}{Ac\vee Ae \vdash Ac, Ae}\vee_L}{\forall y.(Ay\vee Ae)\vdash Ac, Ae}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ac, Ae}\forall_L \quad \dfrac{\dfrac{\dfrac{\dfrac{}{Ad \vdash Ad} \quad \dfrac{}{Ae \vdash Ae}}{Ad\vee Ae \vdash Ad, Ae}\vee_L}{\forall y.(Ay\vee Ae)\vdash Ad, Ae}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ad, Ae}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ae, Ae, Ac\wedge Ad}\wedge_R \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Ac \vdash Ac} \quad \dfrac{}{Af \vdash Af}}{Ac\vee Af \vdash Ac, Af}\vee_L}{\forall y.(Ay\vee Af)\vdash Ac, Af}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ac, Af}\forall_L \quad \dfrac{\dfrac{\dfrac{\dfrac{}{Ad \vdash Ad} \quad \dfrac{}{Af \vdash Af}}{Ad\vee Af \vdash Ad, Af}\vee_L}{\forall y.(Ay\vee Af)\vdash Ad, Af}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ad, Af}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Af, Af, Ac\wedge Ad}\wedge_R}{\forall x.y.(Ay\vee Ax)\vdash Ae, Af, Ac\wedge Ad, Ae\wedge Af}\wedge_R}_{X}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Ac \vdash Ac} \quad \dfrac{}{Ae \vdash Ae}}{Ac\vee Ae \vdash Ac, Ae}\vee_L}{\forall y.(Ay\vee Ae)\vdash Ac, Ae}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ac, Ae}\forall_L \quad \dfrac{\dfrac{\dfrac{\dfrac{}{Ad \vdash Ad} \quad \dfrac{}{Ae \vdash Ae}}{Ad\vee Ae \vdash Ad, Ae}\vee_L}{\forall y.(Ay\vee Ae)\vdash Ad, Ae}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ad, Ae}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ae, Ac\wedge Ad}\wedge_R \quad \dfrac{\begin{matrix}\vdots\\ X\end{matrix} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Ac \vdash Ac} \quad \dfrac{}{Af \vdash Af}}{Ac\vee Af \vdash Ac, Af}\vee_L}{\forall y.(Ay\vee Af)\vdash Ac, Af}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ac, Af}\forall_L \quad \dfrac{\dfrac{\dfrac{\dfrac{}{Ad \vdash Ad} \quad \dfrac{}{Af \vdash Af}}{Ad\vee Af \vdash Ad, Af}\vee_L}{\forall y.(Ay\vee Af)\vdash Ad, Af}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Ad, Af}\forall_L}{\forall x.y.(Ay\vee Ax)\vdash Af, Af, Ac\wedge Ad}\wedge_R}{\forall x.y.(Ay\vee Ax)\vdash Af, Ac\wedge Ad, Ae\wedge Af}\wedge_R}{\forall x.y.(Ay\vee Ax)\vdash Ac\wedge Ad, Ae\wedge Af}\wedge_R$$

Figure 4.4: A normal form of Proof (4.1) shown on Page 110.

form is calculated. We thus restrict the collection of normal forms reachable from a classical proof, but it is commonly accepted that this restriction does not affect the computational content of classical proofs. The restriction allows us to consider larger examples and study their computational content. (A similar remark applies to the mix-instance introduced when reducing a logical cut $\supset_L/\supset_R$. The normal forms that are generated by exploring both ways of nesting the cuts in the reduct differ only in the order of their inference rules.)

In our implementation we thus modify the code for commuting cuts whose cut-formula is not freshly introduced on either side. If in both subterms the (co-)name of the cut-formula occurs freely just once, then it is not hard to show that the normal forms we receive by reducing the cut exploring both choices can be identified using simple permutation rules. Therefore we do not explore both choices; instead we reduce the cut by a single substitution. The modified code is as follows.

```
let rec aux_redu term =
  match term with
    ...
  | Cut(f,(a,m),(x,n))
        when (not (is_fresh_co m a)) & (not (is_fresh_nm n x))
        -> if (free_occ_co a m = 1) & (free_occ_na x n = 1)
           then substR n (x,f) (a,m)
           else Mix(substL m (a,f) (x,n),substR n (x,f) (a,m))
  | ...
;;
```

This function calls `free_occ_na` and `free_occ_co`, which calculate the number of free occurrences of a name and co-name, respectively.

Returning to our example proof given in (4.1) we find that the modified implementation returns 10 normal forms, one of which was shown in Figure 4.4. Our implementation is now streamlined enough to be a useful tool for analysing the computational content of bigger classical proofs.

4.2 Non-Deterministic Computation: Case Study

Hitherto we were mainly concerned with the proof theory of classical logic. In this section and the next we shall study how cut-elimination relates to computation. We started in the introduction arguing that the process of cut-elimination in classical logic should be seen as non-deterministic computation. We motivated this viewpoint with the problems arising from Lafont's example (see Page 4). However there is an obvious question whether we can support this viewpoint with convincing examples. First, recall that via the Curry-Howard correspondence reduction can be seen as a form of computation and a normal form as a result of a computation. Now, consider the (intuitionistic) proof

$$\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A \lor A \vdash A}\lor_L \quad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A \vdash A \land A}\land_R}{A \lor A \vdash A \land A}\text{Cut}$$

which, using our cut-elimination procedures, can be reduced to one of the following two normal forms

$$\dfrac{\dfrac{\overline{A\vdash A}\quad\overline{A\vdash A}}{A\vee A\vdash A}\vee_L\qquad\dfrac{\overline{A\vdash A}\quad\overline{A\vdash A}}{A\vee A\vdash A}\vee_L}{A\vee A\vdash A\wedge A}\wedge_R\qquad\dfrac{\dfrac{\overline{A\vdash A}\quad\overline{A\vdash A}}{A\vdash A\wedge A}\wedge_R\qquad\dfrac{\overline{A\vdash A}\quad\overline{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\vee A\vdash A\wedge A}\vee_L \tag{4.2}$$

The main feature of our cut-elimination procedures is that one can non-deterministically choose to which normal form the proof can be reduced. Consequently, one could argue that via the Curry-Howard correspondence the cut-elimination process in intuitionistic logic corresponds to non-deterministic computation, since we receive two different normal forms (results). However, this argument as it stands is not very strong: one receives several normal forms starting from the proof above, but if we interpret these normal forms as terms in the simply-typed lambda calculus, we find they all correspond to the same value and computation (see Example 3.1.3). This means we cannot decide whether cut-elimination corresponds to non-deterministic computation just by looking at features such as size and shape of the normal forms.

So in this section we shall present a classical proof and use our implementation to compute two of its normal forms. We shall show that the two normal forms differ in "essential" features; trying to identify them is doomed to triviality (the notion of proof would coincide with the notion of provability). The classical proof which we present is adapted from work presented by Barbanera et al. [1997], but originates from work by Stolzenberg [Coquand, 1995, Herbelin, 1995].

Suppose we are given an infinite tape where each cell contains either a '0' or a '1', for example

1	0	0	1	...

For the sake of notational simplicity, we shall in what follows represent a tape as a function, denoted by f, from integers to booleans. Thus we know about the tape that

$$\forall \mathsf{x}.(\mathsf{fx} = 0 \vee \mathsf{fx} = 1)\ . \tag{4.3}$$

This is a first-order formula in which $=$ is a predicate standing for 'equality' between two booleans. As is common, we write this predicate symbol in infix notation. Using (4.3) as assumption, we are going to prove the proposition

$$\exists \mathsf{n}.\mathsf{m}.(\mathsf{n} < \mathsf{m} \wedge \mathsf{fn} = \mathsf{fm}) \tag{4.4}$$

where $<$ is a predicate, written again in infix notation, which stands for 'less than' between two integers. Later, we shall also use $\leq$, which stands for 'less than or equal to'. The proposition just given may be summarised in English as on every tape there are two cells that have the same content.

In order to prove the proposition, we need some principles from arithmetic. For example the principle of transitivity of equality, i.e., the implication

$$\text{if } \mathsf{fn} = \mathsf{i} \text{ and } \mathsf{fm} = \mathsf{i}, \text{then } \mathsf{fn} = \mathsf{fm}.$$

We could add this principle to classical logic as the non-logical axiom

$$\overline{\Gamma \vdash \Delta, (\mathsf{fy} = \mathsf{i} \wedge \mathsf{fx} = \mathsf{i}) \supset \mathsf{fx} = \mathsf{fy}}$$

or as the inference rule

$$\frac{\Gamma \vdash \Delta, \mathsf{fy} = \mathsf{i} \quad \Gamma \vdash \Delta, \mathsf{fx} = \mathsf{i}}{\Gamma \vdash \Delta, \mathsf{fx} = \mathsf{fy}}\,\text{Trans} .$$

However we shall refrain from adding either of them, because it would require a fresh cut-elimination proof (cut-elimination in the presence of non-logical axioms or additional inference rules is non-trivial). Instead, we shall represent principles of arithmetic as formulae, which we add to our assumptions. Transitivity, for example, is represented as the formula

$$\forall \mathsf{i.x.y.}((\mathsf{fy} = \mathsf{i} \wedge \mathsf{fx} = \mathsf{i}) \supset \mathsf{fx} = \mathsf{fy}) \ . \tag{4.5}$$

Other principles we need concern the predicates $<$ and $\leq$. The corresponding formulae are

$$\forall \mathsf{x.y.}(\mathsf{sx} \leq \mathsf{y} \supset \mathsf{x} < \mathsf{y}) \tag{4.6}$$
$$\forall \mathsf{y.x.}(\mathsf{x} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}}) \tag{4.7}$$
$$\forall \mathsf{y.x.}(\mathsf{y} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}}) \tag{4.8}$$

in which s denotes the successor function and $\mathsf{m}^{\mathsf{x},\mathsf{y}}$ denotes the maximum function of two integers, x and y. For instance $\mathsf{m}^{\mathsf{s0},\mathsf{0}}$ is $\mathsf{s0}$. In the sequel, we shall abbreviate $\mathsf{s0}$, $\mathsf{ss0}$, $\mathsf{sss0}$ as 1, 2 and 3, respectively. A proof of the statement $0<1$ is then as follows.

$$\pi \left\{ \frac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{1\leq 1 \vdash 1\leq 1}}{\forall \mathsf{x.}(\mathsf{x}\leq \mathsf{m}^{\mathsf{x},0}) \vdash 1\leq 1}\forall_L}{\forall \mathsf{y.x.}(\mathsf{x} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}}) \vdash 1\leq 1}\forall_L \quad \overline{0<1 \vdash 0<1}}{1\leq 1 \supset 0<1, \forall \mathsf{y.x.}(\mathsf{x} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}}) \vdash 0<1}\supset_L}{\forall \mathsf{y.}(1\leq \mathsf{y} \supset 0<\mathsf{y}), \forall \mathsf{y.x.}(\mathsf{x} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}}) \vdash 0<1}\forall_L}{\forall \mathsf{x.y.}(\mathsf{sx} \leq \mathsf{y} \supset \mathsf{x} < \mathsf{y}), \forall \mathsf{y.x.}(\mathsf{x} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}}) \vdash 0<1}\forall_L \right.$$

There are several ways of how to formalise a proof of $0 < 1$; it will, however, become clear later on why we have chosen this particular formalisation. Before we proceed, let us introduce some abbreviations for the formulae given in (4.3) to (4.8).

$A \stackrel{\text{def}}{=}$	$\forall \mathsf{x.}(\mathsf{fx} = 0 \vee \mathsf{fx} = 1)$	assumption	(4.3)
$P \stackrel{\text{def}}{=}$	$\exists \mathsf{n.m.}(\mathsf{n} < \mathsf{m} \wedge \mathsf{fn} = \mathsf{fm})$	proposition	(4.4)
$T \stackrel{\text{def}}{=}$	$\forall \mathsf{i.x.y.}((\mathsf{fy} = \mathsf{i} \wedge \mathsf{fx} = \mathsf{i}) \supset \mathsf{fx} = \mathsf{fy})$	transitivity	(4.5)
$S \stackrel{\text{def}}{=}$	$\forall \mathsf{x.y.}(\mathsf{sx} \leq \mathsf{y} \supset \mathsf{x} < \mathsf{y})$	successor	(4.6)
$M_1 \stackrel{\text{def}}{=}$	$\forall \mathsf{y.x.}(\mathsf{x} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}})$	maximum$_1$	(4.7)
$M_2 \stackrel{\text{def}}{=}$	$\forall \mathsf{y.x.}(\mathsf{y} \leq \mathsf{m}^{\mathsf{x},\mathsf{y}})$	maximum$_2$	(4.8)

The sequent that we are going to prove is

$$\begin{array}{l} \forall \mathsf{y.x.(x \leq m^{x,y})},\ \forall \mathsf{y.x.(y \leq m^{x,y})},\ \forall \mathsf{x.y.(sx \leq y \supset x < y)}, \\ \forall \mathsf{i.x.y.((fy = i \wedge fx = i) \supset fx = fy)}, \\ \forall \mathsf{x.(fx = 0 \vee fx = 1)} \end{array} \quad \vdash \exists \mathsf{n.m.(n < m \wedge fn = fm)}$$

or using our abbreviations

$$M_1, M_2, S, T, A \vdash P\ . \tag{4.9}$$

We turn now to the question of how one might prove the sequent just given. As one may expect, there are several possibilities. We first consider proofs that “perform” a case analysis; by this we mean they consist of a number of tests, which check whether two cells on a given tape have the same content. Take for example the following tape.

1	0	0	1	...

Here the test $\mathsf{f1} = \mathsf{f2} = 0$, or $\mathsf{f0} = \mathsf{f3} = 1$, is sufficient to show P for this particular tape. However, both tests fail for the tape

1	1	0	0	...

Since we want to show P for all tapes, we have to do some more tests. It is left to the reader to verify that the following tests

$$\begin{array}{ll} \mathsf{f0} = \mathsf{f1} = 0 & \mathsf{f0} = \mathsf{f1} = 1 \\ \mathsf{f0} = \mathsf{f2} = 0 & \mathsf{f0} = \mathsf{f2} = 1 \\ \mathsf{f1} = \mathsf{f2} = 0 & \mathsf{f1} = \mathsf{f2} = 1 \end{array} \tag{4.10}$$

guarantee that there is an n and m such that $\mathsf{n} < \mathsf{m}$ and $\mathsf{fn} = \mathsf{fm}$ for *every* tape. Obviously the collection of tests given above is not the only one to establish the proposition for all tapes. We could, for example, test the 13th, 101th and 999th cell, instead of the first three. Also, we might consider, as we shall see later, more than just three cells.

Let us show how a test translates into a sequent proof. Below we give a proof, which corresponds to the test $\mathsf{f0} = \mathsf{f1} = 0$.

$$\begin{prooftree}
\AxiomC{π}
\noLine
\UnaryInfC{$M_1, S \vdash \mathsf{0{<}1}$}
\AxiomC{}
\UnaryInfC{$\mathsf{f1{=}0} \vdash \mathsf{f1{=}0}$}
\AxiomC{}
\UnaryInfC{$\mathsf{f1{=}1} \vdash \mathsf{f1{=}1}$}
\RightLabel{$\vee^{\circ}_L$}
\BinaryInfC{$\mathsf{f1{=}0{\vee}f1{=}1} \vdash \mathsf{f1{=}0, f1{=}1}$}
\RightLabel{$\forall^{\circ}_L$}
\UnaryInfC{$\forall\mathsf{x.(fx{=}0{\vee}fx{=}1)} \vdash \mathsf{f1{=}0, f1{=}1}$}
\AxiomC{}
\UnaryInfC{$\mathsf{f0{=}0} \vdash \mathsf{f0{=}0}$}
\AxiomC{}
\UnaryInfC{$\mathsf{f0{=}1} \vdash \mathsf{f0{=}1}$}
\RightLabel{$\vee^{\circ}_L$}
\BinaryInfC{$\mathsf{f0{=}0{\vee}f0{=}1} \vdash \mathsf{f0{=}0, f0{=}1}$}
\RightLabel{$\forall^{\circ}_L$}
\UnaryInfC{$\forall\mathsf{x.(fx{=}0{\vee}fx{=}1)} \vdash \mathsf{f0{=}0, f0{=}1}$}
\RightLabel{$\wedge^{\bullet}_R$}
\BinaryInfC{$\forall\mathsf{x.(fx{=}0{\vee}fx{=}1)} \vdash \mathsf{f1{=}0{\wedge}f0{=}0, f1{=}1, f0{=}1}$}
\AxiomC{}
\UnaryInfC{$\mathsf{f0{=}f1} \vdash \mathsf{f0{=}f1}$}
\RightLabel{$\supset^{\bullet}_L$}
\BinaryInfC{$\mathsf{(f1{=}0{\wedge}f0{=}0){\supset}f0{=}f1}, \forall\mathsf{x.(fx{=}0{\vee}fx{=}1)} \vdash \mathsf{f0{=}f1, f1{=}1, f0{=}1}$}
\RightLabel{$\forall^{\bullet}_L$}
\UnaryInfC{$\forall\mathsf{y.((fy{=}0{\wedge}f0{=}0){\supset}f0{=}fy)}, \forall\mathsf{x.(fx{=}0{\vee}fx{=}1)} \vdash \mathsf{f0{=}f1, f1{=}1, f0{=}1}$}
\RightLabel{$\forall^{\bullet}_L$}
\UnaryInfC{$\forall\mathsf{x.y.((fy{=}0{\wedge}fx{=}0){\supset}fx{=}fy)}, \forall\mathsf{x.(fx{=}0{\vee}fx{=}1)} \vdash \mathsf{f0{=}f1, f1{=}1, f0{=}1}$}
\RightLabel{$\forall^{\bullet}_L$}
\UnaryInfC{$T, \forall\mathsf{x.(fx{=}0{\vee}fx{=}1)} \vdash \mathsf{f0{=}f1, f1{=}1, f0{=}1}$}
\RightLabel{$\wedge^{\star}_R$}
\BinaryInfC{$M_1, S, T, \forall\mathsf{x.(fx{=}0 \vee fx{=}1)} \vdash \mathsf{0{<}1 \wedge f0{=}f1,\ f1{=}1,\ f0{=}1}$}
\RightLabel{$\exists^{\star}_R$}
\UnaryInfC{$M_1, S, T, \forall\mathsf{x.(fx{=}0 \vee fx{=}1)} \vdash \exists\mathsf{m.(0{<}m \wedge f0{=}fm),\ f1{=}1,\ f0{=}1}$}
\RightLabel{$\exists^{\star}_R$}
\UnaryInfC{$M_1, S, T, \underbrace{\forall\mathsf{x.(fx{=}0 \vee fx{=}1)}}_{A} \vdash \underbrace{\exists\mathsf{n.m.(n{<}m \wedge fn{=}fm)}}_{P},\ \mathsf{f1{=}1,\ f0{=}1}$}
\end{prooftree}$$

In this proof, the inference rules marked with a star are concerned with the proposition, P, the inference rules marked with a circle are concerned with the assumption, A, and the inference rules marked with a disk concern the formula for transitivity, T. Clearly, the order of the inference rules is not important with respect to which test is made: the inference rules can be easily rearranged. However it is relatively simple to determine in any proof of this kind which test is made: one only has to find the place where the proof deals with the transitivity (shaded formula). So in the proof given above, the inference rule

$$\dfrac{\ldots \vdash \mathsf{f1} = 0 \wedge \mathsf{f0} = 0, \ldots \qquad \mathsf{f1} = \mathsf{f0} \vdash \ldots}{(\mathsf{f1} = 0 \wedge \mathsf{f0} = 0) \supset \mathsf{f0} = \mathsf{f1}, \ldots \vdash \ldots} \supset_L$$

is an indicator for the test $\mathsf{f0} = \mathsf{f1} = 0$ being "performed". Note that the two formulae $\mathsf{f1} = 1$ and $\mathsf{f0} = 0$, which appear in the endsequent, are *not* part of the test $\mathsf{f0} = \mathsf{f1} = 0$, but have to be used in other tests. Two proofs for the sequent $M_1, M_2, S, T, A \vdash P$ are given in Appendix A. Each of them corresponds to a particular collection of tests, as indicated below.

Proof on Pages 132–134		Proof on Pages 135–140	
$\mathsf{f0} = \mathsf{f1} = 0$	$\mathsf{f0} = \mathsf{f1} = 1$	$\mathsf{f0} = \mathsf{f1} = 0$	$\mathsf{f0} = \mathsf{f1} = 1$
$\mathsf{f1} = \mathsf{f2} = 0$	$\mathsf{f2} = \mathsf{f3} = 1$	$\mathsf{f2} = \mathsf{f3} = 0$	$\mathsf{f1} = \mathsf{f2} = 1$
$\mathsf{f0} = \mathsf{f2} = 0$	$\mathsf{f1} = \mathsf{f2} = 1$	$\mathsf{f1} = \mathsf{f2} = 0$	$\mathsf{f0} = \mathsf{f2} = 1$
$\mathsf{f1} = \mathsf{f3} = 0$			$\mathsf{f1} = \mathsf{f3} = 1$

Now we return to the question whether cut-elimination in classical logic corresponds to non-deterministic computation: the preceding discussion suggests the following intuitive criterion for deciding whether two proofs are different.

> **Criterion:** Given two cut-free proofs of the sequent $M_1, M_2, S, T, A \vdash P$. These proofs are said to be *different*, if they consist of different tests.

According to this criterion, the two proofs given in Appendix A are different, since they consist of different tests. Let us point out that our criterion is sufficient for two proofs being different, but it is by no means a necessary one.

In the rest of this section we shall give a proof of Sequent (4.9), whose collection of normal forms includes the two proofs given in Appendix A. This proof contains instances of the cut-rule and is based upon the three observations listed below:

(i) From the assumption, A, one can prove that there are infinitely many '0's or infinitely many '1's on the tape.

(ii) If on the tape there are infinitely many '0's, then there are two cells with the same content.

(iii) Similarly, if on the tape there are infinitely many '1's, then there are two cells with the same content.

The formulae that represent the statements “there are infinitely many ‘0’s” and “there are infinitely many ‘1’s” are

$$\infty_0 \stackrel{\text{def}}{=} \forall \mathsf{n}.\exists \mathsf{k}.(\mathsf{n} \leq \mathsf{k} \land \mathsf{fk} = 0) \quad \text{and} \quad \infty_1 \stackrel{\text{def}}{=} \forall \mathsf{n}.\exists \mathsf{k}.(\mathsf{n} \leq \mathsf{k} \land \mathsf{fk} = 1)\ .$$

The sequent that corresponds to the first observation is

$$M_1, M_2, A \vdash \infty_0, \infty_1$$

for which a proof, denoted by τ, is given in Figure 4.5. The sequent that corresponds to the observations stated in (ii) and (iii) is

$$\infty_i, S, T \vdash P$$

where $i \in \{0, 1\}$. A proof of this sequent, denoted by ε_i, is given in Figure 4.5, too. Now we can use the cut-rule for forming the following proof for Sequent 4.9.

$$\sigma \left\{ \dfrac{\dfrac{\overset{\tau}{M_1, M_2, A \vdash \infty_0, \infty_1} \qquad \overset{\varepsilon_0}{\infty_1, S, T \vdash P}}{M_1, M_2, S, T, A \vdash P, \infty_0}\,\text{Cut} \qquad \overset{\varepsilon_1}{\infty_0, S, T \vdash P}}{M_1, M_2, S, T, A \vdash P}\,\text{Cut} \right. \tag{4.11}$$

If we feed the term that corresponds to this proof into the evaluation function of Section 4.1, we obtain a collection of normal forms. As mentioned earlier, this collection includes the proofs given in Appendix A.

Recall the point we made at the beginning of this section, where we argued that two different normal forms may correspond to the same value and the same computation. This point does not apply to the two normal forms given in Appendix A; these normal forms differ in “essential” features (identifying them would result into the “inconsistency” described in the introduction).

To sum up, let us take a step back and analyse a more global picture. Starting from the classical proof, σ, we can using our cut-elimination procedures reach, amongst others, one of the two normal forms given in Appendix A, by choosing non-deterministically which reductions we apply. This will enable us to encode an erratic choice operator into classical logic. We shall take up this point in the next section.

4.3 A Simple, Non-Deterministic Programming Language

In this section we address the question of what is the computational meaning of classical logic. Clearly, there are numerous ways of how to approach an answer to this question. However, the discussion in this section will be restricted to only one: we shall present a small programming language, whose type annotations correspond to some sequent proofs

$\boldsymbol{\tau} \stackrel{\text{def}}{=}$

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$n' \leq m^{n,n'} \vdash n' \leq m^{n,n'}$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.n' \leq m^{x,n'} \vdash n' \leq m^{n,n'}$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_2 \vdash n' \leq m^{n,n'}$}
\AxiomC{}
\UnaryInfC{$n \leq m^{n,n'} \vdash n \leq m^{n,n'}$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(x \leq m^{x,n'}) \vdash n \leq m^{n,n'}$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1 \vdash n \leq m^{n,n'}$}
\AxiomC{}
\UnaryInfC{$fm^{n,n'}=0 \vdash fm^{n,n'}=0$}
\AxiomC{}
\UnaryInfC{$fm^{n,n'}=1 \vdash fm^{n,n'}=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$fm^{n,n'}=0 \vee fm^{n,n'}=1 \vdash fm^{n,n'}=0, fm^{n,n'}=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0 \vee fx=1) \vdash fm^{n,n'}=0, fm^{n,n'}=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1, \forall x.(fx=0 \vee fx=1) \vdash n \leq m^{n,n'} \wedge fm^{n,n'}=0, fm^{n,n'}=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1, M_2, \forall x.(fx=0 \vee fx=1) \vdash n \leq m^{n,n'} \wedge fm^{n,n'}=0, n' \leq m^{n,n'} \wedge fm^{n,n'}=1$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1, M_2, \forall x.(fx=0 \vee fx=1) \vdash \exists k.(n \leq k \wedge fk=0), n' \leq m^{n,n'} \wedge fm^{n,n'}=1$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1, M_2, \forall x.(fx=0 \vee fx=1) \vdash \exists k.(n \leq k \wedge fk=0), \exists k'.(n' \leq k' \wedge fk'=1)$}
\RightLabel{$\forall_R$}
\UnaryInfC{$M_1, M_2, \forall x.(fx=0 \vee fx=1) \vdash \forall n.\exists k.(n \leq k \wedge fk=0), \exists k'.(n' \leq k' \wedge fk'=1)$}
\RightLabel{$\forall_R$}
\UnaryInfC{$M_1, M_2, \underbrace{\forall x.(fx=0 \vee fx=1)}_{A} \vdash \underbrace{\forall n.\exists k.(n \leq k \wedge fk=0)}_{\infty_0}, \underbrace{\forall n'.\exists k'.(n' \leq k' \wedge fk'=1)}_{\infty_1}$}
\end{prooftree}
$$

$\boldsymbol{\varepsilon_i} \stackrel{\text{def}}{=}$

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$sn \leq m \vdash sn \leq m$}
\AxiomC{}
\UnaryInfC{$n < m \vdash n < m$}
\RightLabel{$\supset_L$}
\BinaryInfC{$sn \leq m \supset n < m, sn \leq m \vdash n < m$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(sn \leq y \supset n < y), sn \leq m \vdash n < m$}
\RightLabel{$\forall_L$}
\UnaryInfC{$sn \leq m, S \vdash n < m$}
\AxiomC{}
\UnaryInfC{$fn=i \vdash fn=i$}
\AxiomC{}
\UnaryInfC{$fm=i \vdash fm=i$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$fm=i, fn=i \vdash fn=i \wedge fm=i$}
\AxiomC{}
\UnaryInfC{$fn=fm \vdash fn=fm$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(fn=i \wedge fm=i) \supset fn=fm, fm=i, fn=i \vdash fn=fm$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fn=i \wedge fy=i) \supset fn=fy), fm=i, fn=i \vdash fn=fm$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fx=i \wedge fy=i) \supset fx=fy), fm=i, fn=i \vdash fn=fm$}
\RightLabel{$\forall_L$}
\UnaryInfC{$fm=i, fn=i, T \vdash fn=fm$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$fm=i, sn \leq m, fn=i, S, T \vdash n < m \wedge fn=fm$}
\RightLabel{$\wedge'_L$}
\UnaryInfC{$fn=i, sn \leq m \wedge fm=i, S, T \vdash n < m \wedge fn=fm$}
\RightLabel{$\wedge_{L_2}$}
\UnaryInfC{$sn \leq m \wedge fm=i, 0 \leq n \wedge fn=i, S, T \vdash n < m \wedge fn=fm$}
\RightLabel{$\exists_R$}
\UnaryInfC{$sn \leq m \wedge fm=i, 0 \leq n \wedge fn=i, S, T \vdash \exists m.(n < m \wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$sn \leq m \wedge fm=i, 0 \leq n \wedge fn=i, S, T \vdash \exists n.m.(n < m \wedge fn=fm)$}
\RightLabel{$\exists_L$}
\UnaryInfC{$\exists k.(sn \leq k \wedge fk=i), 0 \leq n \wedge fn=i, S, T \vdash \exists n.m.(n < m \wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$0 \leq n \wedge fn=i, \forall n.\exists k.(n \leq k \wedge fk=i), S, T \vdash \exists n.m.(n < m \wedge fn=fm)$}
\RightLabel{$\exists_L$}
\UnaryInfC{$\exists k.(0 \leq k \wedge fk=i), \forall n.\exists k.(n \leq k \wedge fk=i), S, T \vdash \exists n.m.(n < m \wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\underbrace{\forall n.\exists k.(n \leq k \wedge fk=i)}_{\infty_i}, S, T \vdash \underbrace{\exists n.m.(n < m \wedge fn=fm)}_{P}$}
\end{prooftree}
$$

where $i \in \{0, 1\}$

Figure 4.5: The sequent proofs τ and ε_i.

of the $\supset$-fragment of classical logic and whose computations can be simulated by cut-elimination.

As we metioned in the introduction, the simply-typed lambda calculus can be seen as a prototypical programming language in which program execution (beta-reduction) is a form of sequential computation. In Section 3.1, we showed that this programming language gives a computational meaning to the $\supset$-fragment of intuitionistic logic (we defined translations between simply-typed lambda terms and proofs in this fragment, and showed that beta-reduction and cut-elimination can simulate each other). Consequently, we know that cut-elimination captures features of sequential computation. The main result of the preceding section was that cut-elimination in classical logic captures also features of non-deterministic computation. In this section we shall combine both results by extending the simply-typed lambda calculus with an erratic choice operator and show an analogous result to that given in Proposition 3.2.4. As will be seen shortly, this result requires classical logic: the erratic choice operator cannot be represented in intuitionistic logic.

Hereafter, we shall refer to the simply-typed lambda calculus extended with an erratic choice operator as λ_+-calculus, or λ_+ for short. Intuitively, a term of λ_+ corresponds to a sequential program, which executed may produce different results (see for example [Hennessy and Ashcroft, 1980]). The raw terms of λ_+ are given by the grammar

$$\begin{array}{lcll} M, N & ::= & x & \text{variable} \\ & | & \lambda x.M & \text{abstraction} \\ & | & M\,N & \text{application} \\ & | & M + N & \text{erratic choice} \end{array}$$

As usual, we are interested in only the set of well-typed terms of λ_+; that is in the set

$$\Lambda_+ \stackrel{\text{def}}{=} \left\{ M \;\middle|\; \Gamma \triangleright M : B \text{ is derivable using the rules given in Figure 4.6} \right\}$$

There are two reduction rules associated with λ_+: one is the beta-reduction rule (see Page 72) and the other is

$$M + N \quad \xrightarrow{+} \quad M \text{ or } N$$

The reduction system of λ_+ is thus defined as $(\Lambda_+, \xrightarrow{\beta+})$, where $\xrightarrow{\beta+}$ consists of $\xrightarrow{\beta}$-reductions and $\xrightarrow{+}$-reductions, which both are assumed to be closed under term formation.

Recall Lafont's example given on Page 4, and consider the term $M+N$. The dynamic behaviour of this term corresponds exactly to the behaviour of cut-elimination in Lafont's example. Thus the translation $|\_|^{\mathsf{s}+} : \Lambda_+ \to \mathcal{T}$ is as follows.

$$\begin{array}{rcll} |x|^{\mathsf{s}+}_c & \stackrel{\text{def}}{=} & \mathsf{Ax}(x, c) & \\ |\lambda x.M|^{\mathsf{s}+}_c & \stackrel{\text{def}}{=} & \mathsf{Imp}_R((x)\langle a\rangle\,|M|^{\mathsf{s}+}_a, c) & \\ |M\;N|^{\mathsf{s}+}_c & \stackrel{\text{def}}{=} & |M|^{\mathsf{s}+}_a[a := (y)\mathsf{Imp}_L(\langle b\rangle|N|^{\mathsf{s}+}_b, (z)\mathsf{Ax}(z,c), y)] & (1) \\ |M+N|^{\mathsf{s}+}_c & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle|M|^{\mathsf{s}+}_c, (x)|N|^{\mathsf{s}+}_c) & (2) \end{array}$$

$$\frac{}{x:B,\Gamma \triangleright x:B}$$

$$\frac{x:B,\Gamma \triangleright M:C}{\Gamma \triangleright \lambda x.M:B{\supset}C}\ \text{Abs} \qquad \frac{\Gamma \triangleright M:B{\supset}C \quad \Gamma \triangleright N:B}{\Gamma \triangleright M\,N:C}\ \text{App}$$

$$\frac{\Gamma \triangleright M:B \quad \Gamma \triangleright N:B}{\Gamma \triangleright M+N:B}\ +$$

Figure 4.6: Typing rules for λ^{+}.

where in (1) $y \notin FN(|N|_b^{\mathsf{s}+})$, and in (2) $a \notin FC(|M|_a^{\mathsf{s}+})$ and $x \notin FN(|M|_a^{\mathsf{s}+})$. Next, we would like to show that this reduction respects typing judgements.

Proposition 4.3.1:
If $M \in \Lambda_{+}$ with the typing judgement $\Gamma \triangleright M:B$, then $|M|_a^{\mathsf{s}+} \in \mathfrak{T}$ with the typing judgement $\Gamma \triangleright |M|_a^{\mathsf{s}+} \triangleright a:B$.

Proof. By routine induction on the structure of M. □

Finally, we can prove that cut-elimination in $\mathfrak{T}$ can simulate normalisation in λ_{+}.

Theorem 4.3.2:

If $M, N \in \Lambda_{+}$ and $M \xrightarrow{\beta+} N$, then for all co-names a, $|M|_a^{\mathsf{s}+} \xrightarrow{cut}{}^{+} |N|_a^{\mathsf{s}+}$.

Proof. Analogous to proof of Proposition 3.2.4. □

On Page 14 we introduced three criteria for designing a cut-elimination procedure. Let us analyse these criteria with respect to the simulation result just given. In particular, let us analyse whether we would have, given analogous translations, obtained the simulation result with an existing cut-elimination procedure for classical logic. We analyse in turn three procedures by Gentzen [1935], Dragalin [1988] and Danos et al. [1997].

- Gentzen's original cut-elimination procedure is defined as innermost reductions strategy. Consequently, we would have obtained the weaker result saying that cut-elimination can simulate only a reduction strategy in λ_{+}. However, as soon as we impose an innermost strategy on the reductions of λ_{+}, we "lose" some normal forms; by this we mean that given a term $M \in \Lambda_{+}$ some of its normal forms are no longer reachable.

- Dragalin's cut-elimination procedure is strongly normalising, and thus we would hope that this procedure is able to simulate every reduction in λ_{+}; unfortunately, Dragalin does not consider rules that allow cuts to be permuted with other cuts. However, this, as remarked in Section 3.2, is necessary to simulate beta-reduction.

- The cut-elimination procedure developed by Danos et. al. for LK^{tq} allows cuts to be permuted with other cuts and is strongly normalising, but every formula requires a colour annotation predetermining how commuting cuts are reduced. Notice, however, that there is no possibility of predetermining the behaviour of a term with an erratic choice. So also in this case an analogous simulation result would fail.

As far as I have been able to ascertain, no other cut-elimination procedure for classical logic gives a simulation result with λ_+.

Clearly, the fragment of classical logic into which we translated Λ_+ is very weak: no new sequents are derivable in comparison with intuitionistic logic. Important future work is to extend suitably the programming language λ_+ so that a simulation result is obtained for larger fragments of classical logic. Good candidates for extensions are the control operators $\mathcal{A}$ and $\mathcal{C}$, which were studied previously in the context of $\lambda\mu$ (see for example [Bierman, 1998]).

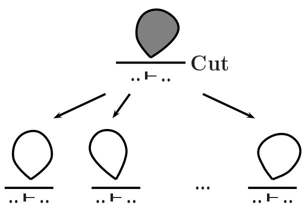
4.4 Notes

In order to show that the two normal forms given in Appendix A differ in essential features we had to extract some kind of data from them. Kreisel [1958, Page 163] demonstrated that extracting data from classical proofs is *not* possible in general. But it is known that data can always be extracted in certain fragments of classical logic, as shown for instance by Friedman [1978]. Our example falls into such a fragment of classical logic, and so the non-determinism, which we inferred from the existence of two different normal forms, is independent from the result by Kreisel. Let us conclude this chapter with showing that the colour annotation introduced in LK^{tq} completely hides this non-determinism.

4.4.1 A Beautiful Theory for the Colours in LK^{tq}, Slain by A Fact

The cut-elimination procedure in LK^{tq} has three remarkable properties: it is strongly normalising, confluent, and there is a strong connection to linear logic since there is a simple translation between LK^{tq}-proofs and proof nets. Furthermore, cut-elimination in LK^{tq} has a similar dynamic behaviour as cut-elimination in proof nets [Danos et al., 1997]. Particularly interesting is the confluence property of LK^{tq}: it precludes the "inconsistency" described in the introduction. This property holds in LK^{tq} because the colour annotations constrain the cut-reductions so that critical pairs cannot arise.

Interestingly, the colour annotation in LK^{tq} acts as a "selector", in that it predetermines the normal form to be reached by cut-elimination. Let us illustrate this point. Starting from a classical proof our cut-elimination procedures give, in general, rise to several normal forms, as shown below.



In contrast, the cut-elimination procedure in LK^{tq} gives rise to only *one* normal form. However, by changing the colour annotation, we can obtain another normal forms. An appealing explanation of the colour annotations would thus be to interpret a classical proof as a juxtaposition of a collection of "programs" and to regard the colours as means of selecting one of these "programs".

Unfortunately, this explanation fades in the light of the following example. Consider the cut-instance

$$\dfrac{\begin{array}{c}\sigma\\ M_1, M_2, S, T, A \vdash P\end{array} \qquad \dfrac{\overline{P \vdash P} \quad \overline{P \vdash P}}{P \vdash P \wedge P}\wedge_R}{M_1, M_2, S, T, A \vdash P \wedge P}\,\text{Cut}$$

where σ is the classical proof given in Section 4.2. One possibility for eliminating the cut is to replace the axioms by two instances of σ, as illustrated in the following proof.

$$\dfrac{\begin{array}{c}\sigma\\ M_1, M_2, S, T, A \vdash P\end{array} \quad \begin{array}{c}\sigma\\ M_1, M_2, S, T, A \vdash P\end{array}}{M_1, M_2, S, T, A \vdash P \wedge P}\wedge_R$$

Now let one copy of σ reduce to the first normal form shown in Appendix A and the other to the second normal form. Barbanera et al. [1997] showed that this particular dynamic behaviour of cut-elimination cannot be achieved using a cut-elimination procedure that depends on colour annotations—the normal form outlined above is not reachable. However, using our intuitive interpretation given in Section 4.2, this normal form corresponds to a pair of case analyses. Taking into account that we obtained strongly normalising cut-elimination procedures *not* depending on colour annotations, we regard the colours of Danos et al. [1997] as a restriction that simplifies the task of proving the strong normalisation property (it enabled them to appeal to the strong normalisation proof for proof nets).

Chapter 5

Conclusion

...it used to be said that classical proof theory does not really exist.

—J. M. E. Hyland
in Proof Theory in the Abstract, 2000.

The theme running through this thesis was that non-determinism should be seen as an *intrinsic* property of classical logic. In effect, we departed from the traditional doctrine, prevailing in intuitionistic and linear logic, that considers cut-elimination as an equality preserving operation on proofs. A pleasing consequence of our point of view is that the "inconsistency" problem arising from Lafont's example (see Page 4) is completely bypassed.

To support our point of view, we reconsidered the proof theory of classical logic and developed several strongly normalising cut-elimination procedures (Chapter 2). One feature recurring in all these procedures is that a subderivation of a commuting cut has to be permuted directly to the place or places where the cut-formula is introduced. This feature was taken from the strongly normalising cut-elimination procedure developed by Danos et al. [1997] for LK^{tq}; however, we showed that their colour annotation, a restriction they imposed, is not necessary to ensure strong normalisation. In fact, no additional information to guide the rewrite system is required at all for strong normalisation. As a pleasing consequence, more normal forms are often reachable from classical proofs containing cuts (see Figure 2.1 for an example). Owing to the smaller number of constraints of our cut-elimination procedures, strong normalisation cannot be proved by translating every cut-reduction onto a series of cut-reductions of proof-nets, as was done for LK^{tq}. Fortunately, the use of term annotations for sequent proofs enabled us to adapt proof techniques from term rewriting [Barbanera and Berardi, 1994, Bloo and Geuvers, 1999], and thus we were able to give direct proofs for strong normalisation.

Another pleasing feature of our cut-elimination procedures is that via simple translations they imply strong normalisation of beta-reduction in the simply-typed lambda calculus (Chapter 3). This was achieved by considering reduction rules that allow cuts to be permuted with other cuts. If these reduction rules are defined naïvely, they cause loops and thereby break the strong normalisation property. We however retained this property by introducing syntactic means (e.g., labelled cuts) which prevent interference between these rules and rules that simply permute a cut upwards in a proof. Because of the generality of our cut-elimination procedures, we could answer the correspondence question between cut-elimination and normalisation positively for all connectives in classical logic. This result improves upon work presented by Zucker [1974].

In Chapter 4 we studied applications of cut-elimination. For one of our cut-elimination procedures we presented an implementation, which we employed to calculate for a particular classical proof two cut-free proofs. We showed that these two proofs differ in essential features and therefore should not be identified. Consequently, in classical logic the process of cut-elimination corresponds to non-deterministic computation. In the light of this result, we proposed a different and more positive reading for Lafont's example: it is an erratic choice operator, which, for example, has an intuitive computational meaning in concurrency theory. We demonstrated that if one adds Lafont's example to the implicational fragment of intuitionistic logic, then this fragment can simulate reduction in the simply-typed lambda calculus enriched with the erratic choice operator.

It is hoped that the reader will now be convinced of the point of view that non-determinism *is* an intrinsic feature of classical logic. This point of view was first taken by Barbanera and Berardi [1994], and later analysed from a programming point of view by Barbanera et al. [1997];[1] it was also hinted at by Coquand [1995] and Herbelin [1995], and more recently suggested by Girard [2001, Page 88] and advocated by Hyland [2000].

5.1 Further Work

There are many areas for further work. We list some of them below, roughly in the order as they appear in the thesis.

Pleasant though our strong normalisation result for cut-elimination is, it is not the strongest result we could hoped for: our strong normalisation theorems say little about strong normalisation of cut-elimination in second-order classical logic. Clearly, it would be nice to have the stronger result: for example it can save lengthy strong normalisation proofs from first principles for other term rewrite systems (see [Benton, 1995]). To give a proof of the stronger result, one has to extend the notion of symmetric reducibility candidates using what is by Gallier [1990] referred to as "Girard's trick". We have not checked whether this extension is possible given the fixed point operation we employed in the propositional and first-order case.

There are several possibilities of how to extend our sequent calculus in order to capture, for example, features of recursive computation. One is to add the induction rule

$$\frac{B[\mathsf{x} := \mathsf{y}], \Gamma \vdash \Delta, B[\mathsf{x} := \mathsf{y}+1]}{B[\mathsf{x} := 0], \Gamma \vdash \Delta, B[\mathsf{x} := \mathsf{t}]}\ \text{Induction}$$

where y is an eigenvariable not occurring elsewhere in the premise, and t is an arbitrary expression. Another is to allow non-logical axioms of the form $\overline{C \vdash D}$. In general, the latter means that some cuts cannot be eliminated. To avoid this problem one has to impose

[1] The main technical novelty in this thesis is that we showed the non-determinism using a standard formulation of classical logic; for this we had to extend their results in order to deal with all connectives.

suitable constraints on the form of the non-logical axioms. Both extensions have been investigated in an intuitionistic setting by McDowell and Miller [2000], and they seem not to pose any difficulties with respect to our strong normalisation proof based on candidates.

We designed our cut-elimination procedures so that restrictions imposed in earlier cut-elimination procedures are removed, but without breaking the strong normalisation property. For example, we showed that the colour annotations of LK^{tq} are not required for strong normalisation. But, we kept one restriction of LK^{tq}: commuting cuts can only be permuted into one direction and not into alternating ones. One side effect of this restriction is that some normal forms are not reachable, which would however be reachable using a completely unrestricted (and thus not strongly normalising) cut-elimination procedure. Consider again the infinite reduction sequence outlined in Example 2.1.3. Starting from the first proof, one can reach the normal form given in Figure 5.1, but we are not aware of any strongly normalising cut-elimination procedure with which we could reach it.[2] The only normal forms that can be reached using our strongly normalising cut-elimination procedures are

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\lor A\vdash A,A}\lor_L}{A\lor A\vdash A}\mathrm{Contr}_R\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\lor A\vdash A,A}\lor_L}{A\lor A\vdash A}\mathrm{Contr}_R}{A\lor A, A\lor A\vdash A\land A}\land_R}{A\lor A\vdash A\land A}\mathrm{Contr}_L \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\land A}\land_R}{A\vdash A\land A}\mathrm{Contr}_L\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\land A}\land_R}{A\vdash A\land A}\mathrm{Contr}_L}{A\lor A\vdash A\land A, A\land A}\lor_L}{A\lor A\vdash A\land A}\mathrm{Contr}_R$$

We find it is highly desirable to have a convincing explanation for what it means computationally that the normal form given in Figure 5.1 is not reachable using our procedures. Because there are some similarities between the two normal forms given above and the one in Figure 5.1, an appealing explanation would be that using our cut-elimination procedure only "essential" normal forms can be reached. However at present, this is only a vague hope.

In Section 4.1 we developed an implementation of the cut-elimination procedure $(\mathcal{T}, \xrightarrow{aux})$ in OCaml. A more suitable programming language for this kind of implementations has been recently proposed by Pitts and Gabbay [2000]. In this programming language, named FreshML, one does not have to deal explicitly with alpha-conversions, and thus one can write much clearer code reminiscent of the simple definitions we presented in this thesis. As soon as there is a working prototype available for FreshML, we hope to provide a simplified implementation of $(\mathcal{T}, \xrightarrow{aux})$.

Griffin [1990] noted a correspondence between classical logic and functional programming languages with control operators. This observation has been exploited in many works, for example [Murthy, 1990, Berger et al., 2000], where programs are extracted from classical proofs via double negation translations. An interesting twist of this approach has been studied by Coquand [1995] and Herbelin [1995]. They analyse a variant of the classical proof given in Section 4.2 and extract two different programs by using

[2] The strongly normalising cut-elimination procedure seems not to be meaningful, where one can do finitely many steps "however one wants", and then annotate colours and reduce the proof according to the cut-elimination procedure of LK^{tq}.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A, A\vdash A\wedge A}\wedge_R}{A\vdash A\wedge A}\text{Contr}_L\qquad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A, A\vdash A\wedge A}\wedge_R}{A\vdash A\wedge A}\text{Contr}_L}{A\vee A\vdash A, A\wedge A}\vee_L\quad\dfrac{}{A\vdash A}}{A, A\vee A\vdash A\wedge A, A\wedge A}\wedge_R}{A\vee A, A\vee A\vdash A\wedge A, A\wedge A, A\wedge A}\vee_L}{A\vee A\vdash A\wedge A, A\wedge A, A\wedge A}\text{Contr}_L}{A\vee A\vdash A\wedge A, A\wedge A}\text{Contr}_R}{A\vee A\vdash A\wedge A}\text{Contr}_R$$

Figure 5.1: A normal form of the LK-proof given in Example 2.1.3, Page 15. This normal form is reachable only if commuting cuts can "change" the direction into which they are permuted.

two different double negation translations. We conjecture that double negation translations correspond to particular strategies of cut-elimination. This conjecture does not seem difficult to verify, but the details remain future work.

Important future work is to reconsider the proof theory of classical logic in a setting where proofs are identified up to *Kleene permutations* [Kleene, 1952b], analogous to proof nets for linear logic [Girard, 1987a]. One practical reason for this identification is the huge number of normal forms we found by feeding the classical proof given in Section 4.2 into our implementation. We conjecture many of these normal forms are equal up to Kleene permutations. In our implementation we introduced some *ad hoc* assumptions, which reduced the number of normal forms somewhat, but we felt that we did not have much control over the cut-elimination process. Identification up to Kleene permutations might alleviate this problem. Unfortunately, syntactic methods (e.g. additional reduction rules) to achieve this identification are rather difficult to obtain [Schwichtenberg, 1999]. So one might have to resort to semantical methods, as proposed by Hyland [2000], instead.

The most interesting future work concerns the computational interpretation of classical proofs. In Section 4.3 we gave an encoding into classical logic of a simply-typed lambda calculus enriched with an erratic choice operator. This result should be seen as proof of concept: it barely scratches at the surface of what can be encoded into classical logic. For example, we did not consider an encoding of a (weak) form of communication into classical logic using quantifiers. For example the cut-instance

$$\dfrac{\dfrac{\ldots\vdash\ldots, B\mathsf{a}}{\ldots\vdash\ldots,\forall\mathsf{x}.B\mathsf{x}}\forall_R\qquad\dfrac{B\mathsf{t},\ldots\vdash\ldots}{\forall\mathsf{x}.B\mathsf{x},\ldots\vdash\ldots}\forall_L}{\ldots\vdash\ldots}\text{Cut}$$

which reduces to

$$\dfrac{\ldots\vdash\ldots, B\mathsf{t}\qquad B\mathsf{t},\ldots\vdash\ldots}{\ldots\vdash\ldots}\text{Cut}$$

can be understood as communicating the expression t from the proof on the right-hand side to the one on the left-hand side. Using multiple quantifiers, one can also encode certain dependencies between communications, as illustrated with the cut-instance

$$\dfrac{\dfrac{\dfrac{\ldots \vdash \ldots, B\,\mathsf{a}\,\mathsf{t}_1}{\ldots \vdash \ldots, \forall \mathsf{x}.B\,\mathsf{x}\,\mathsf{t}_1}\,\forall_R}{\ldots \vdash \ldots, \exists \mathsf{y}.\forall \mathsf{x}.B\,\mathsf{x}\,\mathsf{y}}\,\exists_R \qquad \dfrac{\dfrac{B\,\mathsf{t}_2\,\mathsf{b}, \ldots \vdash \ldots}{\forall \mathsf{x}.B\,\mathsf{x}\,\mathsf{b}, \ldots \vdash \ldots}\,\forall_L}{\exists \mathsf{y}.\forall \mathsf{x}.B\,\mathsf{x}\,\mathsf{y}, \ldots \vdash \ldots}\,\exists_L}{\ldots \vdash \ldots}\,\text{Cut}$$

which reduces so that first t_1 is communicated to the proof on the right-hand side and then t_2 to the one on the left-hand side. Our cut-elimination procedures provide a framework to study this idea further. In terms of a computational interpretation, most unclear, however, are cut-instances whose cut-formula is contracted on both sides.

$$\dfrac{\dfrac{\ldots \vdash \ldots, C, C}{\ldots \vdash \ldots, C}\,\text{Contr}_R \qquad \dfrac{C, C, \ldots \vdash \ldots}{C, \ldots \vdash \ldots}\,\text{Contr}_L}{\ldots \vdash \ldots}\,\text{Cut}$$

Appendix A

Experimental Data

In this appendix we shall give two normal forms for the proof, σ, given on Page 118. Both normal forms prove, by analysing some cases, Sequent (4.9) shown on Page 116. To find which cases are analysed, one has to look for proof fragments of the following form.

$$
\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots \\ \ldots, (\mathsf{f1} = 1 \wedge \mathsf{f0} = 1) \supset \mathsf{f0} = \mathsf{f1}, \ldots \vdash \ldots\end{array}}{\ldots, \forall \mathsf{y}.((\mathsf{fy} = 1 \wedge \mathsf{f1} = 1) \supset \mathsf{f0} = \mathsf{fy}), \ldots \vdash \ldots}\,\forall_L}{\ldots, \forall \mathsf{x}.\mathsf{y}.((\mathsf{fy} = 1 \wedge \mathsf{fx} = 1) \supset \mathsf{fx} = \mathsf{fy}), \ldots \vdash \ldots}\,\forall_L}{\ldots, \forall \mathsf{i}.\mathsf{x}.\mathsf{y}.((\mathsf{fy} = \mathsf{i} \wedge \mathsf{fx} = \mathsf{i}) \supset \mathsf{fx} = \mathsf{fy}), \ldots \vdash \ldots}\,\forall_L
$$

This particular proof fragment indicates that the case $\mathsf{f0} = \mathsf{f1} = 1$ is analysed. The first normal form (Pages 132–134) analyses the cases

$$
\begin{array}{llll}
\mathsf{f0} = \mathsf{f1} = 0 & \mathsf{f1} = \mathsf{f2} = 0 & \mathsf{f0} = \mathsf{f1} = 1 & \mathsf{f2} = \mathsf{f3} = 1 \\
\mathsf{f0} = \mathsf{f2} = 0 & \mathsf{f1} = \mathsf{f3} = 0 & \mathsf{f1} = \mathsf{f2} = 1 &
\end{array}
$$

and the second (Pages 135–140) the cases

$$
\begin{array}{llll}
\mathsf{f0} = \mathsf{f1} = 0 & \mathsf{f2} = \mathsf{f3} = 0 & \mathsf{f0} = \mathsf{f1} = 1 & \mathsf{f1} = \mathsf{f2} = 1 \\
\mathsf{f1} = \mathsf{f2} = 0 & & \mathsf{f0} = \mathsf{f2} = 1 & \mathsf{f1} = \mathsf{f3} = 1
\end{array}
$$

We take this as evidence that both normal forms are “essentially” different (see the criterion given on Page 117). Because of their size, the two normal forms are spread over the following nine figures.

$$
\underbrace{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{3\leq 3\vdash 3\leq 3}}{\forall x.(3\leq m^{x,3})\vdash 3\leq 3}{\scriptstyle\forall_L}}{M_2\vdash 3\leq 3}{\scriptstyle\forall_L}\quad\dfrac{}{2<3\vdash 2<3}}{3\leq 3\supset 2<3,\ M_2\vdash 2<3}{\scriptstyle\supset_L}}{\forall y.(3\leq y\supset 2<y),\ M_2\vdash 2<3}{\scriptstyle\forall_L}}{M_2,\ S\vdash 2<3}{\scriptstyle\forall_L}\quad\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\\ X_2\end{matrix}\quad\dfrac{}{f2=f3\vdash f2=f3}}{(f3=1\wedge f2=1)\supset f2=f3,\ M_1,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f2=f3,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f1=1}{\scriptstyle\supset_L}}{\forall y.((fy=1\wedge f2=1)\supset f2=fy),\ M_1,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f2=f3,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f1=1}{\scriptstyle\forall_L}}{\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ M_1,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f2=f3,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f1=1}{\scriptstyle\forall_L}}{M_1,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f2=f3,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f1=1}{\scriptstyle\forall_L}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ 2<3\wedge f2=f3,\ f1=1}{\scriptstyle\wedge_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash\exists m.(1<m\wedge f1=fm),\ 1<2\wedge f1=f2,\ 2<3\wedge f2=f3,\ f1=1}{\scriptstyle\exists_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash 1<2\wedge f1=f2,\ 2<3\wedge f2=f3,\ f1=1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash\exists m.(1<m\wedge f1=fm),2<3\wedge f2=f3,\ f1=1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash 2<3\wedge f2=f3,\ f1=1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash\exists m.(2<m\wedge f2=fm),\ f1=1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f1=1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists_R}\quad\begin{matrix}\vdots\\ X_5\end{matrix}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f1=1\wedge f0=1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\wedge_R}}_{X_6}
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{1\leq 1\vdash 1\leq 1}}{\forall x.(1\leq m^{x,1})\vdash 1\leq 1}{\scriptstyle\forall_L}}{M_2\vdash 1\leq 1}{\scriptstyle\forall_L}\quad\dfrac{}{0<1\vdash 0<1}}{1\leq 1\supset 0<1,\ M_2\vdash 0<1}{\scriptstyle\supset_L}}{\forall y.(1\leq y\supset 0<y),\ M_2\vdash 0<1}{\scriptstyle\forall_L}}{M_2,\ S\vdash 0<1}{\scriptstyle\forall_L}\quad\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\\ X_6\end{matrix}\quad\dfrac{}{f0=f1\vdash f0=f1}}{(f1=1\wedge f0=1)\supset f0=f1,\ M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f0=f1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\supset_L}}{\forall y.((fy=1\wedge f0=1)\supset f0=fy),\ M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f0=f1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\forall_L}}{\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f0=f1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\forall_L}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash f0=f1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\forall_L}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash 0<1\wedge f0=f1,\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\wedge_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash\exists m.(0<m\wedge f0=fm),\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists_R}}{M_1,\ M_2,\forall x.(fx=0\vee fx=1),\ S,\ T\vdash\exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists_R}
$$

$M_1 = \forall y.x.(x \leq m^{x,y})$ $\quad T = \forall i.x.y.((fy = i \wedge fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \leq m^{x,y})$ $\quad S = \forall x.y.(sx \leq y \supset x < y)$

Figure 1: First normal form and Subproof X_6.

$$\underbrace{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{2\le 2\vdash 2\le 2}}{\forall x.(x\le m^{x,0})\vdash 2\le 2}\forall_L}{M_1\vdash 2\le 2}\forall_L\qquad\dfrac{}{1<2\vdash 1<2}}{2\le 2\supset 1<2,\ M_1\vdash 1<2}\supset_L}{\forall y.(2\le y\supset 1<y),\ M_1\vdash 1<2}\forall_L}{M_1,\ S\vdash 1<2}\forall_L\qquad\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f2=0\vdash f2=0}\quad\dfrac{}{f2=1\vdash f2=1}}{f2=0\lor f2=1\vdash f2=0,\ f2=1}\lor_L}{\forall x.(fx=0\lor fx=1)\vdash f2=0,\ f2=1}\forall_L\quad\dfrac{\dfrac{\dfrac{}{f1=0\vdash f1=0}\quad\dfrac{}{f1=1\vdash f1=1}}{f1=0\lor f1=1\vdash f1=0,\ f1=1}\lor_L}{\forall x.(fx=0\lor fx=1)\vdash f1=0,\ f1=1}\forall_L}{\forall x.(fx=0\lor fx=1)\vdash f2=0\land f1=0,\ f2=1,\ f1=1}\land_R\quad\dfrac{}{f1=f2\vdash f1=f2}}{(f2=0\land f1=0)\supset f1=f2,\forall x.(fx=0\lor fx=1)\vdash f1=f2,\ f2=1,\ f1=1}\supset_L}{\forall y.((fy=0\land f1=0)\supset f1=fy),\forall x.(fx=0\lor fx=1)\vdash f1=f2,\ f2=1,\ f1=1}\forall_L}{\forall x.y.((fy=0\land fx=0)\supset fx=fy),\forall x.(fx=0\lor fx=1)\vdash f1=f2,\ f2=1,\ f1=1}\forall_L}{\forall x.(fx=0\lor fx=1),\ T\vdash f1=f2,\ f2=1,\ f1=1}\forall_L}{M_1,\forall x.(fx=0\lor fx=1),\ S,\ T\vdash f2=1,\ 1<2\land f1=f2,\ f1=1}\land_R}_{X_1}$$

$$\underbrace{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{2\le 3\vdash 2\le 3}}{\forall x.(x\le m^{x,3})\vdash 2\le 3}\forall_L}{M_1\vdash 2\le 3}\forall_L\qquad\dfrac{}{1<3\vdash 1<3}}{2\le 3\supset 1<3,\ M_1\vdash 1<3}\supset_L}{\forall y.(2\le y\supset 1<y),\ M_1\vdash 1<3}\forall_L}{M_1,\ S\vdash 1<3}\forall_L\qquad\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f3=0\vdash f3=0}\quad\dfrac{}{f3=1\vdash f3=1}}{f3=0\lor f3=1\vdash f3=0,\ f3=1}\lor_L}{\forall x.(fx=0\lor fx=1)\vdash f3=0,\ f3=1}\forall_L\quad\dfrac{\dfrac{\dfrac{}{f1=0\vdash f1=0}\quad\dfrac{}{f1=1\vdash f1=1}}{f1=0\lor f1=1\vdash f1=0,\ f1=1}\lor_L}{\forall x.(fx=0\lor fx=1)\vdash f1=0,\ f1=1}\forall_L}{\forall x.(fx=0\lor fx=1)\vdash f3=0\land f1=0,\ f3=1,\ f1=1}\land_R\quad\dfrac{}{f1=f3\vdash f1=f3}}{(f3=0\land f1=0)\supset f1=f3,\forall x.(fx=0\lor fx=1)\vdash f1=f3,\ f3=1,\ f1=1}\supset_L}{\forall y.((fy=0\land f1=0)\supset f1=fy),\forall x.(fx=0\lor fx=1)\vdash f1=f3,\ f3=1,\ f1=1}\forall_L}{\forall x.y.((fy=0\land fx=0)\supset fx=fy),\forall x.(fx=0\lor fx=1)\vdash f1=f3,\ f3=1,\ f1=1}\forall_L}{\forall x.(fx=0\lor fx=1),\ T\vdash f1=f3,\ f3=1,\ f1=1}\forall_L}{M_1,\forall x.(fx=0\lor fx=1),\ S,\ T\vdash f3=1,\ 1<3\land f1=f3,\ f1=1}\land_R\qquad\begin{matrix}\vdots\\ X_1\end{matrix}}{M_1,\forall x.(fx=0\lor fx=1),\ S,\ T\vdash f3=1\land f2=1,\ 1<3\land f1=f3,\ 1<2\land f1=f2,\ f1=1}\land_R}_{X_2}$$

$M_1 = \forall y.x.(x \le m^{x,y})$ $\quad T = \forall i.x.y.((fy = i \land fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \le m^{x,y})$ $\quad S = \forall x.y.(sx \le y \supset x < y)$

Figure 2: Subproofs X_1 and X_2.

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$1\leq 1 \vdash 1\leq 1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(x\leq m^{x,0}) \vdash 1\leq 1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1 \vdash 1\leq 1$}
\AxiomC{}
\UnaryInfC{$0<1 \vdash 0<1$}
\RightLabel{$\supset_L$}
\BinaryInfC{$1\leq 1\supset 0<1,\ M_1 \vdash 0<1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(1\leq y\supset 0<y),\ M_1 \vdash 0<1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1, S \vdash 0<1$}
\AxiomC{}
\UnaryInfC{$f1=0 \vdash f1=0$}
\AxiomC{}
\UnaryInfC{$f1=1 \vdash f1=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f1=0\vee f1=1 \vdash f1=0,\ f1=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f1=0,\ f1=1$}
\AxiomC{}
\UnaryInfC{$f0=0 \vdash f0=0$}
\AxiomC{}
\UnaryInfC{$f0=1 \vdash f0=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f0=0\vee f0=1 \vdash f0=0,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f0=0,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f1=0\wedge f0=0,\ f1=1,\ f0=1$}
\AxiomC{}
\UnaryInfC{$f0=f1 \vdash f0=f1$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f1=0\wedge f0=0)\supset f0=f1, \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=0\wedge f0=0)\supset f0=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=0\wedge fx=0)\supset fx=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1),\ T \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$\underbrace{M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=1,\ 0<1\wedge f0=f1,\ f0=1}_{X_3}$}
\end{prooftree}
$$

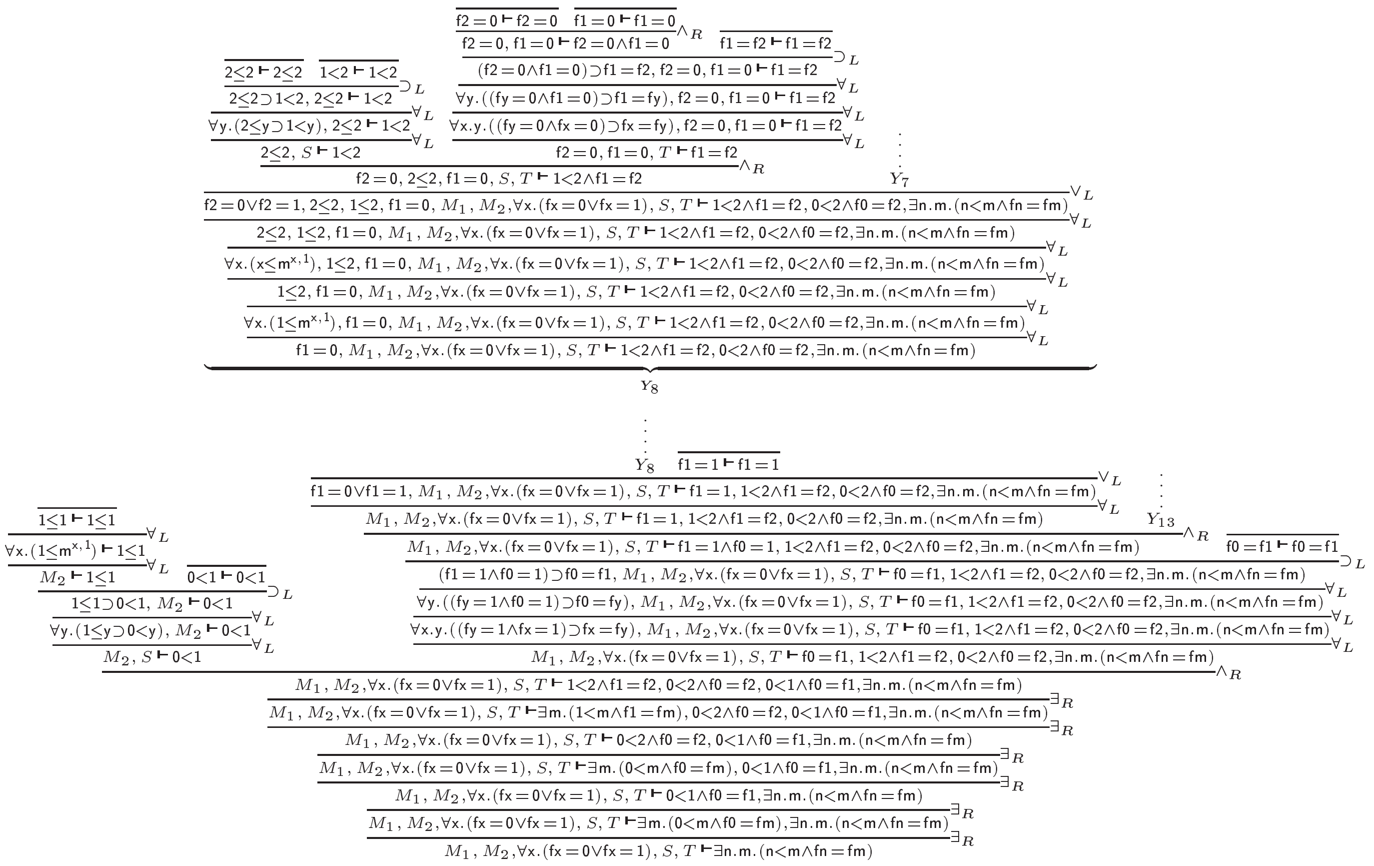
$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$1\leq 2 \vdash 1\leq 2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(x\leq m^{x,2}) \vdash 1\leq 2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1 \vdash 1\leq 2$}
\AxiomC{}
\UnaryInfC{$0<2 \vdash 0<2$}
\RightLabel{$\supset_L$}
\BinaryInfC{$1\leq 2\supset 0<2,\ M_1 \vdash 0<2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(1\leq y\supset 0<y),\ M_1 \vdash 0<2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1, S \vdash 0<2$}
\AxiomC{}
\UnaryInfC{$f2=0 \vdash f2=0$}
\AxiomC{}
\UnaryInfC{$f2=1 \vdash f2=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f2=0\vee f2=1 \vdash f2=0,\ f2=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f2=0,\ f2=1$}
\AxiomC{}
\UnaryInfC{$f0=0 \vdash f0=0$}
\AxiomC{}
\UnaryInfC{$f0=1 \vdash f0=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f0=0\vee f0=1 \vdash f0=0,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f0=0,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f2=0\wedge f0=0,\ f2=1,\ f0=1$}
\AxiomC{}
\UnaryInfC{$f0=f2 \vdash f0=f2$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f2=0\wedge f0=0)\supset f0=f2, \forall x.(fx=0\vee fx=1) \vdash f0=f2,\ f2=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=0\wedge f0=0)\supset f0=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f2,\ f2=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=0\wedge fx=0)\supset fx=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f2,\ f2=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1),\ T \vdash f0=f2,\ f2=1,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f2=1,\ 0<2\wedge f0=f2,\ f0=1$}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{X_3}
\RightLabel{$\wedge_R$}
\BinaryInfC{$\underbrace{M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f2=1\wedge f1=1,\ 0<2\wedge f0=f2,\ 0<1\wedge f0=f1,\ f0=1}_{X_4}$}
\end{prooftree}
$$

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$2\leq 2 \vdash 2\leq 2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(2\leq m^{x,2}) \vdash 2\leq 2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_2 \vdash 2\leq 2$}
\AxiomC{}
\UnaryInfC{$1<2 \vdash 1<2$}
\RightLabel{$\supset_L$}
\BinaryInfC{$2\leq 2\supset 1<2,\ M_2 \vdash 1<2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(2\leq y\supset 1<y),\ M_2 \vdash 1<2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_2, S \vdash 1<2$}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{X_4}
\AxiomC{}
\UnaryInfC{$f1=f2 \vdash f1=f2$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f2=1\wedge f1=1)\supset f1=f2,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 0<2\wedge f0=f2,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=1\wedge f1=1)\supset f1=fy),\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 0<2\wedge f0=f2,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 0<2\wedge f0=f2,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 0<2\wedge f0=f2,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<2\wedge f0=f2,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(1<m\wedge f1=fm),\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$\underbrace{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f0=1, \exists n.m.(n<m\wedge fn=fm)}_{X_5}$}
\end{prooftree}
$$

$M_1 = \forall y.x.(x \leq m^{x,y})$ $\quad T = \forall i.x.y.((fy = i \wedge fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \leq m^{x,y})$ $\quad S = \forall x.y.(sx \leq y \supset x < y)$

Figure 3: Subproofs X_3, X_4 and X_5.



$M_1 = \forall \mathsf{y.x.}(\mathsf{x} \leq \mathsf{m}^{\mathsf{x,y}})$ $\quad T = \forall \mathsf{i.x.y.}((\mathsf{fy} = \mathsf{i} \wedge \mathsf{fx} = \mathsf{i}) \supset \mathsf{fx} = \mathsf{fy})$

$M_2 = \forall \mathsf{y.x.}(\mathsf{y} \leq \mathsf{m}^{\mathsf{x,y}})$ $\quad S = \forall \mathsf{x.y.}(\mathsf{sx} \leq \mathsf{y} \supset \mathsf{x} < \mathsf{y})$

Figure 4: Second normal form and Subproof Y_8.

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$1\leq 1 \vdash 1\leq 1$}
\AxiomC{}
\UnaryInfC{$0<1 \vdash 0<1$}
\RightLabel{$\supset_L$}
\BinaryInfC{$1\leq 1\supset 0<1,\ 1\leq 1 \vdash 0<1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(1\leq y\supset 0<y),\ 1\leq 1 \vdash 0<1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$1\leq 1,\ S \vdash 0<1$}
\AxiomC{}
\UnaryInfC{$f1=0 \vdash f1=0$}
\AxiomC{}
\UnaryInfC{$f0=0 \vdash f0=0$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$f1=0,\ f0=0 \vdash f1=0\wedge f0=0$}
\AxiomC{}
\UnaryInfC{$f0=f1 \vdash f0=f1$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f1=0\wedge f0=0)\supset f0=f1,\ f1=0,\ f0=0 \vdash f0=f1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=0\wedge f0=0)\supset f0=fy),\ f1=0,\ f0=0 \vdash f0=f1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=0\wedge fx=0)\supset fx=fy),\ f1=0,\ f0=0 \vdash f0=f1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$f1=0,\ f0=0,\ T \vdash f0=f1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$f1=0,\ 1\leq 1,\ f0=0,\ S,\ T \vdash 0<1\wedge f0=f1$}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{Y_5}
\RightLabel{$\vee_L$}
\BinaryInfC{$f1=0\vee f1=1,\ 1\leq 1,\ f0=0,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$1\leq 1,\ f0=0,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(x\leq m^{x,0}),\ f0=0,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$f0=0,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\end{prooftree}
$$

$$\underbrace{\qquad\qquad}_{Y_6}$$

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$1\leq 2 \vdash 1\leq 2$}
\AxiomC{}
\UnaryInfC{$0<2 \vdash 0<2$}
\RightLabel{$\supset_L$}
\BinaryInfC{$1\leq 2\supset 0<2,\ 1\leq 2 \vdash 0<2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(1\leq y\supset 0<y),\ 1\leq 2 \vdash 0<2$}
\RightLabel{$\forall_L$}
\UnaryInfC{$1\leq 2,\ S \vdash 0<2$}
\AxiomC{}
\UnaryInfC{$f2=1 \vdash f2=1$}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{Y_6}
\AxiomC{}
\UnaryInfC{$f0=1 \vdash f0=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f0=0\vee f0=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(1<m\wedge f1=fm),\ f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\exists_R$}
\UnaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$f2=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f2=1\wedge f0=1, \exists n.m.(n<m\wedge fn=fm)$}
\AxiomC{}
\UnaryInfC{$f0=f2 \vdash f0=f2$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f2=1\wedge f0=1)\supset f0=f2,\ f2=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f0=f2, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=1\wedge f0=1)\supset f0=fy),\ f2=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f0=f2, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ f2=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f0=f2, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\forall_L$}
\UnaryInfC{$f2=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f0=f2, \exists n.m.(n<m\wedge fn=fm)$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$f2=1,\ 1\leq 2,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<2\wedge f0=f2, \exists n.m.(n<m\wedge fn=fm)$}
\end{prooftree}
$$

$$\underbrace{\qquad\qquad}_{Y_7}$$

$M_1 = \forall y.x.(x \leq m^{x,y})$ $\quad T = \forall i.x.y.((fy = i \wedge fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \leq m^{x,y})$ $\quad S = \forall x.y.(sx \leq y \supset x < y)$

Figure 5: Subproofs Y_6 and Y_7.

$$\dfrac{}{1\le 1 \vdash 1\le 1} \qquad \dfrac{}{0<1 \vdash 0<1}$$

$$\dfrac{1\le 1 \vdash 1\le 1 \qquad 0<1 \vdash 0<1}{1\le 1\supset 0<1,\ 1\le 1 \vdash 0<1}\supset_L$$

$$\dfrac{1\le 1\supset 0<1,\ 1\le 1 \vdash 0<1}{\forall y.(1\le y\supset 0<y),\ 1\le 1 \vdash 0<1}\forall_L$$

$$\dfrac{\forall y.(1\le y\supset 0<y),\ 1\le 1 \vdash 0<1}{1\le 1,\ S \vdash 0<1}\forall_L$$

$$\dfrac{\dfrac{}{f1=0 \vdash f1=0} \qquad \dfrac{}{f0=0 \vdash f0=0}}{f1=0,\ f0=0 \vdash f1=0\wedge f0=0}\wedge_R \qquad \dfrac{}{f0=f1 \vdash f0=f1}$$

$$\dfrac{f1=0,\ f0=0 \vdash f1=0\wedge f0=0 \qquad f0=f1 \vdash f0=f1}{(f1=0\wedge f0=0)\supset f0=f1,\ f1=0,\ f0=0 \vdash f0=f1}\supset_L$$

$$\dfrac{(f1=0\wedge f0=0)\supset f0=f1,\ f1=0,\ f0=0 \vdash f0=f1}{\forall y.((fy=0\wedge f0=0)\supset f0=fy),\ f1=0,\ f0=0 \vdash f0=f1}\forall_L$$

$$\dfrac{\forall y.((fy=0\wedge f0=0)\supset f0=fy),\ f1=0,\ f0=0 \vdash f0=f1}{\forall x.y.((fy=0\wedge fx=0)\supset fx=fy),\ f1=0,\ f0=0 \vdash f0=f1}\forall_L$$

$$\dfrac{\forall x.y.((fy=0\wedge fx=0)\supset fx=fy),\ f1=0,\ f0=0 \vdash f0=f1}{f1=0,\ f0=0,\ T \vdash f0=f1}\forall_L$$

$$\dfrac{1\le 1,\ S \vdash 0<1 \qquad f1=0,\ f0=0,\ T \vdash f0=f1}{f1=0,\ 1\le 1,\ f0=0,\ S,\ T \vdash 0<1\wedge f0=f1}\wedge_R \qquad \begin{array}{c}\vdots\\ Y_3\end{array}$$

$$\dfrac{f1=0,\ 1\le 1,\ f0=0,\ S,\ T \vdash 0<1\wedge f0=f1 \qquad Y_3}{f1=0\vee f1=1,\ 1\le 1,\ f0=0,\ 2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2, \exists n.m.(n<m\wedge fn=fm)}\vee_L$$

$$\dfrac{\cdots}{1\le 1,\ f0=0,\ 2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2, \exists n.m.(n<m\wedge fn=fm)}\forall_L$$

$$\dfrac{\cdots}{\forall x.(x\le m^{x,0}),\ f0=0,\ 2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2, \exists n.m.(n<m\wedge fn=fm)}\forall_L$$

$$\underbrace{\dfrac{\cdots}{f0=0,\ 2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2, \exists n.m.(n<m\wedge fn=fm)}\forall_L}_{Y_4}$$

$$\begin{array}{c}\vdots\\ Y_4\end{array} \qquad \dfrac{}{f0=1 \vdash f0=1}$$

$$\dfrac{Y_4 \qquad f0=1 \vdash f0=1}{f0=0\vee f0=1,\ 2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\vee_L$$

$$\dfrac{\cdots}{2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\forall_L$$

$$\dfrac{\cdots}{2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\exists_R$$

$$\dfrac{\cdots}{2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<3\wedge f1=f3,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\exists_R$$

$$\dfrac{\cdots}{2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(1<m\wedge f1=fm),\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\exists_R$$

$$\dfrac{\cdots}{2\le 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\exists_R$$

$$\dfrac{\cdots}{\forall x.(2\le m^{x,2}),\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\forall_L$$

$$\underbrace{\dfrac{\cdots}{f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\forall_L}_{Y_5}$$

$M_1 = \forall y.x.(x \le m^{x,y})$ $\qquad T = \forall i.x.y.((fy = i\wedge fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \le m^{x,y})$ $\qquad S = \forall x.y.(sx \le y \supset x < y)$

Figure 6: Subproofs Y_4 and Y_5.

$$
\underbrace{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{2\leq 3 \vdash 2\leq 3} \quad \dfrac{}{1<3 \vdash 1<3}}{2\leq 3\supset 1<3,\ 2\leq 3 \vdash 1<3}{\scriptstyle\supset L}}{\forall y.(2\leq y\supset 1<y),\ 2\leq 3 \vdash 1<3}{\scriptstyle\forall L}}{2\leq 3,\ S \vdash 1<3}{\scriptstyle\forall L} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f3=1 \vdash f3=1} \quad \dfrac{}{f1=1 \vdash f1=1}}{f3=1,\ f1=1 \vdash f3=1\wedge f1=1}{\scriptstyle\wedge R} \quad \dfrac{}{f1=f3 \vdash f1=f3}}{(f3=1\wedge f1=1)\supset f1=f3,\ f3=1,\ f1=1 \vdash f1=f3}{\scriptstyle\supset L}}{\forall y.((fy=1\wedge f1=1)\supset f1=fy),\ f3=1,\ f1=1 \vdash f1=f3}{\scriptstyle\forall L}}{\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ f3=1,\ f1=1 \vdash f1=f3}{\scriptstyle\forall L}}{f3=1,\ f1=1,\ T \vdash f1=f3}{\scriptstyle\forall L}}{f3=1,\ 2\leq 3,\ f1=1,\ S,\ T \vdash 1<3\wedge f1=f3}{\scriptstyle\wedge R}}_{Y_1}
$$

$$
\underbrace{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{2\leq 2 \vdash 2\leq 2} \quad \dfrac{}{1<2 \vdash 1<2}}{2\leq 2\supset 1<2,\ 2\leq 2 \vdash 1<2}{\scriptstyle\supset L}}{\forall y.(2\leq y\supset 1<y),\ 2\leq 2 \vdash 1<2}{\scriptstyle\forall L}}{2\leq 2,\ S \vdash 1<2}{\scriptstyle\forall L} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f2=1 \vdash f2=1} \quad \dfrac{}{f1=1 \vdash f1=1}}{f2=1,\ f1=1 \vdash f2=1\wedge f1=1}{\scriptstyle\wedge R} \quad \dfrac{}{f1=f2 \vdash f1=f2}}{(f2=1\wedge f1=1)\supset f1=f2,\ f2=1,\ f1=1 \vdash f1=f2}{\scriptstyle\supset L}}{\forall y.((fy=1\wedge f1=1)\supset f1=fy),\ f2=1,\ f1=1 \vdash f1=f2}{\scriptstyle\forall L}}{\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ f2=1,\ f1=1 \vdash f1=f2}{\scriptstyle\forall L}}{f2=1,\ f1=1,\ T \vdash f1=f2}{\scriptstyle\forall L}}{f2=1,\ 2\leq 2,\ f1=1,\ S,\ T \vdash 1<2\wedge f1=f2}{\scriptstyle\wedge R}}_{Y_2}
$$

$$
\underbrace{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{3\leq 3 \vdash 3\leq 3} \quad \dfrac{}{2<3 \vdash 2<3}}{3\leq 3\supset 2<3,\ 3\leq 3 \vdash 2<3}{\scriptstyle\supset L}}{\forall y.(3\leq y\supset 2<y),\ 3\leq 3 \vdash 2<3}{\scriptstyle\forall L}}{3\leq 3,\ S \vdash 2<3}{\scriptstyle\forall L} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f3=0 \vdash f3=0} \quad \dfrac{}{f2=0 \vdash f2=0}}{f3=0,\ f2=0 \vdash f3=0\wedge f2=0}{\scriptstyle\wedge R} \quad \dfrac{}{f2=f3 \vdash f2=f3}}{(f3=0\wedge f2=0)\supset f2=f3,\ f3=0,\ f2=0 \vdash f2=f3}{\scriptstyle\supset L}}{\forall y.((fy=0\wedge f2=0)\supset f2=fy),\ f3=0,\ f2=0 \vdash f2=f3}{\scriptstyle\forall L}}{\forall x.y.((fy=0\wedge fx=0)\supset fx=fy),\ f3=0,\ f2=0 \vdash f2=f3}{\scriptstyle\forall L}}{f3=0,\ f2=0,\ T \vdash f2=f3}{\scriptstyle\forall L}}{f3=0,\ 3\leq 3,\ f2=0,\ S,\ T \vdash 2<3\wedge f2=f3}{\scriptstyle\wedge R} \quad \begin{matrix}\vdots\\ Y_1\end{matrix}}{f3=0\vee f3=1,\ 3\leq 3,\ f2=0,\ 2\leq 3,\ f1=1,\ S,\ T \vdash 2<3\wedge f2=f3,\ 1<3\wedge f1=f3}{\scriptstyle\vee L}}{3\leq 3,\ f2=0,\ 2\leq 3,\ f1=1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 2<3\wedge f2=f3,\ 1<3\wedge f1=f3}{\scriptstyle\forall L}}{\forall x.(x\leq m^{x,2}),\ f2=0,\ 2\leq 3,\ f1=1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 2<3\wedge f2=f3,\ 1<3\wedge f1=f3}{\scriptstyle\forall L}}{f2=0,\ 2\leq 3,\ f1=1,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 2<3\wedge f2=f3,\ 1<3\wedge f1=f3}{\scriptstyle\forall L} \quad \begin{matrix}\vdots\\ Y_2\end{matrix}}{f2=0\vee f2=1,\ 2\leq 3,\ f1=1,\ 2\leq 2,\ f1=1,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 2<3\wedge f2=f3,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2}{\scriptstyle\vee L}}{2\leq 3,\ f1=1,\ 2\leq 2,\ f1=1,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 2<3\wedge f2=f3,\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2}{\scriptstyle\forall L}}{2\leq 3,\ f1=1,\ 2\leq 2,\ f1=1,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(2<m\wedge f2=fm),\ 1<3\wedge f1=f3,\ 1<2\wedge f1=f2}{\scriptstyle\exists R}}{2\leq 3,\ f1=1,\ 2\leq 2,\ f1=1,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<3\wedge f1=f3,\ 1<2\wedge f1=f2, \exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\exists R}}{\forall x.(2\leq m^{x,2}),\ f1=1,\ 2\leq 2,\ f1=1,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<3\wedge f1=f3,\ 1<2\wedge f1=f2, \exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\forall L}}{f1=1,\ 2\leq 2,\ f1=1,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<3\wedge f1=f3,\ 1<2\wedge f1=f2, \exists n.m.(n<m\wedge fn=fm)}{\scriptstyle\forall L}}_{Y_3}
$$

$M_1 = \forall y.x.(x \leq m^{x,y})$ $\quad$ $T = \forall i.x.y.((fy = i \wedge fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \leq m^{x,y})$ $\quad$ $S = \forall x.y.(sx \leq y \supset x < y)$

Figure 7: Subproofs Y_1, Y_2 and Y_3.

$$\dfrac{\dfrac{\overline{1\leq 1 \vdash 1\leq 1}}{\forall x.(x\leq m^{x,0}) \vdash 1\leq 1}\,\forall_L}{M_1 \vdash 1\leq 1}\,\forall_L$$

$$\dfrac{\dfrac{\dfrac{M_1 \vdash 1\leq 1 \qquad \overline{0<1 \vdash 0<1}}{1\leq 1\supset 0<1,\ M_1 \vdash 0<1}\,\supset_L}{\forall y.(1\leq y\supset 0<y),\ M_1 \vdash 0<1}\,\forall_L}{M_1,\ S \vdash 0<1}\,\forall_L$$

$$\dfrac{\dfrac{\overline{f1=0 \vdash f1=0} \qquad \overline{f1=1 \vdash f1=1}}{f1=0\vee f1=1 \vdash f1=0,\ f1=1}\,\vee_L}{\forall x.(fx=0\vee fx=1) \vdash f1=0,\ f1=1}\,\forall_L$$

$$\dfrac{\dfrac{\overline{f0=0 \vdash f0=0} \qquad \overline{f0=1 \vdash f0=1}}{f0=0\vee f0=1 \vdash f0=0,\ f0=1}\,\vee_L}{\forall x.(fx=0\vee fx=1) \vdash f0=0,\ f0=1}\,\forall_L$$

$$\dfrac{\forall x.(fx=0\vee fx=1) \vdash f1=0,\ f1=1 \qquad \forall x.(fx=0\vee fx=1) \vdash f0=0,\ f0=1}{\forall x.(fx=0\vee fx=1) \vdash f1=0\wedge f0=0,\ f1=1,\ f0=1}\,\wedge_R$$

$$\dfrac{\forall x.(fx=0\vee fx=1) \vdash f1=0\wedge f0=0,\ f1=1,\ f0=1 \qquad \overline{f0=f1 \vdash f0=f1}}{(f1=0\wedge f0=0)\supset f0=f1, \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1}\,\supset_L$$

$$\dfrac{\dfrac{\dfrac{(f1=0\wedge f0=0)\supset f0=f1, \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1}{\forall y.((fy=0\wedge f0=0)\supset f0=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1}\,\forall_L}{\forall x.y.((fy=0\wedge fx=0)\supset fx=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1}\,\forall_L}{\forall x.(fx=0\vee fx=1),\ T \vdash f0=f1,\ f1=1,\ f0=1}\,\forall_L$$

$$\dfrac{M_1,\ S \vdash 0<1 \qquad \forall x.(fx=0\vee fx=1),\ T \vdash f0=f1,\ f1=1,\ f0=1}{M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=1,\ 0<1\wedge f0=f1,\ f0=1}\,\wedge_R$$

$$\underbrace{\dfrac{\begin{matrix}\vdots\\ Y_{11}\end{matrix} \qquad M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=1,\ 0<1\wedge f0=f1,\ f0=1}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f2=1\wedge f1=1,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}\,\wedge_R}_{Y_{12}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{2\leq 2 \vdash 2\leq 2}}{\forall x.(2\leq m^{x,2}) \vdash 2\leq 2}\,\forall_L}{M_2 \vdash 2\leq 2}\,\forall_L \qquad \overline{1<2 \vdash 1<2}}{2\leq 2\supset 1<2,\ M_2 \vdash 1<2}\,\supset_L}{\dfrac{\forall y.(2\leq y\supset 1<y),\ M_2 \vdash 1<2}{M_2,\ S \vdash 1<2}\,\forall_L}\,\forall_L$$

$$\dfrac{\begin{matrix}\vdots\\ Y_{12}\end{matrix} \qquad \overline{f1=f2 \vdash f1=f2}}{(f2=1\wedge f1=1)\supset f1=f2,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}\,\supset_L$$

$$\dfrac{(f2=1\wedge f1=1)\supset f1=f2,\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}{\forall y.((fy=1\wedge f1=1)\supset f1=fy),\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}\,\forall_L$$

$$\dfrac{\forall y.((fy=1\wedge f1=1)\supset f1=fy),\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}{\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}\,\forall_L$$

$$\dfrac{\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}\,\forall_L$$

$$\dfrac{M_2,\ S \vdash 1<2 \qquad M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f2,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ f0=1}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1}\,\wedge_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(2<m\wedge f2=fm),\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(2<m\wedge f2=fm),\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<3\wedge f1=f3,\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(1<m\wedge f1=fm),\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(1<m\wedge f1=fm),\ 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 0<1\wedge f0=f1,\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(0<m\wedge f0=fm),\ 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash 1<2\wedge f1=f2,\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(1<m\wedge f1=fm),\ f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R$$

$$\underbrace{\dfrac{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash \exists m.(1<m\wedge f1=fm),\ f0=1, \exists n.m.(n<m\wedge fn=fm)}{M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f0=1, \exists n.m.(n<m\wedge fn=fm)}\,\exists_R}_{Y_{13}}$$

$M_1 = \forall y.x.(x \leq m^{x,y})$ $\qquad T = \forall i.x.y.((fy = i \wedge fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \leq m^{x,y})$ $\qquad S = \forall x.y.(sx \leq y \supset x < y)$

Figure 8: Subproofs Y_{12} and Y_{13}.

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$3\leq 3 \vdash 3\leq 3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(x\leq m^{x,2}) \vdash 3\leq 3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1 \vdash 3\leq 3$}
\AxiomC{}
\UnaryInfC{$2<3 \vdash 2<3$}
\RightLabel{$\supset_L$}
\BinaryInfC{$3\leq 3\supset 2<3,\ M_1 \vdash 2<3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(3\leq y\supset 2<y),\ M_1 \vdash 2<3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1,\ S \vdash 2<3$}
\AxiomC{}
\UnaryInfC{$f3=0 \vdash f3=0$}
\AxiomC{}
\UnaryInfC{$f3=1 \vdash f3=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f3=0\vee f3=1 \vdash f3=0,\ f3=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f3=0,\ f3=1$}
\AxiomC{}
\UnaryInfC{$f2=0 \vdash f2=0$}
\AxiomC{}
\UnaryInfC{$f2=1 \vdash f2=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f2=0\vee f2=1 \vdash f2=0,\ f2=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f2=0,\ f2=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f3=0\wedge f2=0,\ f3=1,\ f2=1$}
\AxiomC{}
\UnaryInfC{$f2=f3 \vdash f2=f3$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f3=0\wedge f2=0)\supset f2=f3, \forall x.(fx=0\vee fx=1) \vdash f2=f3,\ f3=1,\ f2=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=0\wedge f2=0)\supset f2=fy), \forall x.(fx=0\vee fx=1) \vdash f2=f3,\ f3=1,\ f2=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=0\wedge fx=0)\supset fx=fy), \forall x.(fx=0\vee fx=1) \vdash f2=f3,\ f3=1,\ f2=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1),\ T \vdash f2=f3,\ f3=1,\ f2=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f3=1,\ f2=1,\ 2<3\wedge f2=f3$}
\end{prooftree}
$$

Y_9

$$
\begin{prooftree}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{Y_9}
\AxiomC{}
\UnaryInfC{$1\leq 1 \vdash 1\leq 1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(x\leq m^{x,0}) \vdash 1\leq 1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1 \vdash 1\leq 1$}
\AxiomC{}
\UnaryInfC{$0<1 \vdash 0<1$}
\RightLabel{$\supset_L$}
\BinaryInfC{$1\leq 1\supset 0<1,\ M_1 \vdash 0<1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(1\leq y\supset 0<y),\ M_1 \vdash 0<1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1,\ S \vdash 0<1$}
\AxiomC{}
\UnaryInfC{$f1=0 \vdash f1=0$}
\AxiomC{}
\UnaryInfC{$f1=1 \vdash f1=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f1=0\vee f1=1 \vdash f1=0,\ f1=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f1=0,\ f1=1$}
\AxiomC{}
\UnaryInfC{$f0=0 \vdash f0=0$}
\AxiomC{}
\UnaryInfC{$f0=1 \vdash f0=1$}
\RightLabel{$\vee_L$}
\BinaryInfC{$f0=0\vee f0=1 \vdash f0=0,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f0=0,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$\forall x.(fx=0\vee fx=1) \vdash f1=0\wedge f0=0,\ f1=1,\ f0=1$}
\AxiomC{}
\UnaryInfC{$f0=f1 \vdash f0=f1$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f1=0\wedge f0=0)\supset f0=f1, \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=0\wedge f0=0)\supset f0=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=0\wedge fx=0)\supset fx=fy), \forall x.(fx=0\vee fx=1) \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(fx=0\vee fx=1),\ T \vdash f0=f1,\ f1=1,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=1,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f3=1\wedge f1=1,\ f2=1,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ f0=1$}
\end{prooftree}
$$

Y_{10}

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$2\leq 3 \vdash 2\leq 3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.(2\leq m^{x,2}) \vdash 2\leq 3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_2 \vdash 2\leq 3$}
\AxiomC{}
\UnaryInfC{$1<3 \vdash 1<3$}
\RightLabel{$\supset_L$}
\BinaryInfC{$2\leq 3\supset 1<3,\ M_2 \vdash 1<3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.(2\leq y\supset 1<y),\ M_2 \vdash 1<3$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_2,\ S \vdash 1<3$}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{Y_{10}}
\AxiomC{}
\UnaryInfC{$f1=f3 \vdash f1=f3$}
\RightLabel{$\supset_L$}
\BinaryInfC{$(f3=1\wedge f1=1)\supset f1=f3,\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f3,\ f2=1,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall y.((fy=1\wedge f1=1)\supset f1=fy),\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f3,\ f2=1,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$\forall x.y.((fy=1\wedge fx=1)\supset fx=fy),\ M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f3,\ f2=1,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\forall_L$}
\UnaryInfC{$M_1, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f1=f3,\ f2=1,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ f0=1$}
\RightLabel{$\wedge_R$}
\BinaryInfC{$M_1,\ M_2, \forall x.(fx=0\vee fx=1),\ S,\ T \vdash f2=1,\ 2<3\wedge f2=f3,\ 0<1\wedge f0=f1,\ 1<3\wedge f1=f3,\ f0=1$}
\end{prooftree}
$$

Y_{11}

$M_1 = \forall y.x.(x \leq m^{x,y})$ $\quad T = \forall i.x.y.((fy = i \wedge fx = i) \supset fx = fy)$

$M_2 = \forall y.x.(y \leq m^{x,y})$ $\quad S = \forall x.y.(sx \leq y \supset x < y)$

Figure 9: Subproofs Y_9, Y_{10} and Y_{11}.

Appendix B

Details for some Proofs

B.1 Proofs of Chapter 2

In this section we are mainly concerned with proving strong normalisation for $(\mathcal{T}, \xrightarrow{cut})$, $(\mathcal{T}, \xrightarrow{aux})$ and $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$.

Proof of Proposition 2.2.10. One case is as follows.

Case $M \equiv \mathsf{Cut}(\langle a\rangle S, (x)T)$: Suppose $\Gamma_1 \triangleright S \triangleright \Delta_1$ and $x : B, \Gamma_2 \triangleright T \triangleright \Delta_2$ are typing judgements for S and T, respectively. Furthermore, assume that a is not free in S. By the inference rules we have for the term M the typing judgement

$$\Gamma_1, \Gamma_2 \triangleright \mathsf{Cut}(\langle a\rangle S, (x)T) \triangleright \Delta_1, \Delta_2 \ .$$

Let $\mathsf{Cut}(\langle a\rangle S, (x)T)$ reduce to $S[a := (x)T]$, which, because we assumed that a is not free in S, is equivalent to S. From our assumption that $\Gamma_1 \triangleright S \triangleright \Delta_1$ is a typing judgement for S, we have by Lemma 2.2.9 that $\Gamma_1, \Gamma_2 \triangleright S \triangleright \Delta_1, \Delta_2$ is also a typing judgement for S. Thus we are done. $\square$

Proof of Lemma 2.3.1. The only case that is non-trivial is where M is an axiom (since axioms introduce two labels).

Case $M \equiv \mathsf{Ax}(x, b)$**:** Suppose $\{\sigma\}$ and $\{\tau\}$ are of the form $\{x := \langle a\rangle P\}$ and $\{b := (y)Q\}$, respectively. We analyse in turn the cases $\mathsf{Ax}(x, b)\{\sigma\}\{\tau\}$ and $\mathsf{Ax}(x, b)\{\tau\}\{\sigma\{\tau\}\}$.

$$\begin{aligned} M\{\sigma\}\{\tau\} &\equiv \mathsf{Ax}(x,b)\{x := \langle a\rangle P\}\{b := (y)Q\} \\ &= \mathsf{Cut}(\langle a\rangle P, (x)\mathsf{Ax}(x,b))\{b := (y)Q\} \\ &= \mathsf{Cut}(\langle a\rangle P\{b := (y)Q\}, (y)Q) \end{aligned}$$

$$\begin{aligned} M\{\tau\}\{\sigma\{\tau\}\} &\equiv \mathsf{Ax}(x,b)\{b := (y)Q\}\{x := \langle a\rangle P\{b := (y)Q\}\} \\ &= \mathsf{Cut}(\langle b\rangle \mathsf{Ax}(x,b), (y)Q)\{x := \langle a\rangle P\{b := (y)Q\}\} \\ &= \mathsf{Cut}(\langle a\rangle P\{b := (y)Q\}, (y)Q\{x := \langle a\rangle P\{b := (y)Q\}\}) \\ &\overset{(*)}{\equiv} \mathsf{Cut}(\langle a\rangle P\{b := (y)Q\}, (y)Q) \end{aligned}$$

$^{(*)}$ because by assumption $x \notin FN((y)Q)$

$\square$

Proof of Lemma 2.3.11. We illustrate the proof with two cases. Somewhat surprisingly, it is possible that $M\{\sigma\} \equiv M'\{\sigma\}$, as shown in the first case.

Case $M \equiv \mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y,a), (x)S)$**:** Let S freshly introduce x and suppose $\{\sigma\}$ is of the form $\{y := \langle c\rangle T\}$. Furthermore, assume $M \xrightarrow{aux} M'$ with $M' \equiv S[x \mapsto y]$. Obviously, we have $M'\{\sigma\} \equiv S[x\mapsto y]\{y := \langle c\rangle T\}$. In the following calculation, the equivalence ('0'-case) occurs if $S\{y := \langle c\rangle T\}$ freshly introduces x.

$$\begin{array}{rcl} M\{\sigma\} & \equiv & \mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y,a), (x)S)\{y := \langle c\rangle T\} \\ & = & \mathsf{Cut}(\langle c\rangle T, (x)S\{y := \langle c\rangle T\}) \\ & \xrightarrow{aux}{}^{0/1} & S\{y := \langle c\rangle T\}\{x := \langle c\rangle T\} \\ & \stackrel{(*)}{\equiv} & S[x\mapsto y]\{y := \langle c\rangle T\} \end{array}$$

$(*)$ because by Barendregt-style naming convention $x \notin FN(\langle c\rangle T)$

Case $M \equiv \mathsf{Cut}(\langle a\rangle S, (x)T)$**:** Suppose the term S does not freshly introduce a and assume $M \xrightarrow{aux} M'$ with $M' \equiv S\{a := \langle x\rangle T\}$.

$$\begin{array}{rcl} M\{\sigma\} & \equiv & \mathsf{Cut}(\langle a\rangle S, (x)T)\{\sigma\} \\ & = & \mathsf{Cut}(\langle a\rangle S\{\sigma\}, (x)T\{\sigma\}) \\ & \xrightarrow{aux} & S\{\sigma\}\{a := (x)T\{\sigma\}\} \\ & \stackrel{(*)}{\equiv} & S\{a := \langle x\rangle T\}\{\sigma\} \end{array}$$

$(*)$ by Barendregt-style naming convention $a \notin FC(dom(\{\sigma\}))$ and hence by Lem. 2.3.1

□

Proof of Lemma 2.3.12. From Lemma 2.3.6(i), it follows that if $\langle a{:}B\rangle M \in [\![\langle B\rangle]\!]$, then $\langle a{:}B\rangle M \in \text{NEG}_{\langle B\rangle}([\![(B)]\!])$. So depending on the structure of B we need to analyse all possible sets (generated by the set operators) where $\langle a{:}B\rangle M$ may be member in. Six cases are given for (i); the arguments for (ii) are similar and omitted.

Case $\text{AXIOMS}_{\langle B\rangle}$: $\langle a{:}B\rangle M$ cannot be in $\text{AXIOMS}_{\langle B\rangle}$ because axioms do not reduce.

Case $\text{BINDING}_{\langle B\rangle}([\![(B)]\!])$:

(1)	$\langle a{:}B\rangle M \in \text{BINDING}_{\langle B\rangle}([\![(B)]\!])$ and $M \xrightarrow{aux} M'$	by assumption
(2)	$M\{a := (x{:}B)P\} \in SN_{aux}$ for all $(x{:}B)P \in [\![(B)]\!]$	by Definition 2.3.3
(3)	$M\{a := (x{:}B)P\} \xrightarrow{aux}{}^{0/1} M'\{a := (x{:}B)P\}$	by Lemma 2.3.11
(4)	$M'\{a := (x{:}B)P\} \in SN_{aux}$ for all $(x{:}B)P \in [\![(B)]\!]$	by (2) and (3)
(5)	$\langle a{:}B\rangle M' \in \text{BINDING}_{\langle B\rangle}([\![(B)]\!])$	by Definition 2.3.3
(6)	$\langle a{:}B\rangle M' \in [\![\langle B\rangle]\!]$	by Definition 2.3.4

Case $\textsc{notright}_{\langle\neg C\rangle}(\llbracket(C)\rrbracket)$, $B \equiv \neg C$:

(1)	$M \equiv \mathsf{Not}_R((x)S, a),\ \ M' \equiv \mathsf{Not}_R((x)S', a),\ \ S \xrightarrow{aux} S'$ and $\langle a{:}\neg C\rangle M \in \textsc{notright}_{\langle\neg C\rangle}(\llbracket(C)\rrbracket)$	by assumption
(2)	M freshly introduces a and $(x{:}C)S \in \llbracket(C)\rrbracket$	by Definition 2.3.3
(3)	M' freshly introduces a	by Lemma 2.3.8
(4)	$(x{:}C)S' \in \llbracket(C)\rrbracket$	by induction
(5)	$\langle a{:}\neg C\rangle M' \in \textsc{notright}_{\langle\neg C\rangle}(\llbracket(C)\rrbracket)$	by (3), (4) and Definition 2.3.3
(6)	$\langle a{:}\neg C\rangle M' \in \llbracket\langle\neg C\rangle\rrbracket$	by Definition 2.3.4

Case $\textsc{andright}_{\langle C\wedge D\rangle}(\llbracket\langle C\rangle\rrbracket, \llbracket\langle D\rangle\rrbracket)$, $B \equiv C\wedge D$:

(1)	$M \equiv \mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a),\ \ M' \equiv \mathsf{And}_R(\langle d\rangle S', \langle e\rangle T', a)$ and $\langle a{:}C\wedge D\rangle M \in \textsc{andright}_{\langle C\wedge D\rangle}(\llbracket\langle C\rangle\rrbracket, \llbracket\langle D\rangle\rrbracket)$	by assumption
(2)	$S \xrightarrow{aux} S'$ and $T \equiv T'$ (the other case being similar)	new assumption
(3)	M freshly introduces a, $\langle d{:}C\rangle S \in \llbracket\langle C\rangle\rrbracket$ and $\langle e{:}D\rangle T \in \llbracket\langle D\rangle\rrbracket$	by Def. 2.3.3
(4)	M' freshly introduces a	by Lemma 2.3.8
(5)	$\langle d{:}C\rangle S' \in \llbracket\langle C\rangle\rrbracket$	by induction
(6)	$\langle a{:}C\wedge D\rangle M' \in \textsc{andright}_{\langle C\wedge D\rangle}(\llbracket\langle C\rangle\rrbracket, \llbracket\langle D\rangle\rrbracket)$	by (4), (5) and Def. 2.3.3
(7)	$\langle a{:}C\wedge D\rangle M' \in \llbracket\langle C\wedge D\rangle\rrbracket$	by Definition 2.3.4

Case $\textsc{orright}^i_{\langle C_1\vee C_2\rangle}(\llbracket\langle C_i\rangle\rrbracket)$, $B \equiv C_1\vee C_2$, $(i = 1, 2)$:

(1)	$M \equiv \mathsf{Or}^i_R(\langle d\rangle S, a),\ \ M' \equiv \mathsf{Or}^i_R(\langle d\rangle S', a),\ \ S \xrightarrow{aux} S'$ and $\langle a{:}C_1\vee C_2\rangle M \in \textsc{orright}^i_{\langle C_1\vee C_2\rangle}(\llbracket\langle C_i\rangle\rrbracket)$	by assumption
(2)	M freshly introduces a and $\langle d{:}C_i\rangle S \in \llbracket\langle C_i\rangle\rrbracket$	by Definition 2.3.3
(3)	M' freshly introduces a	by Lemma 2.3.8
(4)	$\langle d{:}C_i\rangle S' \in \llbracket\langle C_i\rangle\rrbracket$	by induction
(5)	$\langle a{:}C_1\vee C_2\rangle M' \in \textsc{orright}^i_{\langle C_1\wedge C_2\rangle}(\llbracket\langle C_i\rangle\rrbracket)$	by (3), (4) and Definition 2.3.3
(6)	$\langle a{:}C_1\vee C_2\rangle M' \in \llbracket\langle C_1\vee C_2\rangle\rrbracket$	by Definition 2.3.4

Case $\textsc{impright}_{\langle C\supset D\rangle}(\llbracket(C)\rrbracket, \llbracket\langle D\rangle\rrbracket)$, $B \equiv C\supset D$:

(1)	$M \equiv \mathsf{Imp}_R((x)\langle d\rangle S, a),\ \ M' \equiv \mathsf{Imp}_R((x)\langle d\rangle S', a),\ \ S \xrightarrow{aux} S'$ and $\langle a{:}C\supset D\rangle M \in \textsc{impright}_{\langle C\supset C\rangle}(\llbracket(C)\rrbracket, \llbracket\langle D\rangle\rrbracket)$	by assumption
(2)	M freshly introduces a, $\langle d{:}D\rangle S\{x := \langle e{:}C\rangle Q\} \in \llbracket\langle D\rangle\rrbracket$ for all $\langle e{:}C\rangle Q \in \llbracket\langle C\rangle\rrbracket$, and $(x{:}C)S\{d := (z{:}D)P\} \in \llbracket(C)\rrbracket$ for all $(z{:}D)P \in \llbracket(D)\rrbracket$	by Definition 2.3.3
(3)	M' freshly introduces a	by Lemma 2.3.8
(4)	$S\{x := \langle e{:}C\rangle Q\} \xrightarrow{aux}{}^{0/1} S'\{x := \langle e{:}C\rangle Q\}$, $S\{d := (z{:}D)P\} \xrightarrow{aux}{}^{0/1} S'\{d := (z{:}D)P\}$	by Lemma 2.3.11
(5)	$\langle d{:}D\rangle S'\{x := \langle e{:}C\rangle Q\} \in \llbracket\langle D\rangle\rrbracket$ for all $\langle e{:}C\rangle Q \in \llbracket\langle C\rangle\rrbracket$, $(x{:}C)S'\{d := (z{:}D)P\} \in \llbracket(C)\rrbracket$ for all $(z{:}D)P \in \llbracket(D)\rrbracket$	by (2) and (4): '0'-case trivial, '1'-case by induction
(6)	$\langle a{:}C\supset D\rangle M' \in \textsc{impright}_{\langle C\supset C\rangle}(\llbracket(C)\rrbracket, \llbracket\langle D\rangle\rrbracket)$	by (3), (5) and Def. 2.3.3
(7)	$\langle a{:}C\supset D\rangle M' \in \llbracket\langle C\supset D\rangle\rrbracket$	by Definition 2.3.4

□

Proof of Lemma 2.3.13. Analogous to the proof of the preceding lemma, we need to analyse all possible sets where $\langle a{:}B\rangle M$ may be member in. Six cases are given for (i); the arguments for (ii) are similar and omitted.

Case $\text{AXIOMS}_{\langle B\rangle}$: In this case M is an axiom and therefore strongly normalising.

Case $\text{BINDING}_{\langle B\rangle}(\llbracket(B)\rrbracket)$:

(1)	$\langle a{:}B\rangle M \in \text{BINDING}_{\langle B\rangle}(\llbracket(B)\rrbracket)$	by assumption
(2)	$M\{a := (x{:}B)P\} \in SN_{aux}$ for all $(x{:}B)P \in \llbracket(B)\rrbracket$	by Definition 2.3.3
(3)	$(x{:}B)\mathsf{Ax}(x,a) \in \llbracket(B)\rrbracket$	by Lemma 2.3.6(ii)
(4)	$M\{a := (x{:}B)\mathsf{Ax}(x,a)\} \in SN_{aux}$	by (2), (3) and $P \equiv \mathsf{Ax}(x,a)$
(5)	$M\{a := (x{:}B)\mathsf{Ax}(x,a)\} \xrightarrow{aux}{}^{*} M$	by Lemma 2.3.9
(6)	$M \in SN_{aux}$	by (4) and (5)

Case $\text{NOTRIGHT}_{\langle\neg C\rangle}(\llbracket(C)\rrbracket)$, $B \equiv \neg C$:

(1)	$M \equiv \mathsf{Not}_R((x)S,a)$, $\langle a{:}\neg C\rangle M \in \text{NOTRIGHT}_{\langle\neg C\rangle}(\llbracket(C)\rrbracket)$	by assumption
(2)	$(x{:}C)S \in \llbracket(C)\rrbracket$	by Definition 2.3.3
(3)	$S \in SN_{aux}$	by induction
(4)	$\mathsf{Not}_R((x)S,a) \in SN_{aux}$	by (3)

Case $\text{ANDRIGHT}_{\langle C\wedge D\rangle}(\llbracket\langle C\rangle\rrbracket,\llbracket\langle D\rangle\rrbracket)$, $B \equiv C\wedge D$:

(1)	$M \equiv \mathsf{And}_R(\langle d\rangle S,\langle e\rangle T,a)$, $\langle a{:}C\wedge D\rangle M \in \text{ANDRIGHT}_{\langle C\wedge D\rangle}(\llbracket\langle C\rangle\rrbracket,\llbracket\langle D\rangle\rrbracket)$	by assumption
(2)	$\langle d{:}C\rangle S \in \llbracket\langle C\rangle\rrbracket$ and $\langle e{:}D\rangle T \in \llbracket\langle D\rangle\rrbracket$	by Definition 2.3.3
(3)	$S,T \in SN_{aux}$	by induction
(4)	$\mathsf{And}_R(\langle d\rangle S,\langle e\rangle T,a) \in SN_{aux}$	by (3)

Case $\text{ORRIGHT}^{i}_{\langle C_1\vee C_2\rangle}(\llbracket\langle C_i\rangle\rrbracket)$, $B \equiv C_1\vee C_2$, $(i=1,2)$:

(1)	$M \equiv \mathsf{Or}^i_R(\langle d\rangle S,a)$, $\langle a{:}C_1\vee C_2\rangle M \in \text{ORRIGHT}^{i}_{\langle C_1\vee C_2\rangle}(\llbracket\langle C_i\rangle\rrbracket)$	by assumption
(2)	$\langle d{:}C_i\rangle S \in \llbracket\langle C_i\rangle\rrbracket$	by Definition 2.3.3
(3)	$S \in SN_{aux}$	by induction
(4)	$\mathsf{Or}^i_R(\langle d\rangle S,a) \in SN_{aux}$	by (3)

Case $\text{IMPRIGHT}_{\langle C\supset D\rangle}(\llbracket(C)\rrbracket,\llbracket\langle D\rangle\rrbracket)$, $B \equiv C\supset D$:

(1)	$M \equiv \mathsf{Imp}_R((x)\langle d\rangle S,a)$, $\langle a{:}C\supset D\rangle M \in \text{IMPRIGHT}_{\langle C\supset C\rangle}(\llbracket(C)\rrbracket,\llbracket\langle D\rangle\rrbracket)$	by assumption
(2)	$(x{:}C)S\{d := (z{:}D)P\} \in \llbracket(C)\rrbracket$ for all $(z{:}D)P \in \llbracket(D)\rrbracket$	by Definition 2.3.3
(3)	$(z{:}D)\mathsf{Ax}(z,d) \in \llbracket(D)\rrbracket$	by Lemma 2.3.6(ii)
(4)	$(x{:}C)S\{d := (z{:}D)\mathsf{Ax}(z,d)\} \in \llbracket(C)\rrbracket$	by (2), (3) and $P \equiv \mathsf{Ax}(z,d)$
(5)	$S\{d := (z{:}D)\mathsf{Ax}(z,d)\} \in SN_{aux}$	by induction
(6)	$S\{d := (z{:}D)\mathsf{Ax}(z,d)\} \xrightarrow{aux}{}^{*} S$	by Lemma 2.3.9
(7)	$S \in SN_{aux}$	by (5) and (6)
(8)	$\mathsf{Imp}_R((x)\langle d\rangle S,a) \in SN_{aux}$	by (7)

$\square$

Proof of Lemma 2.3.14. The induction proceeds over the lexicographically ordered induction value (δ, μ, ν) assigned to each term of the form $\mathsf{Cut}(\langle a{:}B\rangle M, (x{:}B)N)$; δ is the degree of the cut-formula B, and μ and ν are the longest reduction sequences (relative to $\xrightarrow{aux}$) starting from M and N, respectively. By assumption both μ and ν are finite. We shall show that every immediate reduct to which $\mathsf{Cut}(\langle a\rangle M, (x)N)$ reduces is strongly normalising.

First, we shall give one case where an inner reduction occurs.

Inner Reduction:

(1)	$\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{aux} \mathsf{Cut}(\langle a\rangle M', (x)N)$,	
	$\langle a{:}B\rangle M \in [\![\langle B\rangle]\!]$ and $(x{:}B)N \in [\![(B)]\!]$	by assumption
(2)	$M \xrightarrow{aux} M'$	by (1)
(3)	$\langle a{:}B\rangle M' \in [\![\langle B\rangle]\!]$	by Lemma 2.3.12
(4)	$M' \in SN_{aux}$	by Lemma 2.3.13
(5)	$\mathsf{Cut}(\langle a\rangle M', (x)N) \in SN_{aux}$	by induction,
		the degree of the cut-formula is equal in both terms, but $l(M') < l(M)$

Next, we shall give one case where a commuting reduction occurs on the top-level.

Commuting Reduction:

(1) $\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{c'} M\{a := (x)N\}$ and $\langle a{:}B\rangle M \in [\![\langle B\rangle]\!]$ by assumption

We know that only the commuting reduction is applicable, if M does not freshly introduce a. This implies that there are only two possibilities for $\langle a{:}B\rangle M$ to be in $[\![\langle B\rangle]\!]$: it can be in $\text{AXIOMS}_{\langle B\rangle}$ or in $\text{BINDING}_{\langle B\rangle}([\![(B)]\!])$.

In the first case M is an axiom that does not introduce a. Thus $M\{a := (x)N\}$ is equivalent to M, which we know by assumption is strongly normalising.

The proof for the second case is as follows.

(2)	$\langle a{:}B\rangle M \in \text{BINDING}_{\langle B\rangle}([\![(B)]\!])$	new assumption
(3)	$M\{a := (y{:}B)P\} \in SN_{aux}$ for all $(y{:}B)P \in [\![(B)]\!]$	by Definition 2.3.3
(4)	$(x{:}B)N \in [\![(B)]\!]$	by assumption
(5)	$M\{a := (x{:}B)N\} \in SN_{aux}$	by (3), (4) and $(y)P \equiv (x)N$

It remains to check for every logical cut-reduction rule that the immediate reducts of $\mathsf{Cut}(\langle a\rangle M, (x)N)$ are strongly normalising. We shall give three cases to illustrate the proof. The difficult case is the logical reduction $\supset_R/\supset_L$, because both immediate reducts have two nested cuts.

Logical Reduction with Axioms: $\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(y, a), (x)N)$ reduces to $N[x \mapsto y]$. By assumption we know that N is strongly normalising, and therefore $N[x \mapsto y]$ must be strongly normalising.

Logical Reduction $\wedge_R/\wedge_{L_1}$, $B \equiv C\wedge D$:

(1) $\mathsf{Cut}(\langle c\rangle M,(y)N) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle S,(x)U)$,
$M \equiv \mathsf{And}_R(\langle a\rangle S,\langle b\rangle T,c)$, $N \equiv \mathsf{And}^1_L((x)U,y)$,
M and N freshly introduce c and y, respectively,
$\langle c{:}C\wedge D\rangle M \in [\![\langle C\wedge D\rangle]\!]$ and $(y{:}C\wedge D)N \in [\![(C\wedge D)]\!]$ by assumption

Since M and N are not axioms, it follows from Lemma 2.3.6(i) that

$\langle c{:}C\wedge D\rangle M \in \text{BINDING}_{\langle C\wedge D\rangle}([\![(C\wedge D)]\!]) \;\cup\; \text{ANDRIGHT}_{\langle C\wedge D\rangle}([\![\langle C\rangle]\!],[\![\langle D\rangle]\!])$
$(y{:}C\wedge D)N \in \text{BINDING}_{(C\wedge D)}([\![\langle C\wedge D\rangle]\!]) \;\cup\; \text{ANDLEFT}^1_{(C\wedge D)}([\![(C)]\!])$

Now our argument splits into two cases depending on whether at least one of the $\langle c{:}C\wedge D\rangle M$ and $(y{:}C\wedge D)N$ belong to BINDING. Let us assume $\langle c{:}C\wedge D\rangle M$ is an element in $\text{BINDING}_{\langle C\wedge D\rangle}([\![(C\wedge D)]\!])$.

(2.1) $\langle c{:}C\wedge D\rangle M \in \text{BINDING}_{\langle C\wedge D\rangle}([\![(C\wedge D)]\!])$ new assumption
(2.2) $M\{c := (z)P\} \in SN_{aux}$ for all $(z{:}C\wedge D)P \in [\![(C\wedge D)]\!]$ by Def. 2.3.3
(2.3) $M\{c := (y)N\} \in SN_{aux}$ by (1), (2.2) and $(z)P \equiv (y)N$
(2.4) The following calculation shows that $\mathsf{Cut}(\langle c\rangle M,(y)N) \in SN_{aux}$.

$$\begin{aligned} M\{c := (y)N\} &\equiv \mathsf{And}_R(\langle a\rangle S,\langle b\rangle T,c)\{c := (y)N\}\\ &= \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S\{c := (y)N\},\langle b\rangle T\{c := (y)N\},c),(y)N)\\ &\stackrel{(*)}{\equiv} \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S,\langle b\rangle T,c),(y)N)\\ &\equiv \mathsf{Cut}(\langle c\rangle M,(y)N) \end{aligned}$$

$(*)$ because M freshly introduces c.

If $\mathsf{Cut}(\langle c\rangle M,(y)N)$ is strongly normalising, then its reduct $\mathsf{Cut}(\langle a\rangle S,(x)U)$ must be strongly normalising too. In case $(y{:}C\wedge D)N$ is in $\text{BINDING}_{(C\wedge D)}([\![\langle C\wedge D\rangle]\!])$, we reason analogous.

If neither $\langle c{:}C\wedge D\rangle M$ nor $(y{:}C\wedge D)N$ are in BINDING, then we proceed as follows.

(3.1) $\langle c{:}C\wedge D\rangle M \in \text{ANDRIGHT}_{\langle C\wedge D\rangle}([\![\langle C\rangle]\!],[\![\langle D\rangle]\!])$,
$(y{:}C\wedge D)N \in \text{ANDLEFT}^1_{(C\wedge D)}([\![(C)]\!])$ new assumption
(3.2) $\langle a : C\rangle S \in [\![\langle C\rangle]\!]$ and $(x : C)U \in [\![(C)]\!]$ by Definition 2.3.3
(3.3) $S, U \in SN_{aux}$ by Lemma 2.3.13
(3.4) $\mathsf{Cut}(\langle a{:}C\rangle S,(x{:}C)U) \in SN_{aux}$ by induction (the degree decreased)

Logical Reduction $\supset_R/\supset_L$, $B \equiv C\supset D$:

(1) $M \equiv \mathsf{Imp}_R((x)\langle a\rangle S,b)$, $N \equiv \mathsf{Imp}_L(\langle c\rangle T,(y)U,z)$,
M and N freshly introduce b and z, respectively,
$\langle c{:}C\supset D\rangle M \in [\![\langle C\supset D\rangle]\!]$ and $(y{:}C\supset D)N \in [\![(C\supset D)]\!]$ by assumption

The term $\mathsf{Cut}(\langle b\rangle M,(z)N)$ reduces to either:

$$\mathsf{Cut}(\langle a\rangle\ \mathsf{Cut}(\langle c\rangle T,(x)S),(y)U) \quad\text{or}\quad \mathsf{Cut}(\langle c\rangle T,(x)\ \mathsf{Cut}(\langle a\rangle S,(y)U))\ .$$

We have to show that both reducts are strongly normalising. We shall give the details for the first case.

Since M and N are not axioms, it follows from Lemma 2.3.6(i) that

$\langle b{:}C\supset D\rangle M \in \text{BINDING}_{\langle C\supset D\rangle}([\![(C\supset D)]\!]) \;\cup\; \text{IMPRIGHT}_{\langle C\supset D\rangle}([\![(C)]\!],[\![\langle D\rangle]\!])$
$(z{:}C\supset D)N \in \text{BINDING}_{(C\supset D)}([\![\langle C\supset D\rangle]\!]) \;\cup\; \text{IMPLEFT}_{(C\supset D)}([\![\langle C\rangle]\!],[\![(D)]\!])$

Again the proof splits into two cases depending on whether the co-named term $\langle b:C\supset D\rangle M$ or the named term $(z:C\supset D)N$ belong to BINDING. Let us assume $\langle b{:}C\supset D\rangle M$ is an element in $\text{BINDING}_{\langle C\supset D\rangle}([\![(C\supset D)]\!])$.

(2.1) $\langle b{:}C\supset D\rangle M \in \text{BINDING}_{\langle C\supset D\rangle}([\![(C\supset D)]\!])$ new assumption
(2.2) $M\{b := (v)P\} \in SN_{aux}$ for all $(v{:}C\supset D)P \in [\![(C\supset D)]\!]$ by Def. 2.3.3
(2.3) $M\{b := (z)N\} \in SN_{aux}$ by (1), (2.2) and $(v)P \equiv (z)N$
(2.4) The following calculation shows that $\mathsf{Cut}(\langle c\rangle M,(z)N) \in SN_{aux}$.

$$\begin{aligned} M\{b := (z)N\} &\equiv \mathsf{Imp}_R((x)\langle a\rangle S, b)\{b := (z)N\}\\ &= \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle S\{b := (z)N\}, b),(z)N)\\ &\overset{(*)}{\equiv} \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle S, b),(z)N)\\ &\equiv \mathsf{Cut}(\langle b\rangle M,(z)N) \end{aligned}$$

$(*)$ because M freshly introduces b.

Therefore we know that the term $\mathsf{Cut}(\langle b\rangle M,(z)N)$ is strongly normalising, and hence its reduct $\mathsf{Cut}(\langle a\rangle\ \mathsf{Cut}(\langle c\rangle T,(x)S),(y)U)$ must be strongly normalising, too. In fact both reducts must be strongly normalising. The case where $(z{:}C\supset D)N$ belongs to $\text{BINDING}_{(C\supset D)}([\![\langle C\supset D\rangle]\!])$ is similar.

We now have to show that the reduct is strongly normalising in the case where $\langle b:C\supset D\rangle M$ and $(z:C\supset D)N$ belong to $\text{IMPRIGHT}_{\langle C\supset D\rangle}([\![(C)]\!],[\![\langle D\rangle]\!])$ and to $\text{IMPLEFT}_{(C\supset D)}([\![\langle C\rangle]\!],[\![(D)]\!])$, respectively.

We first show that the inner cut of the reduct is strongly normalising.

(3.1) $\langle b:C\supset D\rangle M \in \text{IMPRIGHT}_{\langle C\supset D\rangle}([\![(C)]\!],[\![\langle D\rangle]\!])$,
$(z:C\supset D)N \in \text{IMPLEFT}_{(C\supset D)}([\![\langle C\rangle]\!],[\![(D)]\!])$ new assumption
(3.2) $(x:C)S\{a := (v{:}D)P\} \in [\![(C)]\!]$ for all $(v{:}D)P \in [\![(D)]\!]$,
$\langle c{:}C\rangle T \in [\![\langle C\rangle]\!]$ by Definition 2.3.3
(3.3) $S\{a := (v{:}D)P\}, T \in SN_{aux}$ by Lemma 2.3.13
(3.4) $\mathsf{Cut}(\langle c\rangle T,(x)S\{a := (v{:}D)P\}) \in SN_{aux}$ by ind. (the degree decreased)

As stated in Remark 2.2.7, we require that a is not free in $\langle c\rangle T$, and therefore we may move the substitution on the top-level. Thus we have that

$$\mathsf{Cut}(\langle c\rangle T,(x)S\{a := (v{:}D)P\}) \equiv \mathsf{Cut}(\langle c\rangle T,(x)S)\{a := (v{:}D)P\}\ .$$

(3.5) $\mathsf{Cut}(\langle c\rangle T,(x)S)\{a := (v{:}D)P\} \in SN_{aux}$ for all $(v{:}D)P \in [\![(D)]\!]$ by (3.4)
(3.6) $\langle a{:}D\rangle \mathsf{Cut}(\langle c\rangle T,(x)S) \in [\![\langle D\rangle]\!]$ by Definition 2.3.3

Now we show that the outer cut is strongly normalising.

(3.7) $(y{:}D)U \in [\![(D)]\!]$ by (3.1) and Definition 2.3.3
(3.8) $\mathsf{Cut}(\langle c\rangle T,(x)S), U \in SN_{aux}$ by Lemma 2.3.13
(3.9) $\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle T,(x)S),(y)U) \in SN_{aux}$ by ind. (the degree decreased)

We reason analogous in the case where

$$\mathsf{Cut}(\langle b\rangle M,(z)N) \xrightarrow{l} \mathsf{Cut}(\langle c\rangle T,(x)\ \mathsf{Cut}(\langle a\rangle S,(y)U))\ .$$

We have shown that all immediate reducts of $\mathsf{Cut}(\langle a\rangle M,(x)N)$ are strongly normalising. Consequently, $\mathsf{Cut}(\langle a\rangle M,(x)N)$ must be strongly normalising. Thus we are done. □

Proof of Lemma 2.3.18. We shall give five representative cases, in which the expression $\hat{\sigma},\{\sigma\}$ will stand for the set $\hat{\sigma}\cup\{\sigma\}$ with $\{\sigma\}\notin\hat{\sigma}$.

Case $\mathsf{Ax}(x,a)$**:** We have to prove that $\mathsf{Ax}(x,a)\ \hat{\sigma},\{x := \langle b\rangle P\},\{a := (y)Q\}$ is strongly normalising for arbitrary (co-)named terms $\langle b{:}B\rangle P \in [\![\langle B\rangle]\!]$ and $(y{:}B)Q \in [\![(B)]\!]$.

(1) $\mathsf{Ax}(x,a)\ \hat{\sigma},\{x := \langle b\rangle P\},\{a := (y)Q\} = \mathsf{Cut}(\langle b\rangle P,(y)Q)$ by Def. of $\{\_\}$
(2) $\langle b{:}B\rangle P \in [\![\langle B\rangle]\!]$ and $(y{:}B)Q \in [\![(B)]\!]$ by assumption
(3) $P \in SN_{aux}$ and $Q \in SN_{aux}$ by Lemma 2.3.13
(4) $\mathsf{Cut}(\langle b\rangle P,(y)Q) \in SN_{aux}$ by Lemma 2.3.14
(5) $\mathsf{Ax}(x,a)\ \hat{\sigma},\{x := \langle b\rangle P\},\{a := (y)Q\} \in SN_{aux}$ by (1) and (4)

Case $\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)$**:** We have to prove that $\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)\ \hat{\sigma},\{c := (z)R\}$ is strongly normalising for an arbitrary named term $(z{:}B\wedge C)R \in [\![(B\wedge C)]\!]$.

(1) $\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)\ \hat{\sigma},\{c{:=}(z)R\} =$
$\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle\ M\hat{\sigma},\{c := (z)R\},\langle b\rangle\ N\hat{\sigma},\{c := (z)R\},c),(z)R)$ Def. of $\{\_\}$
(2) $M\ \hat{\sigma},\{c := (z)R\},\{a := (y)P\} \in SN_{aux}$ for arbitrary $(y{:}B)P \in [\![(B)]\!]$,
$N\ \hat{\sigma},\{c := (z)R\},\{b := (x)Q\} \in SN_{aux}$ for arbitrary $(x{:}C)Q \in [\![(C)]\!]$
by induction
(3) $(M\hat{\sigma},\{c := (z)R\})\{a := (y)P\} \in SN_{aux}$,
$(N\hat{\sigma},\{c := (z)R\})\{b := (x)Q\} \in SN_{aux}$ by (2) and sss
(4) $\langle a{:}B\rangle(M\hat{\sigma},\{c := (z)R\}) \in [\![\langle B\rangle]\!]$,
$\langle b{:}C\rangle(N\hat{\sigma},\{c := (z)R\}) \in [\![\langle C\rangle]\!]$ by Definition 2.3.3
(5) $\mathsf{And}_R(\langle a\rangle M\hat{\sigma},\{c := (z)R\},\langle b\rangle N\hat{\sigma},\{c := (z)R\},c)$ freshly introduces c by (1)
(6) $\langle c{:}B\wedge C\rangle \mathsf{And}_R(\langle a\rangle M\hat{\sigma},\{c := (z)R\},\langle b\rangle N\hat{\sigma},\{c := (z)R\},c) \in [\![\langle B\wedge C\rangle]\!]$
by (4), (5) and Definition 2.3.3
(7) $\mathsf{And}_R(\langle a\rangle M\hat{\sigma},\{c := (z)R\},\langle b\rangle N\hat{\sigma},\{c := (z)R\},c) \in SN_{aux}$,
$R \in SN_{aux}$ by Lemma 2.3.13
(8) $\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M\hat{\sigma},\{c := (z)R\},\langle b\rangle N\hat{\sigma},\{c := (z)R\},c),(z)R) \in SN_{aux}$
by Lemma 2.3.14
(9) $\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)\ \hat{\sigma},\{c := (z)R\} \in SN_{aux}$ by (1) and (8)

Case $\mathsf{And}^i_L((x)M,y)$ $(i=1,2)$**:** We have to prove that $\mathsf{And}^i_L((x)M,y)\ \hat{\sigma},\{y:=\langle a\rangle R\}$ is strongly normalising for an arbitrary co-named term $\langle a{:}C_1\wedge C_2\rangle R\in[\![\langle C_1\wedge C_2\rangle]\!]$.

(1) $\mathsf{And}^i_L((x)M,y)\ \hat{\sigma},\{y:=\langle a\rangle R\}=$
$\mathsf{Cut}(\langle a\rangle R,(y)\mathsf{And}^i_L((x)M\ \hat{\sigma},\{y:=\langle a\rangle R\},y))$ by Definition of $\{\_\}$
(2) $M\ \hat{\sigma},\{y:=\langle a\rangle R\},\{x:=\langle b\rangle P\}\in SN_{aux}$ for arbitrary $\langle b{:}C_i\rangle P\in[\![\langle C_i\rangle]\!]$ by induction
(3) $(M\ \hat{\sigma},\{y:=\langle a\rangle R\})\{x:=\langle b\rangle P\}\in SN_{aux}$ by (2) and sss
(4) $(x{:}C_i)(M\hat{\sigma},\{y:=\langle a\rangle R\})\in[\![(C_i)]\!]$ by Definition 2.3.3
(5) $\mathsf{And}^i_L((x)M\ \hat{\sigma},\{y:=\langle a\rangle R\},y)$ freshly introduces y by (1)
(6) $(y{:}C_1\wedge C_2)\mathsf{And}^i_L((x)M\ \hat{\sigma},\{y:=\langle a\rangle R\},y)\in[\![(C_1\wedge C_2)]\!]$ by (4), (5) and Def. 2.3.3
(7) $\mathsf{And}^i_L((x)M\ \hat{\sigma},\{y:=\langle a\rangle R\},y),\ R\in SN_{aux}$ by Lemma 2.3.13
(8) $\mathsf{Cut}(\langle a\rangle R,(y)\mathsf{And}^i_L((x)M\ \hat{\sigma},\{y:=\langle a\rangle R\},y))\in SN_{aux}$ by Lemma 2.3.14
(9) $\mathsf{And}^i_L((x)M,y)\ \hat{\sigma},\{y:=\langle a\rangle R\}\in SN_{aux}$ by (1) and (8)

Case $\mathsf{Imp}_R((x)\langle a\rangle M,b)$**:** We prove that $\mathsf{Imp}_R((x)\langle a\rangle M,b)\ \hat{\sigma},\{b:=(z)R\}$ is strongly normalising for an arbitrary named term $(z{:}B\supset C)R$ that belongs to $[\![(B\supset C)]\!]$.

(1) $\mathsf{Imp}_R((x)\langle a\rangle M,b)\ \hat{\sigma},\{b:=(z)R\}=$
$\mathsf{Cut}(\langle c\rangle\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma},\{b:=(z)R\},b),(z)R)$ by Definition of $\{\_\}$
(2) $M\ \hat{\sigma},\{b:=(z)R\},\{a:=(y)P\},\{x:=\langle c\rangle Q\}\in SN_{aux}$
for arbitrary $(y{:}B)P\in[\![(B)]\!]$ and $\langle c{:}C\rangle Q\in[\![\langle C\rangle]\!]$ by induction
(3) $(M\hat{\sigma},\{b:=(z)R\},\{x:=\langle c\rangle Q\})\{a:=(y)P\}\in SN_{aux}$,
$(M\hat{\sigma},\{b:=(z)R\},\{a:=(y)P\})\{x:=\langle c\rangle Q\}\in SN_{aux}$ by (2) and sss
(4) $\langle a{:}B\rangle\ M\hat{\sigma},\{b:=(z)R\},\{x:=\langle c\rangle Q\}\in[\![\langle B\rangle]\!]$
$(x{:}C)\ M\hat{\sigma},\{b:=(z)R\},\{a:=(y)P\}\in[\![(C)]\!]$ by Definition 2.3.3
(5) $\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma},\{b:=(z)R\},b)$ freshly introduces b by (1)
(6) $\langle b{:}B\supset C\rangle\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma},\{b:=(z)R\},b)\in[\![\langle B\supset C\rangle]\!]$ by (4) and (5) and Definition 2.3.3
(7) $\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma},\{b:=(z)R\},b),\ R\in SN_{aux}$ by Lemma 2.3.13
(8) $\mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma},\{b:=(z)R\},b),(z)R)\in SN_{aux}$ by Lemma 2.3.14
(9) $\mathsf{Imp}_R((x)\langle a\rangle M,b)\ \hat{\sigma},\{b:=(z)R\}\in SN_{aux}$ by (1) and (8)

Case $\mathsf{Cut}(\langle a\rangle M,(x)N)$**:** Since we introduced in the definition of $\{\_\}$ two special clauses for cuts with an axiom as immediate subterm, we have to distinguish two cases.

Subcase I: M is an axiom that freshly introduces the label of the cut-formula (the case where N is such an axiom is similar). We have to show that for arbitrary $\langle b{:}B\rangle R\in[\![\langle B\rangle]\!]$ the term $\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)N)\ \hat{\sigma},\{x:=\langle b\rangle R\}$ is strongly normalising.

(1) $\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)N)\ \hat{\sigma},\{x := \langle b\rangle R\} =$
$\mathsf{Cut}(\langle b\rangle R,(y)\ N\hat{\sigma},\{x := \langle b\rangle R\})$ by Definition of $\{\_\}$
(2) $N\ \hat{\sigma},\{x := \langle b\rangle R\},\{y := \langle c\rangle P\} \in SN_{aux}$ for arbitrary $\langle c{:}B\rangle P \in [\![\langle B\rangle]\!]$ by induction
(3) $(N\ \hat{\sigma},\{x := \langle b\rangle R\})\{y := \langle c\rangle P\} \in SN_{aux}$ by (2) and sss
(4) $(y{:}\,B)\ N\hat{\sigma},\{x := \langle b\rangle R\} \in [\![(B)]\!]$ by Definition 2.3.3
(5) $N\ \hat{\sigma},\{x := \langle b\rangle R\} \in SN_{aux}$ and $R \in SN_{aux}$ by Lemma 2.3.13
(6) $\mathsf{Cut}(\langle b\rangle R,(y)\ N\hat{\sigma},\{x := \langle b\rangle R\}) \in SN_{aux}$ by Lemma 2.3.14
(7) $\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a),(y)N)\ \hat{\sigma},\{x := \langle b\rangle R\} \in SN_{aux}$ by (1) and (6)

Subcase II: M and N are not axioms that freshly introduce the label of the cut-formula. We have to prove that $\mathsf{Cut}(\langle a\rangle M,(x)N)\ \hat{\sigma}$ is strongly normalising.

(1) $\mathsf{Cut}(\langle a\rangle M,(x)N)\ \hat{\sigma} = \mathsf{Cut}(\langle a\rangle M\hat{\sigma},(x)N\hat{\sigma})$ by Definition of $\{\_\}$
(2) $M\ \hat{\sigma},\{a := (y)S\} \in SN_{aux}$ for arbitrary $(y{:}B)S \in [\![(B)]\!]$,
$N\ \hat{\sigma},\{x := \langle b\rangle T\} \in SN_{aux}$ for arbitrary $\langle b{:}B\rangle T \in [\![\langle B\rangle]\!]$ by induction
(3) $(M\hat{\sigma})\{a := (y)S\} \in SN_{aux}$ and $(N\hat{\sigma})\{x := \langle b\rangle T\} \in SN_{aux}$ by (2) and sss
(4) $\langle a{:}B\rangle\ M\hat{\sigma} \in [\![\langle B\rangle]\!]$ and $(x{:}B)\ N\hat{\sigma} \in [\![(B)]\!]$ by Definition 2.3.3
(5) $M\hat{\sigma} \in SN_{aux}$ and $N\hat{\sigma} \in SN_{aux}$ by Lemma 2.3.13
(6) $\mathsf{Cut}(\langle a\rangle\ M\hat{\sigma},(x)\ N\hat{\sigma}) \in SN_{aux}$ by Lemma 2.3.14
(7) $\mathsf{Cut}(\langle a\rangle M,(x)N)\ \hat{\sigma} \in SN_{aux}$ by (1) and (6)

□

Proof of Lemma 2.3.20. The non-trivial cases are where the definitions of $\{\_\}$ and $[\_]$ differ; two of them are listed below.

Case $M \equiv \mathsf{Ax}(y,a)$**:** Consider the substitutions:

$$\begin{array}{lclcl} M\{a := (x)N\} & \equiv & \mathsf{Ax}(y,a)\{a := (x)N\} & = & \mathsf{Cut}(\langle a\rangle\mathsf{Ax}(y,a),(x)N) \\ M[a := (x)N] & \equiv & \mathsf{Ax}(y,a)[a := (x)N] & = & N[x \mapsto y] \end{array}$$

We have two cases depending on whether N freshly introduces x. If N freshly introduces x, then $\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(y,a),(x)N) \xrightarrow{aux} N[x \mapsto y]$ by a logical reduction; in the other case

$$\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(y,a),(x)N) \xrightarrow{aux} N\{x := \langle a\rangle\mathsf{Ax}(y,a)\} \underset{\text{(by 2.3.9)}}{\xrightarrow{aux}{}^{*}} N[x \mapsto y]\ .$$

Thus we have that $\mathsf{Ax}(y,a)[a := (x)N] \xrightarrow{aux}{}^{*} \mathsf{Ax}(y,a)\{a := (x)N\}$.

Case $M \equiv \mathsf{Cut}(\langle b\rangle P,(y)\mathsf{Ax}(y,a))$**:** Consider the substitutions:

$$\begin{array}{lcl} M\{a := (x)N\} & \equiv & \mathsf{Cut}(\langle b\rangle P,(y)\mathsf{Ax}(y,a))\{a := (x)N\} \\ & = & \mathsf{Cut}(\langle b\rangle\ P\{a := (x)N\},(x)N) \\ & \underset{\text{(by IH)}}{\xrightarrow{aux}{}^{*}} & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N],(x)N) \\ M[a := (x)N] & \equiv & \mathsf{Cut}(\langle b\rangle P,(y)\mathsf{Ax}(y,a))[a := (x)N] \\ & = & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N],(y)\ \mathsf{Ax}(y,a)[a := (x)N]) \\ & = & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N],(y)\ N[x \mapsto y]) \\ & \overset{(*)}{\equiv} & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N],(x)N) \end{array}$$

$(*)$ because by the Barendregt-style naming convention y cannot be free in $(x)N$.

□

Proof of Lemma 2.3.21. We shall analyse all possible cases of $\xrightarrow{cut}$-reductions.

Inner Reduction: Given that $M \xrightarrow{cut} N$, there is a proper subterm in M, say S, which reduces to S'. This term S' is a subterm of N. We know by induction that $S \xrightarrow{aux}{}^{+} S'$ and by context closure that $M \xrightarrow{aux}{}^{+} N$.

Logical Reduction: This case is obvious, because both $\xrightarrow{cut}$ and $\xrightarrow{aux}$ perform the same logical reductions.

Commuting Reduction: Suppose $M \xrightarrow{c} N$ with $M \equiv \mathsf{Cut}(\langle a\rangle S, (x)T)$ and $N \equiv S[a := (x)T]$, then we know that $M \xrightarrow{c'} S\{a := (x)T\}$. From Lemma 2.3.20 we have that $S\{a := (x)T\} \xrightarrow{aux}{}^{*} N$ and therefore $M \xrightarrow{aux}{}^{+} N$. The symmetric case is analogous. $\square$

Proof of Lemma 2.6.4. The following measure reduces in every $\xrightarrow{x}$-reduction.

$$
\begin{array}{rclrcl}
[\![\mathsf{Ax}(x,a)]\!] & \stackrel{\text{def}}{=} & 1 & [\![\mathsf{Cut}(\langle a\rangle M,(x)N)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+[\![N]\!]+1\\
[\![\mathsf{Not}_R((x)M,a)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+1 & [\![\mathsf{Not}_L(\langle a\rangle M,x)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+1\\
[\![\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+[\![N]\!]+1 & [\![\mathsf{And}^i_L((x)M,y)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+1\\
[\![\mathsf{Or}^i_R(\langle a\rangle M,b)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+1 & [\![\mathsf{Or}_L((x)M,(y)N,z)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+[\![N]\!]+1\\
[\![\mathsf{Imp}_R((x)\langle a\rangle M,b)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+1 & [\![\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)]\!] & \stackrel{\text{def}}{=} & [\![M]\!]+[\![N]\!]+1\\
[\![\overrightarrow{\mathsf{Cut}}(\langle a\rangle M,(x)N)]\!] & \stackrel{\text{def}}{=} & ([\![M]\!]+1)\cdot(4[\![N]\!]+1) & [\![\overleftarrow{\mathsf{Cut}}(\langle a\rangle M,(x)N)]\!] & \stackrel{\text{def}}{=} & (4[\![M]\!]+1)\cdot([\![N]\!]+1)
\end{array}
$$

$\square$

Proof of Lemma 2.6.10. The calculation for (i) is as follows, where $(*)$ is by confluence of $\xrightarrow{x}$.

$$
|\overleftarrow{\mathsf{Cut}}(\langle a\rangle M,(x)N)|_x \stackrel{(*)}{\equiv} |\overleftarrow{\mathsf{Cut}}(\langle a\rangle\,|M|_x,(x)\,|N|_x)|_x \stackrel{(2.6.9)}{\equiv} |M|_x\{a := (x)\,|N|_x\}
$$

$\square$

Proof of Lemma 2.6.11. We shall analyse same cases of $\xrightarrow{loc}$-reductions.

Inner Reduction: We give two cases.

$M \equiv \mathsf{And}_R(\langle a\rangle S,\langle x\rangle T,b)$ and $N \equiv \mathsf{And}_R(\langle a\rangle S',\langle x\rangle T,b)$

(1)	$S \xrightarrow{loc} S'$	by assumption
(2)	$\lvert M\rvert_x \equiv \mathsf{And}_R(\langle a\rangle\lvert S\rvert_x,\langle x\rangle\lvert T\rvert_x,b)$	by Lemma 2.6.7
(3)	$\mathsf{And}_R(\langle a\rangle\lvert S\rvert_x,\langle x\rangle\lvert T\rvert_x,b) \xrightarrow{aux}{}^{*} \mathsf{And}_R(\langle a\rangle\lvert S'\rvert_x,\langle x\rangle\lvert T\rvert_x,b)$	by induction
(4)	$\mathsf{And}_R(\langle a\rangle\lvert S'\rvert_x,\langle x\rangle\lvert T\rvert_x,b) \equiv \lvert N\rvert_x$	by Lemma 2.6.7

$M \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(x)T)$ and $N \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle S',(x)T)$

(1)	$S \xrightarrow{loc} S'$	by assumption
(2)	$\lvert M\rvert_x \equiv \lvert S\rvert_x\{a := (x)\,\lvert T\rvert_x\}$	by Lemma 2.6.10
(3)	$\lvert S\rvert_x\{a := (x)\,\lvert T\rvert_x\} \xrightarrow{aux}{}^{*} \lvert S'\rvert_x\{a := (x)\,\lvert T\rvert_x\}$	by induction and Lem 2.3.11
(4)	$\lvert S'\rvert_x\{a := (x)\,\lvert T\rvert_x\} \equiv \lvert N\rvert_x$	by Lemma 2.6.10

Commuting Reduction: We give one case.

$M \equiv \mathsf{Cut}(\langle a\rangle S,(x)T)$ and $N \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(x)T)$

(1) $|M|_x \equiv \mathsf{Cut}(\langle a\rangle |S|_x,(x)|T|_x)$ by Lemma 2.6.7

(2) $\mathsf{Cut}(\langle a\rangle |S|_x,(x)|T|_x) \xrightarrow{aux}{}^{0/1} |S|_x\{a := (x)\,|T|_x\}$

'0'-case: if $|S|_x$ freshly introduces a

(3) $|S|_x\{a := (x)\,|T|_x\} \equiv |N|_x$ by Lemma 2.6.10

Labelled Cut Reduction: Trivial, because if $M \xrightarrow{x} N$, we have $|M|_x \equiv |N|_x$.

Logical Reduction: Routine calculation using Lemma 2.6.7.

Garbage Reduction: Take for example $M \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(x)T)$ with a not free in S, then by Lemma 2.6.10 $|M|_x \equiv |N|_x$. □

Proof of Lemma 2.6.16. We shall analyse in detail one case where an $\xrightarrow{x}$-reduction occurs on the top-level. Suppose

$$M \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle \mathsf{Not}_R((x)S,a),(y)T) \quad \text{and}$$
$$N \equiv \mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)\overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)T),a),(y)T).$$

Hence we have $M \xrightarrow{x} N$. The subterms of N are:

(1) all subterms of S and T,
(2) $\overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)T)$,
(3) $\mathsf{Not}_R((x)\overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)T),a)$ and
(4) $N \equiv \mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)\overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)T),a),(y)T)$

We have to show that their x-normal form is an element in SN_{aux}. Case (1) follows by assumption: the x-normal form of every subterm of M is an element in SN_{aux}, and therefore the x-normal form of every subterm of S and T must be in SN_{aux}, too. Case (4) follows by Lemma 2.6.11, which says that $|M|_x \xrightarrow{aux}{}^{*} |N|_x$, and thus $|N|_x \in SN_{aux}$. By Lemma 2.6.7 we have the identities

$$\begin{array}{rl} & |\mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)\overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)T),a),(y)T)|_x \\ \equiv & \mathsf{Cut}(\langle a\rangle |\mathsf{Not}_R((x)\overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)T),a)|_x,(y)|T|_x) \\ \equiv & \mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)|\overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)T)|_x,a),(y)|T|_x) \end{array}$$

and therefore the x-normal form of (2) and (3) are in SN_{aux}; so we are done. □

Proof of Lemma 2.6.20. For every reduction we have to do check whether the corresponding terms are ordered decreasingly according to definition of $>^{rpo}$. Since there are many cases, we shall present only a few representative of them. We write rpo as shorthand for Definition 2.6.12.

Inner Reduction: We give one case.

$M \equiv \mathsf{Cut}(\langle a\rangle S,(x)T) \xrightarrow{loc} \mathsf{Cut}(\langle a\rangle S',(x)T) \equiv N$

(1) $S \xrightarrow{loc} S'$ and $\underline{S} >^{rpo} \underline{S'}$ — by assumption and induction

(2) $\underline{M} = \underline{S} \cdot_m \underline{T}$ and $\underline{N} = \underline{S'} \cdot_n \underline{T}$ — by Definition 2.6.17

(3) $m \geq n$ — by Lemma 2.6.19(i)

(4) $\underline{S} \cdot_m \underline{T} >^{rpo} \underline{S'}$ and $\underline{S} \cdot_m \underline{T} >^{rpo} \underline{T}$ — by (1) and rpo(i)

(5) $\{\!\!\{\underline{S},\underline{T}\}\!\!\} >^{rpo}_{mult} \{\!\!\{\underline{S'},\underline{T}\}\!\!\}$ — by (1) and rpo(i)

(6) $\underline{M} >^{rpo} \underline{N}$ — if $m = n$, then by (5) and rpo(iii); if $m > n$, then by (4) and rpo(ii)

Labelled Cut Reduction: We give five typical cases.

$M \equiv \overleftarrow{\mathsf{Cut}}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S,\langle b\rangle T,c),(x)U)$
$\quad \xrightarrow{x} \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle \overleftarrow{\mathsf{Cut}}(\langle c\rangle S,(x)U),\langle b\rangle \overleftarrow{\mathsf{Cut}}(\langle c\rangle T,(x)U),c),(x)U) \equiv N$

(1) $\underline{M} = (\!|\underline{S},\underline{T}|\!)\,\langle \underline{U}\rangle_m$ and $\underline{N} = (\!|\underline{S}\langle\underline{U}\rangle_r,\underline{T}\langle\underline{U}\rangle_s|\!) \cdot_t \underline{U}$ — by Definition 2.6.17

(2) $m \geq t,r,s$ — by Lemma 2.6.19(i,ii)

(3) $\_\langle\_\rangle_m \gg \_ \cdot_t \_$ — by (2) and Definition 2.6.14

(4) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{S}$ and $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{U}$ — by rpo(i)

(5) $\{\!\!\{(\!|\underline{S},\underline{T}|\!),\underline{U}\}\!\!\} >^{rpo}_{mult} \{\!\!\{\underline{S},\underline{U}\}\!\!\}$ — by rpo(i)

(6) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{S}\langle\underline{U}\rangle_r$ — if $m = r$, then by (5) and rpo(iii); if $m > r$, then by (4) and rpo(ii)

(7) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{T}\langle\underline{U}\rangle_s$ — analogous to (4,5,6)

(8) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} (\!|\underline{S}\langle\underline{U}\rangle_r,\underline{T}\langle\underline{U}\rangle_s|\!)$ — by (6,7) and rpo(ii)

(9) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{U}$ — by rpo(i)

(10) $\underline{M} >^{rpo} \underline{N}$ — by (3,8,9) and rpo(ii)

$M \equiv \overleftarrow{\mathsf{Cut}}(\langle d\rangle \mathsf{And}_R(\langle a\rangle S,\langle b\rangle T,c),(x)U)$
$\quad \xrightarrow{x} \mathsf{And}_R(\langle a\rangle \overleftarrow{\mathsf{Cut}}(\langle d\rangle S,(x)U),\langle b\rangle \overleftarrow{\mathsf{Cut}}(\langle d\rangle T,(x)U),c) \equiv N$

(1) $\underline{M} = (\!|\underline{S},\underline{T}|\!)\,\langle \underline{U}\rangle_m$ and $\underline{N} = (\!|\underline{S}\langle\underline{U}\rangle_r,\underline{T}\langle\underline{U}\rangle_s|\!)$ — by Definition 2.6.17

(2) $m \geq r,s$ — by Lemma 2.6.19(i,ii)

(3) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{S}$ and $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{U}$ — by rpo(i)

(4) $\{\!\!\{(\!|\underline{S},\underline{T}|\!),\underline{U}\}\!\!\} >^{rpo}_{mult} \{\!\!\{\underline{S},\underline{U}\}\!\!\}$ — by rpo(i)

(5) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{S}\langle\underline{U}\rangle_r$ — if $m = r$, then by (4) and rpo(iii); if $m > r$, then by (3) and rpo(ii)

(6) $(\!|\underline{S},\underline{T}|\!)\langle\underline{U}\rangle_m >^{rpo} \underline{T}\langle\underline{U}\rangle_s$ — analogous to (3,4,5)

(7) $\underline{M} >^{rpo} \underline{N}$ — by (5,6) and rpo(ii)

$M \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S,(x)T),(y)U)$
$\quad \xrightarrow{x} \mathsf{Cut}(\langle b\rangle \overleftarrow{\mathsf{Cut}}(\langle a\rangle S,(y)U),(x)\overleftarrow{\mathsf{Cut}}(\langle a\rangle T,(y)U)) \equiv N$

(1) $\underline{M} = (\underline{S} \cdot_m \underline{T})\langle\underline{U}\rangle_n$ and $\underline{N} = \underline{S}\langle\underline{U}\rangle_r \cdot_s \underline{T}\langle\underline{U}\rangle_t$ — by Definition 2.6.17

(2) $n \geq s,r,t$ — by Lemma 2.6.19(i,ii)

(3) $\_\langle\_\rangle_n \gg \_ \cdot_s \_$ — by (2) and Definition 2.6.14

(4)	$(\underline{S} \cdot_m \underline{T})\langle\underline{U}\rangle_n >^{rpo} \underline{S}$ and $(\underline{S} \cdot_m \underline{T})\langle\underline{U}\rangle_n >^{rpo} \underline{U}$	by rpo(i)
(5)	$\{\!\vert(\underline{S} \cdot_m \underline{T}), \underline{U}\vert\!\} >^{rpo}_{mult} \{\!\vert\underline{S}, \underline{U}\vert\!\}$	by rpo(i)
(6)	$(\underline{S} \cdot_m \underline{T})\langle\underline{U}\rangle_n >^{rpo} \underline{S}\langle\underline{U}\rangle_r$	if $n = r$, then by (5) and rpo(iii)
		if $n > r$, then by (4) and rpo(ii)
(7)	$(\underline{S} \cdot_m \underline{T})\langle\underline{U}\rangle_n >^{rpo} \underline{T}\langle\underline{U}\rangle_t$	analogous to (4,5,6)
(8)	$\underline{M} >^{rpo} \underline{N}$	by (3,6,7) and rpo(ii)

$M \equiv \mathsf{C\overleftarrow{u}t}(\langle a\rangle\mathsf{Cut}(\langle b\rangle S, (x)\mathsf{Ax}(x,a)), (y)U)$
$\xrightarrow{x} \mathsf{Cut}(\langle b\rangle\mathsf{C\overleftarrow{u}t}(\langle a\rangle S, (y)U), (y)U) \equiv N$

(1)	$\underline{M} = (\underline{S} \cdot_m \star)\langle\underline{U}\rangle_n$ and $\underline{N} = \underline{S}\langle\underline{U}\rangle_r \cdot_s \underline{U}$	by Definition 2.6.17
(2)	$n \geq s, r$	by Lemma 2.6.19(i,ii)
(3)	$\_\langle\_\rangle_n \gg \_ \cdot_s \_$	by (2) and Definition 2.6.14
(4)	$(\underline{S} \cdot_m \star)\langle\underline{U}\rangle_n >^{rpo} \underline{S}$ and $(\underline{S} \cdot_m \star)\langle\underline{U}\rangle_n >^{rpo} \underline{U}$	by rpo(i)
(5)	$\{\!\vert(\underline{S} \cdot_m \star), \underline{U}\vert\!\} >^{rpo}_{mult} \{\!\vert\underline{S}, \underline{U}\vert\!\}$	by rpo(i)
(6)	$(\underline{S} \cdot_m \star)\langle\underline{U}\rangle_n >^{rpo} \underline{S}\langle\underline{U}\rangle_r$	if $n = r$, then by (5) and rpo(iii)
		if $n > r$, then by (4) and rpo(ii)
(7)	$\underline{M} >^{rpo} \underline{N}$	by (3,4,6) and rpo(ii)

$M \equiv \mathsf{C\overleftarrow{u}t}(\langle a\rangle\mathsf{Ax}(x,a), (y)S) \xrightarrow{x} \mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a), (y)S) \equiv N$

(1)	$\underline{M} = \star\langle\underline{S}\rangle_m$ and $\underline{N} = \star \cdot_n \underline{S}$	by Definition 2.6.17
(2)	$m \geq n$	by Lemma 2.6.19(i)
(3)	$\_\langle\_\rangle_m \gg \_ \cdot_n \_$	by (2) and Definition 2.6.14
(4)	$\star\langle\underline{S}\rangle_m >^{rpo} \star$ and $\star\langle\underline{S}\rangle_m >^{rpo} \underline{S}$	by rpo(i)
(5)	$\underline{M} >^{rpo} \underline{N}$	by (3,4) and rpo(ii)

Commuting Reduction: One case is as follows.

$M \equiv \mathsf{Cut}(\langle a\rangle S, (x)T) \xrightarrow{c''} \mathsf{C\overleftarrow{u}t}(\langle a\rangle S, (x)T) \equiv N$

(1)	$\underline{M} = \underline{S} \cdot_m \underline{T}$ and $\underline{N} = \underline{S}\langle\underline{T}\rangle_n$	by Definition 2.6.17
(2)	$m > n$	by Lemma 2.6.19(iii)
(3)	$\_ \cdot_m \_ \gg \_\langle\_\rangle_n$	by (2) and Definition 2.6.14
(4)	$\underline{S} \cdot_m \underline{T} >^{rpo} \underline{S}$ and $\underline{S} \cdot_m \underline{T} >^{rpo} \underline{T}$	by rpo(i)
(5)	$\underline{M} >^{rpo} \underline{N}$	by (3,4) and rpo(ii)

Logical Reduction: We tackle three representative cases.

$M \equiv \mathsf{Cut}(\langle a\rangle\mathsf{Ax}(x,a), (y)S) \xrightarrow{l} S[y \mapsto x] \equiv N$

(1)	$\underline{M} = \star \cdot_n \underline{S}$ and $\underline{N} = \underline{S}$	by Definition 2.6.17
(2)	$\underline{M} >^{rpo} \underline{N}$	by (1) and rpo(ii)

$M \equiv \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c), (y)\mathsf{And}_L^1((x)U, y)) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle S, (x)U) \equiv N$

(1)	$\underline{M} = (\!|\underline{S}, \underline{T}|\!) \cdot_m (\!|\underline{U}|\!)$ and $\underline{N} = \underline{S} \cdot_n \underline{U}$	by Definition 2.6.17
(2)	$m > n$	by Lemma 2.6.19(iii)
(3)	$(\!|\underline{S}, \underline{T}|\!) \cdot_m (\!|\underline{U}|\!) >^{rpo} \underline{S}$, $(\!|\underline{S}, \underline{T}|\!) \cdot_m (\!|\underline{U}|\!) >^{rpo} \underline{U}$	by rpo(i)
(4)	$\underline{M} >^{rpo} \underline{N}$	by (2,3) and rpo(ii)

$M \equiv \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle S, b), (z)\mathsf{Imp}_L(\langle c\rangle T, (y)U, z))$
$\quad \xrightarrow{l} \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle T, (x)S), (y)U) \equiv N$

(1)	$\underline{M} = (\!|\underline{S}|\!) \cdot_m (\!|\underline{T}, \underline{U}|\!)$ and $\underline{N} = (\underline{T} \cdot_s \underline{S}) \cdot_t \underline{U}$	by Definition 2.6.17
(2)	$m > s, t$	by Lemma 2.6.19(iii,ii)
(3)	$(\!|\underline{S}|\!) \cdot_m (\!|\underline{T}, \underline{U}|\!) >^{rpo} \underline{S}$, $(\!|\underline{S}|\!) \cdot_m (\!|\underline{T}, \underline{U}|\!) >^{rpo} \underline{T}$	by rpo(i)
(4)	$(\!|\underline{S}|\!) \cdot_m (\!|\underline{T}, \underline{U}|\!) >^{rpo} \underline{T} \cdot_s \underline{S}$	by (2,3) and rpo(ii)
(5)	$(\!|\underline{S}|\!) \cdot_m (\!|\underline{T}, \underline{U}|\!) >^{rpo} \underline{U}$	by Definition 2.6.12(i)
(6)	$\underline{M} >^{rpo} \underline{N}$	by (2,4,5) and rpo(ii)

Garbage Reduction: Finally, we give one case where a garbage reduction is performed.

$M \equiv \overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T) \xrightarrow{gc} S \equiv N$

(1)	$\underline{M} = \underline{S}\langle \underline{T}\rangle_m$ and $\underline{N} = \underline{S}$	by Definition 2.6.17
(2)	$\underline{M} >^{rpo} \underline{N}$	by (1) and rpo(i)

$\square$

Proof of Lemma 2.6.22. We give the details of how to replace $\overleftarrow{\mathsf{Cut}}$-instances. Suppose $\overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)$ is a subterm of M with the typing judgement

$$\dfrac{\Gamma_1 \triangleright S \triangleright \Delta_1 \qquad \Gamma_2 \triangleright T \triangleright \Delta_2}{\Gamma_1, \Gamma_2 \triangleright \overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T) \triangleright \Delta_1, \Delta_2}\ \overleftarrow{\mathsf{Cut}}\ .$$

Our argument splits into two cases:

- If the subterm S does not freshly introduce a, then we replace $\overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)$ simply by $\mathsf{Cut}(\langle a\rangle S, (x)T)$ (both terms have the same typing judgement). In this case we have $\mathsf{Cut}(\langle a\rangle S, (x)T) \xrightarrow{c''} \overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)$.
- The more interesting case is where S freshly introduces a. Here we cannot simply replace $\overleftarrow{\mathsf{Cut}}$ with Cut, because there is no reduction that gives $N \xrightarrow{loc}{}^{+} M$. Therefore we replace $\overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T)$ by $\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y, c)), (x)T)$ where b and c are fresh co-names that do not occur anywhere else. On the level of type derivations this means that an instance of the form

$$\dfrac{\Gamma_1 \triangleright S \triangleright \Delta_1 \qquad \Gamma_2 \triangleright T \triangleright \Delta_2}{\Gamma_1, \Gamma_2 \triangleright \overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T) \triangleright \Delta_1, \Delta_2}\ \overleftarrow{\mathsf{Cut}}$$

  is replaced by

$$\dfrac{\dfrac{\Gamma_1 \triangleright S \triangleright \Delta_1, b : B \qquad \overline{y : B \triangleright \mathsf{Ax}(y, c) \triangleright c : B}}{\Gamma_1 \triangleright \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y, c)) \triangleright \Delta_1, c : B}\ \mathsf{Cut} \qquad \Gamma_2 \triangleright T \triangleright \Delta_2}{\Gamma_1, \Gamma_2 \triangleright \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y, c)), (x)T) \triangleright \Delta_1, \Delta_2, c : B}\ \mathsf{Cut}$$

It is routine to verify that the new cut-instances are all well-typed. Now we show how the new term can reduce. Because $\mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c))$ does not freshly introduce a, we can first perform a commuting reduction

$$\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T) \xrightarrow{c''} \overleftarrow{\mathsf{Cut}}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T)$$

and subsequently we can remove the cut on the axiom by a $\xrightarrow{c''}$-reduction followed by a $\xrightarrow{gc}$-reduction.

$$\begin{array}{rl} & \overleftarrow{\mathsf{Cut}}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T) \\ \xrightarrow{c''} & \overleftarrow{\mathsf{Cut}}(\langle a\rangle \overleftarrow{\mathsf{Cut}}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T) \\ \xrightarrow{gc} & \overleftarrow{\mathsf{Cut}}(\langle a\rangle S, (x)T) \end{array}$$

Thus we are done. □

B.2 Proofs of Chapter 3

In this section we are mainly concerned with translations between natural deduction calculi and sequent calculi. First let us give two facts, which will be useful later.

Fact B.2.1:

(i) $\mathsf{Cut}(\langle a\rangle T, (x)S) \xrightarrow{int}{}^{*} T[a := (x)S]$

(ii) $\mathsf{Cut}(\langle a\rangle T, (x)S) \xrightarrow{int}{}^{*} S[x := \langle a\rangle T]$

Proof. We give the details for (i). If T does not freshly introduce a, then by a commuting reduction $\mathsf{Cut}(\langle a\rangle T, (x)S) \xrightarrow{int} T[a := (x)S]$. Otherwise, if T freshly introduces a, but is not an axiom, then $\mathsf{Cut}(\langle a\rangle T, (x)S) \equiv T[a := (x)S]$; if T is an axiom, for example $\mathsf{Ax}(z,a)$, then

$$\mathsf{Cut}(\langle a\rangle T, (x)S) \xrightarrow{int}{}^{+} S[x \mapsto z] = \mathsf{Ax}(z,a)[a := (x)S]$$

by a logical reduction or by a commuting reduction and Lemma 2.2.11. Thus we are done. □

Proof of Lemma 3.2.1. The difficult case is as follows.

$$\begin{array}{rll} & \mathsf{Ax}(x,a)[a := (y)S][x := \langle b\rangle T] & \\ = & S[y \mapsto x][x := \langle b\rangle T] & \\ \equiv & S[y := \langle b\rangle T][x := \langle b\rangle T] & \\ = & \mathsf{Cut}(\langle b\rangle T, (y)S)[x := \langle b\rangle T] & (1) \\ = & \mathsf{Cut}(\langle b\rangle T, (y)S[x := \langle b\rangle T]) & \\ \xrightarrow{int}{}^{*} & T[b := (y)S[x := \langle b\rangle T]] & (2) \\ \equiv & T[b \mapsto a][a := (y)S[x := \langle b\rangle T]] & (3) \\ = & \mathsf{Ax}(x,a)[x := \langle b\rangle T][a := (y)S[x := \langle b\rangle T]] & \end{array}$$

(1) by assumption S freshly introduces y and is not an axiom
(2) by Fact B.2.1
(3) by assumption $a \notin FC(\langle b\rangle T)$

□

Proof of Lemma 3.2.3. If M is a variable, then the lemma is by routine calculation. In case M corresponds to a term given in Lemma 3.2.2, then $|M|^{\mathsf{s}}_{a}$ cannot introduce x and hence the substitution is moved inside the subterm(s). In consequence, one can apply the induction hypothesis. Otherwise, the top-level term constructor of M corresponds to an elimination rule, for which we illustrate the lemma with one case.

Case $M \equiv S\,T$:

$$
\begin{array}{rll}
 & |S\,T|^{\mathsf{s}}_{a}[\sigma] & \text{with } [\sigma] \equiv [x := \langle b\rangle|N|^{\mathsf{s}}_{b}] \\
= & |S|^{\mathsf{s}}_{c}[c := (y)\mathsf{Imp}_L(\langle d\rangle|T|^{\mathsf{s}}_{d}, (z)\mathsf{Ax}(z,a), y)][\sigma] & \text{by (6) of } |\_|^{\mathsf{s}} \\
\xrightarrow{int}{}^{*} & |S|^{\mathsf{s}}_{c}[\sigma][c := (y)\mathsf{Imp}_L(\langle d\rangle|T|^{\mathsf{s}}_{d}, (z)\mathsf{Ax}(z,a), y)[\sigma]] & \text{by Lemma 3.2.1} \\
= & |S|^{\mathsf{s}}_{c}[\sigma][c := (y)\mathsf{Imp}_L(\langle d\rangle|T|^{\mathsf{s}}_{d}[\sigma], (z)\mathsf{Ax}(z,a), y)] & \\
\xrightarrow{int}{}^{*} & |S[x := N]|^{\mathsf{s}}_{c}[c := (y)\mathsf{Imp}_L(\langle d\rangle|T[x := N]|^{\mathsf{s}}_{d}, (z)\mathsf{Ax}(z,a), y)] & \text{by IH} \\
= & |S[x := N]\; T[x := N]|^{\mathsf{s}}_{a} & \text{by (6) of } |\_|^{\mathsf{s}} \\
\equiv & |(S\,T)[x := N]|^{\mathsf{s}}_{a} &
\end{array}
$$

□

Proof of Proposition 3.2.4. We shall begin by giving in turn the details where a beta-reduction occurs on the top-level.

Case $M \equiv (\lambda x.S)\,T \xrightarrow{\beta} S[x := T] \equiv M'$:

$$
\begin{array}{rll}
\text{(i)} & |(\lambda x.S)\,T|^{\mathsf{s}}_{c} & \\
= & \mathsf{Imp}_R((x)\langle a\rangle|S|^{\mathsf{s}}_{a}, b)[b := (y)\mathsf{Imp}_L(\langle d\rangle|T|^{\mathsf{s}}_{d}, (z)\mathsf{Ax}(z,c), y)] & \text{by (2,6) of } |\_|^{\mathsf{s}} \\
= & \mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle|S|^{\mathsf{s}}_{a}, b), (y)\mathsf{Imp}_L(\langle d\rangle|T|^{\mathsf{s}}_{d}, (z)\mathsf{Ax}(z,c), y)) & \text{by Lem. 3.2.2} \\
\xrightarrow{int} & \mathsf{Cut}(\langle a\rangle\mathsf{Cut}(\langle d\rangle|T|^{\mathsf{s}}_{d}, (x)|S|^{\mathsf{s}}_{a}), (z)\mathsf{Ax}(z,c)) & y \text{ is freshly introduced by (6) of } |\_|^{\mathsf{s}} \\
\xrightarrow{int}{}^{+} & \mathsf{Cut}(\langle d\rangle|T|^{\mathsf{s}}_{d}, (x)|S|^{\mathsf{s}}_{a})[a \mapsto c] & \text{by a comm. reduction and Lem. 2.2.11} \\
\equiv & \mathsf{Cut}(\langle d\rangle|T|^{\mathsf{s}}_{d}, (x)|S|^{\mathsf{s}}_{c}) & a \notin FC(\langle d\rangle|T|^{\mathsf{s}}_{d}) \text{ by Remark 2.2.7} \\
\xrightarrow{int}{}^{*} & |S|^{\mathsf{s}}_{c}[x := \langle d\rangle|T|^{\mathsf{s}}_{d}] & \text{by Fact B.2.1} \\
\xrightarrow{int}{}^{*} & |S[x := T]|^{\mathsf{s}}_{c} & \text{by Lemma 3.2.3}
\end{array}
$$

$$
\begin{array}{rll}
\text{(ii)} & |(\lambda x.S)\,T|^{\mathsf{s}}_{c}[\sigma] & \text{with } [\sigma] \equiv [c := (y)N] \\
= & \mathsf{Imp}_R((x)\langle c\rangle|S|^{\mathsf{s}}_{c}, a)[a := (z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (x)\mathsf{Ax}(x,c), z)][\sigma] & \text{by (2,6) of } |\_|^{\mathsf{s}} \\
= & \mathsf{Cut}(\langle a\rangle\mathsf{Imp}_R((x)\langle c\rangle|S|^{\mathsf{s}}_{c}, a), (z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (x)\mathsf{Ax}(x,c), z))[\sigma] & \text{by Lem. 3.2.2} \\
= & \mathsf{Cut}(\langle a\rangle\mathsf{Imp}_R((x)\langle c\rangle|S|^{\mathsf{s}}_{c}, a), (z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (y)N, z)) & c \notin FC(\langle b\rangle|T|^{\mathsf{s}}_{b}, \langle c\rangle|S|^{\mathsf{s}}_{c}) \\
\xrightarrow{int} & \mathsf{Cut}(\langle b\rangle|T|^{\mathsf{s}}_{b}, (x)\mathsf{Cut}(\langle c\rangle|S|^{\mathsf{s}}_{c}, (y)N)) & z \text{ is freshly introduced by (6) of } |\_|^{\mathsf{s}} \\
\xrightarrow{int} & \mathsf{Cut}(\langle c\rangle|S|^{\mathsf{s}}_{c}, (y)N)[x := \langle b\rangle|T|^{\mathsf{s}}_{b}] & \\
= & \mathsf{Cut}(\langle c\rangle|S|^{\mathsf{s}}_{c}[x := \langle b\rangle|T|^{\mathsf{s}}_{b}], (y)N) & x \notin FN((y)N) \text{ by Remark 2.2.7} \\
\xrightarrow{int}{}^{*} & \mathsf{Cut}(\langle c\rangle|S[x := T]|^{\mathsf{s}}_{c}, (y)N) & \text{by Lemma 3.2.3} \\
\xrightarrow{int}{}^{*} & |S[x := T]|^{\mathsf{s}}_{c}[\sigma] & \text{by Fact B.2.1}
\end{array}
$$

We omit the other base cases and proceed with the details for inner reductions. In case M corresponds to a term given in Lemma 3.2.2, (i) and (ii) are by routine calculation using the induction hypothesis (note in (ii) $|M|^{\mathsf{s}}_{c}[c := (x)N]$ expands to a cut). We are left to check the cases where the top-level term constructor of M corresponds to an elimination rule.

Case $M \equiv S\,T \xrightarrow{\beta} S'\,T \equiv M'$ with $S \xrightarrow{\beta} S'$:

By IH we have $|S|^{\mathsf{s}}_{c} \xrightarrow{int}{}^{+} |S|^{\mathsf{s}}_{c}$ and $|S|^{\mathsf{s}}_{c}[c := (y)P] \xrightarrow{int}{}^{+} |S'|^{\mathsf{s}}_{c}[c := (y)P]$ for any substitution where P freshly introduces y and is not an axiom. Take $(y)P$ to be $(y)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)$, then we have the following reduction.

$$
\begin{array}{lll}
\text{(i)} & |S\,T|^{\mathsf{s}}_{c} & \\
= & |S|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)] & \text{by (6) of } |\_|^{\mathsf{s}} \\
\xrightarrow{int}{}^{+} & |S'|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)] & \text{by IH} \\
= & |S'\,T|^{\mathsf{s}}_{c} & \text{by (6) of } |\_|^{\mathsf{s}}
\end{array}
$$

In (ii) we take $(y)P$ to be $(z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (u)\mathsf{Ax}(u,c), z)[c := (y)N]$.

$$
\begin{array}{lll}
\text{(ii)} & |S\,T|^{\mathsf{s}}_{c}[c := (y)N] & \\
= & |S|^{\mathsf{s}}_{a}[a := (z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (u)\mathsf{Ax}(u,c), z)][c := (y)N] & \text{by (6) of } |\_|^{\mathsf{s}} \\
\equiv & |S|^{\mathsf{s}}_{a}[a := (z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (u)\mathsf{Ax}(u,c), z)[c := (y)N]] & \\
\xrightarrow{int}{}^{+} & |S'|^{\mathsf{s}}_{a}[a := (z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (u)\mathsf{Ax}(u,c), z)[c := (y)N]] & \text{by IH} \\
\equiv & |S'|^{\mathsf{s}}_{a}[a := (z)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (u)\mathsf{Ax}(u,c), z)][c := (y)N] & \\
= & |S'\,T|^{\mathsf{s}}_{c}[c := (y)N] & \text{by (6) of } |\_|^{\mathsf{s}}
\end{array}
$$

Case $M \equiv S\,T \xrightarrow{\beta} S\,T' \equiv M'$ with $T \xrightarrow{\beta} T'$:

By IH we have $|T|^{\mathsf{s}}_{c} \xrightarrow{int}{}^{+} |T|^{\mathsf{s}}_{c}$ and $|T|^{\mathsf{s}}_{c}[c := (y)P] \xrightarrow{int}{}^{+} |T'|^{\mathsf{s}}_{c}[c := (y)P]$ for any substitution where P freshly introduces y and is not an axiom.

$$
\begin{array}{lll}
\text{(i)} & |S\,T|^{\mathsf{s}}_{c} & \\
= & |S|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)] & \text{by (6) of } |\_|^{\mathsf{s}} \\
\xrightarrow{int}{}^{+} & |S|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T'|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)] & \text{by IH} \\
= & |S\,T'|^{\mathsf{s}}_{c} & \text{by (6) of } |\_|^{\mathsf{s}}
\end{array}
$$

$$
\begin{array}{lll}
\text{(ii)} & |S\,T|^{\mathsf{s}}_{c}[c := (y)N] & \\
= & |S|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)][c := (y)N] & \text{by (6) of } |\_|^{\mathsf{s}} \\
\equiv & |S|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)[c := (y)N]] & \\
\xrightarrow{int}{}^{+} & |S|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T'|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)[c := (y)N]] & \text{by IH} \\
\equiv & |S|^{\mathsf{s}}_{a}[a := (y)\mathsf{Imp}_L(\langle b\rangle|T'|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)][c := (y)N] & \\
= & |S\,T'|^{\mathsf{s}}_{c}[c := (y)N] & \text{by (6) of } |\_|^{\mathsf{s}}
\end{array}
$$

□

Proof of Proposition 3.2.5. By induction on the definition of $\xrightarrow{\gamma}$. Here the induction steps are trivial. We illustrate the proof with one base case.

Case $\mathtt{case}(P, \lambda x.M, \lambda y.N)\ Q \xrightarrow{\gamma} \mathtt{case}(P, \lambda x.(M\,Q), \lambda y.(N\,Q))$:

$$
\begin{array}{lll}
 & |\mathtt{case}(P, \lambda x.M, \lambda y.N)\,Q|^{\mathsf{s}}_{c} & \text{by (6) of } |\_|^{\mathsf{s}} \\
= & |\mathtt{case}(P, \lambda x.M, \lambda y.N)|^{\mathsf{s}}_{a}\,[\sigma] \qquad \text{with } [\sigma] \equiv [a := (y)\mathsf{Imp}_L(\langle b\rangle|Q|^{\mathsf{s}}_{b}, (z)\mathsf{Ax}(z,c), y)] & \\
= & |P|^{\mathsf{s}}_{d}\,[d := (u)\mathsf{Or}_L((x)|M|^{\mathsf{s}}_{a}, (y)|N|^{\mathsf{s}}_{a}, u)]\,[\sigma] & \text{by (9) of } |\_|^{\mathsf{s}} \\
\equiv & |P|^{\mathsf{s}}_{d}\,[d := (u)\mathsf{Or}_L((x)\,|M|^{\mathsf{s}}_{a}[\sigma], (y)\,|N|^{\mathsf{s}}_{a}[\sigma], u)] & \\
= & |P|^{\mathsf{s}}_{d}\,[d := (u)\mathsf{Or}_L((x)|M\,Q|^{\mathsf{s}}_{c}, (y)|N\,Q|^{\mathsf{s}}_{c}, u)] & \text{by (6) of } |\_|^{\mathsf{s}} \\
= & |\mathtt{case}(P, \lambda x.(M\,Q), \lambda y.(N\,Q))|^{\mathsf{s}}_{c} & \text{by (9) of } |\_|^{\mathsf{s}}
\end{array}
$$

□

Proof of Theorem 3.2.6. The following measure reduces in every $\xrightarrow{\gamma}$-reduction.

$$
\begin{aligned}
[\![x]\!] &\stackrel{\text{def}}{=} 1 \\
[\![\lambda x.M]\!] &\stackrel{\text{def}}{=} [\![M]\!]+1 \\
[\![\langle M,N\rangle]\!] &\stackrel{\text{def}}{=} [\![M]\!]+[\![N]\!]+1 \\
[\![\mathtt{inl}(M)]\!] &\stackrel{\text{def}}{=} [\![M]\!]+1 \\
[\![\mathtt{inr}(M)]\!] &\stackrel{\text{def}}{=} [\![M]\!]+1
\end{aligned}
\qquad
\begin{aligned}
[\![M\,N]\!] &\stackrel{\text{def}}{=} [\![M]\!]*([\![N]\!]+1) \\
[\![\mathtt{fst}(M)]\!] &\stackrel{\text{def}}{=} 3[\![M]\!]+1 \\
[\![\mathtt{snd}(M)]\!] &\stackrel{\text{def}}{=} 3[\![M]\!]+1 \\
[\![\mathtt{case}(P,\lambda x.M,\lambda y.N)]\!] &\stackrel{\text{def}}{=} [\![P]\!]*([\![M]\!]+[\![N]\!]+1)
\end{aligned}
$$

□

Proof of Lemma 3.2.7. Both by induction on the structure of M. We first tackle three cases of (i).

Case $M \equiv \mathsf{Ax}(x,b)$:

$$
\begin{aligned}
& |\mathsf{Ax}(x,b)[x:=\langle a\rangle N]|^{\mathsf{n}} \\
=\ & |N[a\mapsto b]|^{\mathsf{n}} \\
\equiv\ & |N|^{\mathsf{n}} \\
\equiv\ & x[x:=|N|^{\mathsf{n}}] \\
=\ & |\mathsf{Ax}(x,b)|^{\mathsf{n}}[x:=|N|^{\mathsf{n}}] && \text{by (1) of } |\_|^{\mathsf{n}}
\end{aligned}
$$

Case $M \equiv \mathsf{Imp}_L(\langle b\rangle S,(y)T,z)$: Let $[\sigma]$ and $[\sigma]^{\mathsf{n}}$ be $[x:=\langle a\rangle N]$ and $[x:=|N|^{\mathsf{n}}]$, respectively.

$$
\begin{aligned}
& |\mathsf{Imp}_L(\langle b\rangle S,(y)T,z)[\sigma]|^{\mathsf{n}} \\
=\ & |\mathsf{Imp}_L(\langle b\rangle\, S[\sigma],(y)\,T[\sigma],z)|^{\mathsf{n}} \\
=\ & |T[\sigma]|^{\mathsf{n}}[y:=z\,|S[\sigma]|^{\mathsf{n}}] && \text{by (10) of } |\_|^{\mathsf{n}} \\
\equiv\ & |T|^{\mathsf{n}}[\sigma]^{\mathsf{n}}[y:=z\,|S|^{\mathsf{n}}[\sigma]^{\mathsf{n}}] && \text{by IH} \\
\equiv\ & |T|^{\mathsf{n}}[y:=z\,|S|^{\mathsf{n}}][\sigma]^{\mathsf{n}} \\
=\ & |\mathsf{Imp}_L(\langle b\rangle S,(y)T,z)|^{\mathsf{n}}[\sigma]^{\mathsf{n}} && \text{by (10) of } |\_|^{\mathsf{n}}
\end{aligned}
$$

Case $M \equiv \mathsf{Imp}_L(\langle b\rangle S,(y)T,x)$: Again, let $[\sigma]$ and $[\sigma]^{\mathsf{n}}$ be $[x:=\langle a\rangle N]$ and $[x:=|N|^{\mathsf{n}}]$, respectively.

$$
\begin{aligned}
& |\mathsf{Imp}_L(\langle b\rangle S,(y)T,x)[\sigma]|^{\mathsf{n}} \\
=\ & |\mathsf{Cut}(\langle a\rangle N,(x)\mathsf{Imp}_L(\langle b\rangle\, S[\sigma],(y)\,T[\sigma],x))|^{\mathsf{n}} \\
=\ & |T[\sigma]|^{\mathsf{n}}[y:=x\,|S[\sigma]|^{\mathsf{n}}][x:=|N|^{\mathsf{n}}] && \text{by (2,10) of } |\_|^{\mathsf{n}} \\
=\ & |T[\sigma]|^{\mathsf{n}}[y:=|N|^{\mathsf{n}}\ |S[\sigma]|^{\mathsf{n}}] && x\notin FV(|T[\sigma]|^{\mathsf{n}},|S[\sigma]|^{\mathsf{n}}) \\
\equiv\ & |T|^{\mathsf{n}}[\sigma]^{\mathsf{n}}[y:=|N|^{\mathsf{n}}\ |S|^{\mathsf{n}}[\sigma]^{\mathsf{n}}] && \text{by IH} \\
\equiv\ & |T|^{\mathsf{n}}[y:=x\,|S|^{\mathsf{n}}][\sigma]^{\mathsf{n}} \\
=\ & |\mathsf{Imp}_L(\langle b\rangle S,(y)T,x)|^{\mathsf{n}}[\sigma]^{\mathsf{n}} && \text{by (10) of } |\_|^{\mathsf{n}}
\end{aligned}
$$

We proceed with three cases of (ii). Notice that $a\in FC(M)$.

Case $M \equiv \mathsf{Ax}(x,a)$:

$$
\begin{aligned}
& |\mathsf{Ax}(y,a)[a:=(x)N]|^{\mathsf{n}} \\
=\ & |N[x\mapsto y]|^{\mathsf{n}} \\
\equiv\ & |N|^{\mathsf{n}}[x:=y] \\
=\ & |N|^{\mathsf{n}}[x:=|\mathsf{Ax}(y,a)|^{\mathsf{n}}] && \text{by (1) of } |\_|^{\mathsf{n}}
\end{aligned}
$$

Case $M \equiv \mathsf{And}_R(\langle c\rangle S, \langle d\rangle T, a)$: Notice that M has to freshly introduce a.

$$\begin{array}{ll} & |\mathsf{And}_R(\langle c\rangle S, \langle d\rangle T, a)[a := (x)N]|^{\mathsf{n}} \\ = & |\mathsf{Cut}(\langle a\rangle M, (x)N)|^{\mathsf{n}} \\ = & |N|^{\mathsf{n}}[x := |M|^{\mathsf{n}}] \qquad \text{by (2) of } |\_|^{\mathsf{n}} \end{array}$$

Case $M \equiv \mathsf{Cut}(\langle c\rangle S, (y)T)$:

$$\begin{array}{lll} & |\mathsf{Cut}(\langle c\rangle S, (y)T)[a := (x)N]|^{\mathsf{n}} & \\ = & |\mathsf{Cut}(\langle c\rangle S, (y)\,T[a := (x)N])|^{\mathsf{n}} & a \notin FC(\langle c\rangle S) \text{ because } M \in \mathfrak{J}^{(\supset,\wedge)} \\ \equiv & |T[a := (x)N]|^{\mathsf{n}}[y := |S|^{\mathsf{n}}] & \text{by (2) of } |\_|^{\mathsf{n}} \\ = & |N|^{\mathsf{n}}[x := |T|^{\mathsf{n}}][y := |S|^{\mathsf{n}}] & \text{by IH} \\ \equiv & |N|^{\mathsf{n}}[x := |T|^{\mathsf{n}}[y := |S|^{\mathsf{n}}]] & y \notin FN((x)N) \text{ by } [\_] \\ = & |N|^{\mathsf{n}}[x := |\mathsf{Cut}(\langle c\rangle S, (y)T)|^{\mathsf{n}}] & \text{by (2) of } |\_|^{\mathsf{n}} \end{array}$$

□

Proof of Proposition 3.2.8. The inductive steps are quite routine; therefore we give the calculations for four base cases only.

Logical Cut with Axiom:

$$\mathsf{Cut}(\langle a\rangle M, (x)\mathsf{Ax}(x,b)) \xrightarrow{int} M[a \mapsto b]$$

$$\begin{array}{lll} & |\mathsf{Cut}(\langle a\rangle M, (x)\mathsf{Ax}(x,b))|^{\mathsf{n}} & \\ = & x[x := |M|^{\mathsf{n}}] & \text{by (1,2) of } |\_|^{\mathsf{n}} \\ = & |M|^{\mathsf{n}} & \\ \equiv & |M[a \mapsto b]|^{\mathsf{n}} & \end{array}$$

Logical Cut $\wedge_R/\wedge_{L_1}$:

$$\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c), (y)\mathsf{And}^1_L((x)P, y)) \xrightarrow{int} \mathsf{Cut}(\langle a\rangle M, (x)P)$$

$$\begin{array}{lll} & |\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c), (y)\mathsf{And}^1_L((x)P, y))|^{\mathsf{n}} & \\ = & |P|^{\mathsf{n}}[x := \mathtt{fst}(y)][y := \langle |M|^{\mathsf{n}}, |N|^{\mathsf{n}}\rangle] & \text{by (2,3,7) of } |\_|^{\mathsf{n}} \\ \equiv & |P|^{\mathsf{n}}[x := \mathtt{fst}(\langle |M|^{\mathsf{n}}, |N|^{\mathsf{n}}\rangle)] & y \notin FV(|P|^{\mathsf{n}}) \text{ by logical cut } \wedge_R/\wedge_{L_1} \\ \xrightarrow{\iota}{}^{*} & |P|^{\mathsf{n}}[x := |M|^{\mathsf{n}}] & \\ = & |\mathsf{Cut}(\langle a\rangle M, (x)P)|^{\mathsf{n}} & \text{by (2) of } |\_|^{\mathsf{n}} \end{array}$$

Logical Cut $\supset_R/\supset_L$:

$$\begin{array}{l} \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle P, (y)Q, z)) \\ \quad \xrightarrow{int} \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle P, (x)M), (y)Q) \quad \text{or} \\ \quad \xrightarrow{int} \mathsf{Cut}(\langle c\rangle P, (x)\mathsf{Cut}(\langle a\rangle M, (y)Q)) \end{array}$$

For the first reduct we have:

$$\begin{array}{lll} & |\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle P, (y)Q, z))|^{\mathsf{n}} & \\ = & |Q|^{\mathsf{n}}[y := (z\ \ |P|^{\mathsf{n}})][z := \lambda x.|M|^{\mathsf{n}}] & \text{by (2,6,10) of } |\_|^{\mathsf{n}} \\ \equiv & |Q|^{\mathsf{n}}[y := (\lambda x.|M|^{\mathsf{n}})\ \ |P|^{\mathsf{n}}] & \text{by logical cut } z \notin FV(|Q|^{\mathsf{n}}, |P|^{\mathsf{n}}) \\ \xrightarrow{\iota}{}^{*} & |Q|^{\mathsf{n}}[y := |M|^{\mathsf{n}}[x := |P|^{\mathsf{n}}]] & \\ = & |\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle P, (x)M), (y)Q)|^{\mathsf{n}} & \text{by (2) of } |\_|^{\mathsf{n}} \end{array}$$

For the other reduct we have:

$$\begin{array}{lll} & |\mathsf{Cut}(\langle c\rangle P, (x)\mathsf{Cut}(\langle a\rangle M, (y)Q))|^{\mathsf{n}} & \\ = & |Q|^{\mathsf{n}}[y := |M|^{\mathsf{n}}][x := |P|^{\mathsf{n}}] & \text{by (2) of } |\_|^{\mathsf{n}} \\ \equiv & |Q|^{\mathsf{n}}[y := |M|^{\mathsf{n}}[x := |P|^{\mathsf{n}}]] & x \notin FV(|Q|^{\mathsf{n}}) \text{ by Remark 2.2.7} \end{array}$$

Commuting Cut:

$$\begin{array}{lll} \mathsf{Cut}(\langle a\rangle M,(x)N) & \xrightarrow{int} & M[a:=(x)N] \quad \text{or} \\ & \xrightarrow{int} & N[x:=\langle a\rangle M] \end{array}$$

$$|\mathsf{Cut}(\langle a\rangle M,(x)N)|^{\mathsf{n}} = |N|^{\mathsf{n}}[x:=|M|^{\mathsf{n}}] \equiv \begin{cases} |M[a:=(x)N]|^{\mathsf{n}} \\ |N[x:=\langle a\rangle M]|^{\mathsf{n}} \end{cases} \quad \text{by Lem. 3.2.7}$$

□

Proof of Lemma 3.4.1. First we notice that whenever $|M|^{\mathsf{S}}$ does not introduce a, then the substitution $[\_]$ is pushed inside the subterms and the lemma is by routine induction and calculation. So we are left with the cases where $|M|^{\mathsf{S}}$ introduces a. If M is of the form $\mathsf{Id}(y,a)$, we reason as follows.

Case $M \equiv \mathsf{Id}(y,a)$

$$\begin{array}{lll} & |\mathsf{Id}(y,a)|^{\mathsf{S}}[a:=(x)|N|^{\mathsf{S}}] & \\ = & \mathsf{Ax}(y,a)[a:=(x)|N|^{\mathsf{S}}] & \text{by (1) of } |\_|^{\mathsf{S}} \\ = & |N|^{\mathsf{S}}[x\mapsto y] & \\ \equiv & |N[x\mapsto y]|^{\mathsf{S}} & \\ = & |\mathsf{Id}(y,a)(\!|a:=(x)N|\!)|^{\mathsf{S}} & \text{by (2) of } (\!|\_|\!) \end{array}$$

For the other cases we illustrate the calculations with one example. Let $[\sigma]^{\mathsf{S}}$ and $(\!|\sigma|\!)$ be $[a:=(x)|N|^{\mathsf{S}}]$ and $(\!|a:=(x)N|\!)$, respectively.

Case $M \equiv \mathsf{And}_I(\langle c\rangle S,\langle b\rangle T,a)$

$$\begin{array}{lll} & |\mathsf{And}_I(\langle c\rangle S,\langle b\rangle T,a)|^{\mathsf{S}}[\sigma]^{\mathsf{S}} & \\ = & \mathsf{And}_R(\langle c\rangle|S|^{\mathsf{S}},\langle b\rangle|T|^{\mathsf{S}},a)[\sigma]^{\mathsf{S}} & \text{by (3) of } |\_|^{\mathsf{S}} \\ = & \mathsf{Cut}(\langle a\rangle\mathsf{And}_R(\langle c\rangle|S|^{\mathsf{S}}[\sigma]^{\mathsf{S}},\langle b\rangle|T|^{\mathsf{S}}[\sigma]^{\mathsf{S}},a),(x)|N|^{\mathsf{S}}) & \\ \equiv & \mathsf{Cut}(\langle a\rangle\mathsf{And}_R(\langle c\rangle|S(\!|\sigma|\!)|^{\mathsf{S}},\langle b\rangle|T(\!|\sigma|\!)|^{\mathsf{S}},a),(x)|N|^{\mathsf{S}}) & \text{by IH} \\ = & |\mathsf{Subst}(\langle a\rangle\mathsf{And}_I(\langle c\rangle S(\!|\sigma|\!),\langle b\rangle T(\!|\sigma|\!),a),(x)N)|^{\mathsf{S}} & \text{by (2,3) of } |\_|^{\mathsf{S}} \\ = & |\mathsf{And}_I(\langle c\rangle S,\langle b\rangle T,a)(\!|\sigma|\!)|^{\mathsf{S}} & \text{by (3) of } (\!|\_|\!) \end{array}$$

□

Proof of Lemma 3.4.2. Again, whenever $|M|^{\mathsf{S}}$ does not introduce x, then the substitution $[\_]$ is pushed inside the subterms and the lemma is by routine induction and calculation. So we are left with the cases where $|M|^{\mathsf{S}}$ introduces x. If M is of the form $\mathsf{Id}(x,b)$, we reason as follows.

Case $M \equiv \mathsf{Id}(x,b)$

$$\begin{array}{lll} & |\mathsf{Id}(x,b)|^{\mathsf{S}}[x:=\langle a\rangle|N|^{\mathsf{S}}] & \\ = & \mathsf{Ax}(x,b)[x:=\langle a\rangle|N|^{\mathsf{S}}] & \text{by (1) of } |\_|^{\mathsf{S}} \\ = & |N|^{\mathsf{S}}[a\mapsto b] & \\ \equiv & |N[a\mapsto b]|^{\mathsf{S}} & \\ = & |\mathsf{Id}(x,b)(\!|x:=\langle a\rangle N|\!)|^{\mathsf{S}} & \text{by (1) of } (\!|\_|\!) \end{array}$$

The only other cases which need to be considered are where $|M|^{\mathsf{S}}$ introduces x and is not an axiom. We illustrate the calculations with $M \equiv \mathsf{And}^i_E(\langle b\rangle\mathsf{Id}(x,b),(y)S)$. There are two subcases depending on whether N is an axiom that introduces a. Let $[\sigma]^{\mathsf{S}}$ and $(\!|\sigma|\!)$ be $[x := \langle a\rangle|N|^{\mathsf{S}}]$ and $(\!|x := \langle a\rangle N|\!)$, respectively.

Subcase $M \equiv \mathsf{And}^i_E(\langle b\rangle\mathsf{Id}(x,b),(y)S)$ and $N \equiv \mathsf{Id}(z,a)$

$$\begin{array}{rll}
& |\mathsf{And}^i_E(\langle b\rangle\mathsf{Id}(x,b),(y)S)|^{\mathsf{S}}[\sigma]^{\mathsf{S}} & \\
= & \mathsf{And}^i_L((y)|S|^{\mathsf{S}},x)[\sigma]^{\mathsf{S}} & \text{by (7) of } |_|^{\mathsf{S}} \\
= & \mathsf{Cut}(\langle a\rangle|\mathsf{Id}(z,a)|^{\mathsf{S}},(x)\mathsf{And}^i_L((y)|S|^{\mathsf{S}}[\sigma]^{\mathsf{S}},x)) & \text{by assumption} \\
\xrightarrow{cut}{}^{*} & \mathsf{Cut}(\langle a\rangle|\mathsf{Id}(z,a)|^{\mathsf{S}},(x)\mathsf{And}^i_L((y)|S(\!|\sigma|\!)|^{\mathsf{S}},x)) & \text{by IH} \\
\xrightarrow{cut} & \mathsf{And}^i_L((y)|S(\!|\sigma|\!)|^{\mathsf{S}},x)[x\mapsto z] & \\
\equiv & \mathsf{And}^i_L((y)|S(\!|\sigma|\!)|^{\mathsf{S}},z) & x \notin FN(|S(\!|\sigma|\!)|^{\mathsf{S}}) \\
= & |\mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(z,a),(y)S(\!|\sigma|\!))|^{\mathsf{S}} & \text{by (7) of } |_|^{\mathsf{S}} \\
= & |\mathsf{And}^i_E(\langle b\rangle\mathsf{Id}(x,b),(y)S)(\!|\sigma|\!)|^{\mathsf{S}} &
\end{array}$$

Otherwise the lemma follows by the calculation given below.

Subcase $M \equiv \mathsf{And}^i_E(\langle b\rangle\mathsf{Id}(x,b),(y)S)$ and $N \not\equiv \mathsf{Id}(z,a)$

$$\begin{array}{rll}
& |\mathsf{And}^i_E(\langle b\rangle\mathsf{Id}(x,b),(y)S)|^{\mathsf{S}}[\sigma]^{\mathsf{S}} & \\
= & \mathsf{And}^i_L((y)|S|^{\mathsf{S}},x)[\sigma]^{\mathsf{S}} & \text{by (7) of } |_|^{\mathsf{S}} \\
= & \mathsf{Cut}(\langle a\rangle|N|^{\mathsf{S}},(x)\mathsf{And}^i_L((y)|S|^{\mathsf{S}}[\sigma]^{\mathsf{S}},x)) & \text{by assumption} \\
\xrightarrow{cut}{}^{*} & \mathsf{Cut}(\langle a\rangle|N|^{\mathsf{S}},(x)\mathsf{And}^i_L((y)|S(\!|\sigma|\!)|^{\mathsf{S}},x)) & \text{by IH} \\
= & |\mathsf{And}^i_E(\langle a\rangle N,(y)S(\!|\sigma|\!))|^{\mathsf{S}} & \text{by (11) of } |_|^{\mathsf{S}} \\
= & |\mathsf{And}^i_E(\langle b\rangle\mathsf{Id}(x,b),(y)S)(\!|\sigma|\!)|^{\mathsf{S}} &
\end{array}$$

□

Proof of Lemma 3.4.3. As in Lemma 3.4.1, the difficult cases are where $|M|^{\mathsf{S}}$ introduces a. We illustrate the calculations with two examples. Let $\langle\!|\sigma|\!\rangle$ and $[\sigma]^{\mathsf{S}}$ be of the form $\langle\!|\mathsf{And}^i_E(\langle a\rangle\_,(x)N)|\!\rangle$ and $[a := (z)\mathsf{And}^i_L((x)|N|^{\mathsf{S}},z)]$, respectively.

Case $M \equiv \mathsf{Id}(y,a)$

$$\begin{array}{rll}
& |\mathsf{Id}(y,a)|^{\mathsf{S}}[\sigma]^{\mathsf{S}} & \\
= & \mathsf{Ax}(y,a)[a := (z)\mathsf{And}^i_L((x)|N|^{\mathsf{S}},z)] & \text{by (1) of } |_|^{\mathsf{S}} \\
= & \mathsf{And}^i_L((x)|N|^{\mathsf{S}},y) & \text{by assumption } z \notin FN((x)|N|^{\mathsf{S}}) \\
= & |\mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(y,a),(x)N)|^{\mathsf{S}} & \\
= & |\mathsf{Id}(y,a)\langle\!|\sigma|\!\rangle|^{\mathsf{S}} & \text{by (1) of } \langle\!|_|\!\rangle
\end{array}$$

Case $M \equiv \mathsf{And}_I(\langle c\rangle S,\langle d\rangle T,a)$

$$\begin{array}{rll}
& |\mathsf{And}_I(\langle c\rangle|S|^{\mathsf{S}},\langle d\rangle|T|^{\mathsf{S}},a)|^{\mathsf{S}}[\sigma]^{\mathsf{S}} & \\
= & \mathsf{And}_R(\langle c\rangle|S|^{\mathsf{S}},\langle d\rangle|T|^{\mathsf{S}},a)[\sigma]^{\mathsf{S}} & \text{by (3) of } |_|^{\mathsf{S}} \\
= & \mathsf{Cut}(\langle a\rangle\mathsf{And}_R(\langle c\rangle|S|^{\mathsf{S}}[\sigma]^{\mathsf{S}},\langle d\rangle|T|^{\mathsf{S}}[\sigma]^{\mathsf{S}},a),(z)\mathsf{And}^i_L((x)|N|^{\mathsf{S}},z)) & \\
\equiv & \mathsf{Cut}(\langle a\rangle\mathsf{And}_R(\langle c\rangle|S\langle\!|\sigma|\!\rangle|^{\mathsf{S}},\langle d\rangle|T\langle\!|\sigma|\!\rangle|^{\mathsf{S}},a),(z)\mathsf{And}^i_L((x)|N|^{\mathsf{S}},z)) & \text{by IH} \\
= & |\mathsf{And}^i_E(\langle a\rangle\mathsf{And}_I(\langle c\rangle S\langle\!|\sigma|\!\rangle,\langle d\rangle T\langle\!|\sigma|\!\rangle,a),(x)N)|^{\mathsf{S}} & \text{by (2,11) of } |_|^{\mathsf{S}} \\
= & |\mathsf{And}_I(\langle c\rangle S,\langle d\rangle T,a)\langle\!|\sigma|\!\rangle|^{\mathsf{S}} & \text{by (2) of } \langle\!|_|\!\rangle
\end{array}$$

□

Proof of Theorem 3.4.4. We give some illustrative examples.

Beta-Reduction:

$$\mathsf{Imp}_E(\langle b\rangle \mathsf{Imp}_I((x)\langle a\rangle M, b), \langle c\rangle N, (y)P) \xrightarrow{\beta} \mathsf{Subst}(\langle c\rangle N, (x)\mathsf{Subst}(\langle a\rangle M, (y)P))$$

$$\begin{array}{lll}
& |\mathsf{Imp}_E(\langle b\rangle \mathsf{Imp}_I((x)\langle a\rangle M, b), \langle c\rangle N, (y)P)|^{\mathsf{S}} & \\
= & \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle |M|^{\mathsf{S}}, b), (z)\mathsf{Imp}_L(\langle c\rangle |N|^{\mathsf{S}}, (y)|P|^{\mathsf{S}}, z)) & \text{by (5,13) of } |_|^{\mathsf{S}} \\
\xrightarrow{cut} & \mathsf{Cut}(\langle c\rangle |N|^{\mathsf{S}}, (x)\mathsf{Cut}(\langle a\rangle |M|^{\mathsf{S}}, (y)|P|^{\mathsf{S}})) & \\
& & z \text{ is freshly introduced by side-condition (13) of } |_|^{\mathsf{S}} \\
& & \text{and } b \text{ is freshly introduced by } \xrightarrow{\beta} \\
= & |\mathsf{Subst}(\langle c\rangle N, (x)\mathsf{Subst}(\langle a\rangle M, (y)P))|^{\mathsf{S}} & \text{by (2) of } |_|^{\mathsf{S}}
\end{array}$$

Commuting Reduction:

$$\mathsf{And}_E^i(\langle a\rangle M, (x)N) \xrightarrow{\gamma} M\{\!\!\{\mathsf{And}_E^i(\langle a\rangle\_, (x)N)\}\!\!\}$$

$$\begin{array}{lll}
& |\mathsf{And}_E^i(\langle a\rangle M, (x)N)|^{\mathsf{S}} & \\
= & \mathsf{Cut}(\langle a\rangle |M|^{\mathsf{S}}, (z)\mathsf{And}_L^i((x)|N|^{\mathsf{S}}, z)) & \text{by (11) of } |_|^{\mathsf{S}} \\
\xrightarrow{cut} & |M|^{\mathsf{S}}[a := (z)\mathsf{And}_L^i((x)|N|^{\mathsf{S}}, z)] & a \text{ is not freshly introduced by } \xrightarrow{\gamma} \\
\equiv & |M\{\!\!\{\mathsf{And}_E^i(\langle a\rangle_, (x)N)\}\!\!\}|^{\mathsf{S}} & \text{by Lemma 3.4.3}
\end{array}$$

Substitution Elimination:

$$\mathsf{Subst}(\langle a\rangle M, (x)N) \quad \xrightarrow{\sigma} \quad M[\![a := (x)N]\!]$$

$$\begin{array}{lll}
& |\mathsf{Subst}(\langle a\rangle M, (x)N)|^{\mathsf{S}} & \\
= & \mathsf{Cut}(\langle a\rangle |M|^{\mathsf{S}}, (x)|N|^{\mathsf{S}}) & \text{by (2) of } |_|^{\mathsf{S}} \\
\xrightarrow{cut} & |M|^{\mathsf{S}}[a := (x)|N|^{\mathsf{S}}] & a \text{ is not freshly introduced by } \xrightarrow{\sigma} \\
\equiv & |M[\![a := (x)N]\!]|^{\mathsf{S}} & \text{by Lemma 3.4.1}
\end{array}$$

$$\mathsf{Subst}(\langle a\rangle M, (x)N) \quad \xrightarrow{\sigma} \quad N[\![x := \langle a\rangle M]\!]$$

$$\begin{array}{lll}
& |\mathsf{Subst}(\langle a\rangle M, (x)N)|^{\mathsf{S}} & \\
= & \mathsf{Cut}(\langle a\rangle |M|^{\mathsf{S}}, (x)|N|^{\mathsf{S}}) & \text{by (2) of } |_|^{\mathsf{S}} \\
\xrightarrow{cut} & |N|^{\mathsf{S}}[x := \langle a\rangle |M|^{\mathsf{S}}] & x \text{ is not freshly introduced by } \xrightarrow{\sigma} \\
\xrightarrow{cut}{}^{*} & |N[\![x := \langle a\rangle M]\!]|^{\mathsf{S}} & \text{by Lemma 3.4.2}
\end{array}$$

Inner Reduction: The non-trivial cases are where M is translated using the clauses (11)–(14) of $|\_|^{\mathsf{S}}$, while N using (7)–(10). We give one case.

$$\mathsf{And}_E^i(\langle a\rangle S, (x)T) \xrightarrow{\kappa} \mathsf{And}_E^i(\langle a\rangle S', (x)T) \text{ with } S' \equiv \mathsf{Id}(y, a)$$

$$\begin{array}{lll}
& |\mathsf{And}_E^i(\langle a\rangle S, (x)T)|^{\mathsf{S}} & \\
= & \mathsf{Cut}(\langle a\rangle |S|^{\mathsf{S}}, (z)\mathsf{And}_L^i((x)|T|^{\mathsf{S}}, z)) & \text{by (11) of } |_|^{\mathsf{S}} \\
\xrightarrow{cut}{}^{*} & \mathsf{Cut}(\langle a\rangle |S'|^{\mathsf{S}}, (z)\mathsf{And}_L^i((x)|T|^{\mathsf{S}}, z)) & \text{by IH} \\
\xrightarrow{cut} & \mathsf{And}_L^i((x)|T|^{\mathsf{S}}, z)[z \mapsto y] & \\
\equiv & \mathsf{And}_L^i((x)|T|^{\mathsf{S}}, y) & z \notin FN(|T|^{\mathsf{S}}) \text{ by (11) of } |_|^{\mathsf{S}} \\
= & |\mathsf{And}_E^i(\langle a\rangle \mathsf{Id}(y, a), (x)T)|^{\mathsf{S}} \equiv |N|^{\mathsf{S}} & \text{by (7) of } |_|^{\mathsf{S}}
\end{array}$$

$\square$

Proof of Lemma 3.4.6. The cases corresponding to the clauses (1)–(5) and (14) of $|\_|^{\mathsf{N}}$ are routine. Below we give the details for the clauses (6) and (10). Let $[\sigma]$ and $\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}$ be $[x := \langle a\rangle N]$ and $\llparenthesis x := \langle a\rangle|N|^{\mathsf{N}}\rrparenthesis$, respectively.

Case $M \equiv \mathsf{And}^i_L((y)S, x)$

$$
\begin{array}{lll}
 & |\mathsf{And}^i_L((y)S, x)[\sigma]|^{\mathsf{N}} & \\
= & |\mathsf{Cut}(\langle a\rangle N, (x)\mathsf{And}^i_L((y)S[\sigma], x))|^{\mathsf{N}} & \\
= & \mathsf{And}^i_E(\langle a\rangle|N|^{\mathsf{N}}, (y)|S[\sigma]|^{\mathsf{N}}) & \text{by (10) of } |\_|^{\mathsf{N}} \\
\equiv & \mathsf{And}^i_E(\langle a\rangle|N|^{\mathsf{N}}, (y)|S|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}) & \text{by IH} \\
\equiv & \mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(x, a)\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}, (y)|S|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}) & \\
= & \mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(x, a), (y)|S|^{\mathsf{N}})\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \\
= & |\mathsf{And}^i_L((y)S, x)|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \text{by (6) of } |\_|^{\mathsf{N}}
\end{array}
$$

Case $M \equiv \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{And}^i_L((z)T, y))$

$$
\begin{array}{lll}
 & |\mathsf{Cut}(\langle b\rangle S, (y)\mathsf{And}^i_L((z)T, y))[\sigma]|^{\mathsf{N}} & \\
= & |\mathsf{Cut}(\langle b\rangle S[\sigma], (y)\mathsf{And}^i_L((z)T[\sigma], y))|^{\mathsf{N}} & \\
= & \mathsf{And}^i_E(\langle b\rangle|S[\sigma]|^{\mathsf{N}}, (z)|T[\sigma]|^{\mathsf{N}}) & \text{by (10) of } |\_|^{\mathsf{N}} \\
\equiv & \mathsf{And}^i_E(\langle b\rangle|S|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}, (z)|T|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}) & \text{by IH} \\
= & \mathsf{And}^i_E(\langle b\rangle|S|^{\mathsf{N}}, (z)|T|^{\mathsf{N}})\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \\
= & |\mathsf{Cut}(\langle b\rangle S, (y)\mathsf{And}^i_L((z)T, y))|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \text{by (10) of } |\_|^{\mathsf{N}}
\end{array}
$$

$\square$

Proof of Lemma 3.4.7. There are 23 cases to be considered, three of which are given below. We assume that $[\sigma]$ and $\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}}$ are of the form $[a := (x)\mathsf{And}^i_L((y)P, x)]$ and $\langle\!\langle\mathsf{And}^i_E(\langle a\rangle\_, (y)|P|^{\mathsf{N}})\rangle\!\rangle$, respectively.

Case $M \equiv \mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a)$

$$
\begin{array}{lll}
 & |\mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a)|^{\mathsf{N}}\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}} & \\
= & \mathsf{And}_I(\langle d\rangle|S|^{\mathsf{N}}, \langle e\rangle|T|^{\mathsf{N}}, a)\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}} & \text{by (2) of } |\_|^{\mathsf{N}} \\
= & \mathsf{And}^i_E(\langle a\rangle\mathsf{And}_I(\langle d\rangle|S|^{\mathsf{N}}\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}}, \langle e\rangle|T|^{\mathsf{N}}\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}}, a), (y)|P|^{\mathsf{N}}) & \text{by (2) of } \langle\!\langle\_\rangle\!\rangle \\
\xrightarrow{\kappa}{}^{*} & \mathsf{And}^i_E(\langle a\rangle\mathsf{And}_I(\langle d\rangle|S[\sigma]|^{\mathsf{N}}, \langle e\rangle|T[\sigma]|^{\mathsf{N}}, a), (y)|P|^{\mathsf{N}}) & \text{by IH} \\
= & |\mathsf{Cut}(\langle a\rangle\mathsf{And}_R(\langle d\rangle S[\sigma], \langle e\rangle T[\sigma], a), (x)\mathsf{And}^i_L((y)P, x))|^{\mathsf{N}} & \text{by (2,10) of } |\_|^{\mathsf{N}} \\
= & |\mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a)[\sigma]|^{\mathsf{N}} &
\end{array}
$$

Case $M \equiv \mathsf{Cut}(\langle b\rangle S, (z)\mathsf{And}^i_L((u)T, z))$

$$
\begin{array}{lll}
 & |\mathsf{Cut}(\langle b\rangle S, (z)\mathsf{And}^i_L((u)T, z))|^{\mathsf{N}}\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}} & \\
= & \mathsf{And}^i_E(\langle b\rangle|S|^{\mathsf{N}}, (u)|T|^{\mathsf{N}})\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}} & \text{by (10) of } |\_|^{\mathsf{N}} \\
= & \mathsf{And}^i_E(\langle b\rangle|S|^{\mathsf{N}}\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}}, (u)|T|^{\mathsf{N}}\langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}}) & \\
\xrightarrow{\kappa}{}^{*} & \mathsf{And}^i_E(\langle b\rangle|S[\sigma]|^{\mathsf{N}}, (u)|T[\sigma]|^{\mathsf{N}}) & \text{by IH} \\
= & |\mathsf{Cut}(\langle b\rangle S[\sigma], (z)\mathsf{And}^i_L((u)T[\sigma], z))|^{\mathsf{N}} & \text{by (10) of } |\_|^{\mathsf{N}} \\
= & |\mathsf{Cut}(\langle b\rangle S, (z)\mathsf{And}^i_L((u)T, z))[\sigma]|^{\mathsf{N}} &
\end{array}
$$

Case $M \equiv \mathsf{Cut}(\langle b\rangle S, (z)\mathsf{Ax}(z,a))$

$$
\begin{array}{rll}
 & |\mathsf{Cut}(\langle b\rangle S, (z)\mathsf{Ax}(z,a))|^{\mathsf{N}} \langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}} & \\
= & \mathsf{Subst}(\langle b\rangle |S|^{\mathsf{N}}, (z)\mathsf{Id}(z,a)) \langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}} & \text{by (1,14) of } |\_|^{\mathsf{N}} \\
= & \mathsf{Subst}(\langle b\rangle |S|^{\mathsf{N}} \langle\!\langle\sigma\rangle\!\rangle^{\mathsf{N}}, (z)\mathsf{And}^i_E(\langle a\rangle \mathsf{Id}(z,a), (y)|P|^{\mathsf{N}})) & \text{by (1) of } \langle\!\langle\_\rangle\!\rangle \\
\xrightarrow{\kappa}{}^{*} & \mathsf{Subst}(\langle b\rangle |S[\sigma]|^{\mathsf{N}}, (z)\mathsf{And}^i_E(\langle a\rangle \mathsf{Id}(z,a), (y)|P|^{\mathsf{N}})) & \text{by IH} \\
\xrightarrow{\kappa} & \mathsf{And}^i_E(\langle b\rangle |S[\sigma]|^{\mathsf{N}}, (y)|P|^{\mathsf{N}}) & z \notin FN((y)|P|^{\mathsf{N}}) \text{ by } \langle\!\langle\_\rangle\!\rangle \\
= & |\mathsf{Cut}(\langle b\rangle S[\sigma], (x)\mathsf{And}^i_L((y)P, x))|^{\mathsf{N}} & \text{by (10) of } |\_|^{\mathsf{N}} \\
= & |\mathsf{Cut}(\langle b\rangle S, (z)\mathsf{Ax}(z,a))[\sigma]|^{\mathsf{N}} &
\end{array}
$$

□

Proof of Lemma 3.4.8. Let $[\sigma]$ and $\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}$ be $[a := (x)N]$ and $\llparenthesis a := (x)|N|^{\mathsf{N}}\rrparenthesis$, respectively. There are 20 cases to be considered, two of which are given below.

Case $M \equiv \mathsf{And}_R(\langle c\rangle S, \langle d\rangle T, a)$

$$
\begin{array}{rll}
 & |\mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a)[\sigma]|^{\mathsf{N}} & \\
= & |\mathsf{Cut}(\langle a\rangle \mathsf{And}_R(\langle d\rangle S[\sigma], \langle e\rangle T[\sigma], a), (x)N)|^{\mathsf{N}} & \text{by ass. (10–13) of } |\_|^{\mathsf{N}} \text{ do not apply} \\
= & \mathsf{Subst}(\langle a\rangle \mathsf{And}_I(\langle d\rangle |S[\sigma]|^{\mathsf{N}}, \langle e\rangle |T[\sigma]|^{\mathsf{N}}, a), (x)|N|^{\mathsf{N}}) & \text{by (2,14) of } |\_|^{\mathsf{N}} \\
\equiv & \mathsf{Subst}(\langle a\rangle \mathsf{And}_I(\langle d\rangle |S|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}, \langle e\rangle |T|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}, a), (x)|N|^{\mathsf{N}}) & \text{by IH} \\
= & \mathsf{And}_I(\langle d\rangle |S|^{\mathsf{N}}, \langle e\rangle |T|^{\mathsf{N}}, a)\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \text{by (3) of } \llparenthesis\_\rrparenthesis \\
= & |\mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a)|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \text{by (2) of } |\_|^{\mathsf{N}}
\end{array}
$$

Case $M \equiv \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,a))$

$$
\begin{array}{rll}
 & |\mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,a))[\sigma]|^{\mathsf{N}} & \\
= & |\mathsf{Cut}(\langle b\rangle S[\sigma], (y)N[x \mapsto y])|^{\mathsf{N}} & \text{by assumtion (10–13) of } |\_|^{\mathsf{N}} \text{ do not apply} \\
= & \mathsf{Subst}(\langle b\rangle |S[\sigma]|^{\mathsf{N}}, (y)|N|^{\mathsf{N}}[x \mapsto y]) & \text{by (14) of } |\_|^{\mathsf{N}} \\
\equiv & \mathsf{Subst}(\langle b\rangle |S|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}, (y)|N|^{\mathsf{N}}[x \mapsto y]) & \text{by IH} \\
= & \mathsf{Subst}(\langle b\rangle |S|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}, (y)\mathsf{Id}(y,a)\llparenthesis\sigma\rrparenthesis^{\mathsf{N}}) & \\
= & \mathsf{Subst}(\langle b\rangle |S|^{\mathsf{N}}, (y)\mathsf{Id}(y,a))\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \\
= & |\mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,a))|^{\mathsf{N}}\llparenthesis\sigma\rrparenthesis^{\mathsf{N}} & \text{by (1,14) of } |\_|^{\mathsf{N}}
\end{array}
$$

□

Proof of Theorem 3.4.9. We first give the calculations for some base cases.

Logical Cut with Axiom:

$$M \equiv \mathsf{Cut}(\langle a\rangle S, (y)\mathsf{Ax}(y,b)) \xrightarrow{cut} S[a \mapsto b] \equiv N$$

$$
\begin{array}{rll}
 & |\mathsf{Cut}(\langle a\rangle M, (y)\mathsf{Ax}(y,b))|^{\mathsf{N}} & \\
= & \mathsf{Subst}(\langle a\rangle |S|^{\mathsf{N}}, (y)\mathsf{Id}(y,b)) & \text{by (1,14) of } |\_|^{\mathsf{N}} \\
\xrightarrow{\kappa} & |S|^{\mathsf{N}}[a \mapsto b] & \\
\equiv & |S[a \mapsto b]|^{\mathsf{N}} &
\end{array}
$$

$M \equiv \mathsf{Cut}(\langle a\rangle\mathsf{Ax}(y,a),(x)S) \xrightarrow{cut} S[x \mapsto y] \equiv N$ and one of the clauses (10–13) of $|\_|^{\mathsf{N}}$ applies to M, for example (10):

$$\begin{array}{rll} & |\mathsf{Cut}(\langle a\rangle\mathsf{Ax}(y,a),(x)S)|^{\mathsf{N}} & \\ = & \mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(y,a),(x)|S|^{\mathsf{N}}) & \text{by (1,10) of } |\_|^{\mathsf{N}} \\ \equiv & |N|^{\mathsf{N}} & \end{array}$$

Logical Cut $\wedge_R/\wedge_{L_1}$:

$M \equiv \mathsf{Cut}(\langle c\rangle\mathsf{And}_R(\langle a\rangle S,\langle b\rangle T,c),(y)\mathsf{And}^1_L((x)P,y)) \xrightarrow{cut} \mathsf{Cut}(\langle a\rangle S,(x)P) \equiv N$

$$\begin{array}{rll} & |\mathsf{Cut}(\langle c\rangle\mathsf{And}_R(\langle a\rangle S,\langle b\rangle T,c),(y)\mathsf{And}^1_L((x)P,y))|^{\mathsf{N}} & \\ = & \mathsf{And}^i_E(\langle c\rangle\mathsf{And}_I(\langle a\rangle|S|^{\mathsf{N}},\langle b\rangle|T|^{\mathsf{N}},c),(x)|P|^{\mathsf{N}}) & \text{by (2,10) of } |\_|^{\mathsf{N}} \\ \xrightarrow{\kappa} & \mathsf{Subst}(\langle a\rangle|S|^{\mathsf{N}},(x)|P|^{\mathsf{N}}) & c \text{ is freshly introduced by } \xrightarrow{cut} \end{array}$$

Now we have two subcases depending on whether P freshly introduces x, but is not an axiom. In this case we reason as follows.

$$\mathsf{Subst}(\langle a\rangle|S|^{\mathsf{N}},(x)|P|^{\mathsf{N}}) \xrightarrow{\kappa} |P|^{\mathsf{N}}\{\!\!\{x := \langle a\rangle|S|^{\mathsf{N}}\}\!\!\} \overset{(3.4.6)}{\equiv} |P[x := \langle a\rangle S]|^{\mathsf{N}} \equiv |N|^{\mathsf{N}}$$

Otherwise by (14) of $|\_|^{\mathsf{N}}$:

$$\mathsf{Subst}(\langle a\rangle|S|^{\mathsf{N}},(x)|P|^{\mathsf{N}}) = |\mathsf{Cut}(\langle a\rangle S,(x)P)|^{\mathsf{N}} \equiv |N|^{\mathsf{N}}$$

Logical Cut $\supset_R/\supset_L$:

$$\begin{array}{rl} M \equiv & \mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle S,b),(z)\mathsf{Imp}_L(\langle c\rangle T,(y)P,z)) \\ & \xrightarrow{cut} \mathsf{Cut}(\langle c\rangle T,(x)\mathsf{Cut}(\langle a\rangle S,(y)P)) \equiv N \end{array}$$

$$\begin{array}{rll} & |\mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle S,b),(z)\mathsf{Imp}_L(\langle c\rangle T,(y)P,z))|^{\mathsf{N}} & \\ = & \mathsf{Imp}_E(\langle b\rangle\mathsf{Imp}_I((x)\langle a\rangle|S|^{\mathsf{N}},b),\langle c\rangle|T|^{\mathsf{N}},(y)|P|^{\mathsf{N}}) & \text{by (4,12) of } |\_|^{\mathsf{N}} \\ \xrightarrow{\kappa} & \mathsf{Subst}(\langle c\rangle|N|^{\mathsf{N}},(x)\mathsf{Subst}(\langle a\rangle|M|^{\mathsf{N}},(y)|P|^{\mathsf{N}})) & b \text{ is freshly introduced by } \xrightarrow{cut} \end{array}$$

Again we have two subcases depending on whether P freshly introduces x, but is not an axiom. In this case, we reason analogous as in the case given above. Otherwise by (14) of $|\_|^{\mathsf{N}}$:

$$\begin{array}{rl} & \mathsf{Subst}(\langle c\rangle|N|^{\mathsf{N}},(x)\mathsf{Subst}(\langle a\rangle|M|^{\mathsf{N}},(y)|P|^{\mathsf{N}})) \\ = & |\mathsf{Cut}(\langle c\rangle N,(x)\mathsf{Cut}(\langle a\rangle M,(y)P))|^{\mathsf{N}} \equiv |N|^{\mathsf{N}} \end{array}$$

Commuting Cut:

$M \equiv \mathsf{Cut}(\langle a\rangle S,(x)T) \xrightarrow{cut} S[a := (x)T] \equiv N$

If one of the clauses (10–13) applies to M, then we have for example for $T \equiv \mathsf{And}^i_L((y)P,x)$

$$\begin{array}{rll} & |\mathsf{Cut}(\langle a\rangle S,(x)\mathsf{And}^i_L((y)P,x))|^{\mathsf{N}} & \\ = & \mathsf{And}^i_E(\langle a\rangle|S|^{\mathsf{N}},(y)|P|^{\mathsf{N}}) & \text{by (10) of } |\_|^{\mathsf{N}} \\ \xrightarrow{\kappa} & |S|^{\mathsf{N}}\{\!\!\{\mathsf{And}^i_E(\langle a\rangle\_,(y)|P|^{\mathsf{N}})\}\!\!\} & \\ \xrightarrow{\kappa}{}^{*} & |S[a := (x)T]|^{\mathsf{N}} \equiv |N|^{\mathsf{N}} & \text{by Lemma 3.4.7} \end{array}$$

Otherwise we have

$$\begin{array}{rll} & |\mathsf{Cut}(\langle a\rangle S,(x)T)|^{\mathsf{N}} & \\ = & \mathsf{Subst}(\langle a\rangle|S|^{\mathsf{N}},(x)|T|^{\mathsf{N}}) & \text{by (14) of } |\_|^{\mathsf{N}} \\ \xrightarrow{\kappa} & |S|^{\mathsf{N}}\llbracket a := (x)|T|^{\mathsf{N}}\rrbracket & \\ \equiv & |S[a := (x)T]|^{\mathsf{N}} & \text{by Lemma 3.4.8} \end{array}$$

$M \equiv \mathsf{Cut}(\langle a\rangle S,(x)T) \xrightarrow{cut} T[x := \langle a\rangle S] \equiv N$

$$\begin{array}{rll} & |\mathsf{Cut}(\langle a\rangle S,(x)T)|^{\mathsf{N}} & \\ = & \mathsf{Subst}(\langle a\rangle|S|^{\mathsf{N}},(x)|T|^{\mathsf{N}}) & \text{by (14) of } |\_|^{\mathsf{N}} \\ \xrightarrow{\kappa} & |T|^{\mathsf{N}}\llbracket x := \langle a\rangle|S|^{\mathsf{N}}\rrbracket & \\ \equiv & |T[x := \langle a\rangle S]|^{\mathsf{N}} \equiv |N|^{\mathsf{N}} & \text{by Lemma 3.4.6} \end{array}$$

Inner Reduction: The inner reductions are by simple induction arguments except where M is translated by clause (14) of $|\_|^{\mathsf{N}}$ and N by one of the clauses (10–13). We give the details for one such case. Suppose y is free in $(x)T$, but not in $(x)T'$.

$M \equiv \mathsf{Cut}(\langle c\rangle S,(y)\mathsf{And}^i_L((x)T,y)) \xrightarrow{cut} \mathsf{Cut}(\langle c\rangle S,(y)\mathsf{And}^i_L((x)T',y)) \equiv N$

$$\begin{array}{rll} & |\mathsf{Cut}(\langle c\rangle S,(y)\mathsf{And}^i_L((x)T,y))|^{\mathsf{N}} & \\ = & \mathsf{Subst}(\langle c\rangle|S|^{\mathsf{N}},(y)\mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(a,y),(x)|T|^{\mathsf{N}})) & \text{by (6,14) of } |\_|^{\mathsf{N}} \\ \xrightarrow{\kappa}{}^{*} & \mathsf{Subst}(\langle c\rangle|S|^{\mathsf{N}},(y)\mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(a,y),(x)|T'|^{\mathsf{N}})) & \text{by IH} \\ \xrightarrow{\kappa} & \mathsf{And}^i_E(\langle a\rangle\mathsf{Id}(a,y),(x)|T'|^{\mathsf{N}})\llbracket y := \langle c\rangle|S|^{\mathsf{N}}\rrbracket & \\ = & \mathsf{And}^i_E(\langle c\rangle|S|^{\mathsf{N}},(x)|T'|^{\mathsf{N}}) & \text{by assumption } y \notin FN((x)T') \\ = & |\mathsf{Cut}(\langle c\rangle S,(y)\mathsf{And}^i_L((x)T',y))|^{\mathsf{N}} \equiv |N|^{\mathsf{N}} & \text{by (10) of } |\_|^{\mathsf{N}} \end{array}$$

□

B.3 Proofs of Chapter 4

In this section we prove that a leftmost-outermost reduction strategy of $\xrightarrow{aux}$ does not restrict the collection of cut-free proofs reachable from a given proof.

Proof of Lemma 4.1.2. By induction on the structure of M_1. We analyse four representative cases.

- Suppose we have the term with the reduction sequence

$$\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c) \xrightarrow{bad} \mathsf{And}_R(\langle a\rangle M,\langle b\rangle S,c) \xrightarrow{\overline{aux}} \mathsf{And}_R(\langle a\rangle T,\langle b\rangle S,c)$$

  where a reduction is performed in the right subterm, although the left subterm is not yet normal. In this case we transform the reduction sequence into

$$\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c) \xrightarrow{\overline{aux}} \mathsf{And}_R(\langle a\rangle T,\langle b\rangle N,c) \xrightarrow{aux} \mathsf{And}_R(\langle a\rangle T,\langle b\rangle S,c)\ .$$

- Suppose we have a term with the reduction sequence

$$\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c) \xrightarrow{bad} \mathsf{And}_R(\langle a\rangle S,\langle b\rangle N,c) \xrightarrow{\overline{aux}} \mathsf{And}_R(\langle a\rangle T,\langle b\rangle N,c)$$

where first a bad reduction is performed in the left subterm and then a reduction in the same term which respects the constraints of the strategy. Thus we have the reduction sequence $M \xrightarrow{bad} S \xrightarrow{\overline{aux}} T$. By induction hypothesis there is a reduction $M \xrightarrow{\overline{aux}} R \xrightarrow{aux}{}^{*} T$, from which we obtain the reduction sequence

$$\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c) \xrightarrow{\overline{aux}} \mathsf{And}_R(\langle a\rangle R, \langle b\rangle N, c) \xrightarrow{aux}{}^{*} \mathsf{And}_R(\langle a\rangle T, \langle b\rangle N, c)\ .$$

- Suppose we have a term with the reduction sequence

$$\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{bad} \mathsf{Cut}(\langle a\rangle M, (x)S) \xrightarrow{\overline{aux}} M\{a := (x)S\}$$

  which we may transform into

$$\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{\overline{aux}} M\{a := (x)N\} \xrightarrow{aux}{}^{*} M\{a := (x)S\}$$

  using the fact that if $N \xrightarrow{aux} S$ then $M\{a := (x)N\} \xrightarrow{aux}{}^{*} M\{a := (x)S\}$.

- Suppose we have a term with reduction sequence

$$\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{bad} \mathsf{Cut}(\langle a\rangle S, (x)N) \xrightarrow{\overline{aux}} S\{a := (x)N\}$$

  which we may transform into

$$\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{\overline{aux}} M\{a := (x)N\} \xrightarrow{aux}{}^{*} S\{a := (x)N\}$$

  using Lemma 2.3.11. □

Proof of Theorem 4.1.3. Without loss of generality we may assume that the reduction sequence $M \xrightarrow{aux}{}^{*} N$ is of the form $M \xrightarrow{\overline{aux}}{}^{*} M_i \xrightarrow{aux}{}^{*} N$ where either $M_i \equiv N$ or $M_i \xrightarrow{bad} M_{i+1}$. The proof then proceeds by induction on the number of steps of the longest reduction sequence starting from M_i, i.e., on $\textsc{maxred}_{aux}(M_i)$. Since we prove the theorem for terms belonging to $\mathcal{T}$, we know their longest reduction sequences must be finite.

- If $\textsc{maxred}_{aux}(M_i) = 0$, then $M \xrightarrow{\overline{aux}}{}^{*} M_i \equiv N$ is a sequence which respects the restrictions imposed on $\xrightarrow{\overline{aux}}$.
- If $\textsc{maxred}_{aux}(M_i) > 0$, we first observe that the last reduction leading to a normal form must be a reduction of the form $\xrightarrow{\overline{aux}}$. Therefore the reduction sequence $M_i \xrightarrow{aux}{}^{*} N$ contains at least one $\xrightarrow{\overline{aux}}$. Let M_j be the leftmost term in this sequence from which a reduction $\xrightarrow{\overline{aux}}$ is performed. Hence the reduction sequence $M_i \xrightarrow{aux}{}^{*} N$ is of the form $M_i \xrightarrow{bad}{}^{+} M_j \xrightarrow{\overline{aux}} M_{j+1} \xrightarrow{aux}{}^{*} N$. By iteratively applying Lemma 4.1.2, we can move the reduction $\xrightarrow{\overline{aux}}$ next to M_i and obtain a reduction sequence of the form

$$M_i \xrightarrow{\overline{aux}} M' \xrightarrow{aux}{}^{*} M_{j+1} \xrightarrow{aux}{}^{*} N\ .$$

Without loss of generality we may assume the sequence $M' \xrightarrow{aux}{}^* M_{j+1} \xrightarrow{aux}{}^* N$ is of the form

$$M' \xrightarrow{\overline{aux}}{}^* M_k \xrightarrow{aux}{}^* N$$

where either $M_k \equiv N$ or $M_k \xrightarrow{bad} M_{k+1}$. This means M_k is the first term in this reduction sequence from which a bad reduction is performed. Clearly, we have $\textsc{maxred}_{aux}(M_i) > \textsc{maxred}_{aux}(M') \geq \textsc{maxred}_{aux}(M_k)$, and thus we can apply the induction hypothesis which gives us a reduction sequence $M' \xrightarrow{\overline{aux}}{}^* N$. We conclude this case with the reduction sequence

$$M \xrightarrow{\overline{aux}}{}^* M_i \xrightarrow{\overline{aux}} M' \xrightarrow{\overline{aux}}{}^* N\ .$$

Thus we are done. $\square$

Bibliography

Abramsky, S. [1993]. Computational Interpretations of Linear Logic. *Theoretical Computer Science*, 111(1–2):3–57.

Baader, F. and Nipkow, T. [1998]. *Term Rewriting and All That*. Cambridge University Press.

Barbanera, F. and Berardi, S. [1994]. A Symmetric Lambda Calculus for "Classical" Program Extraction. In *Theoretical Aspects of Computer Software*, volume 789 of *LNCS*, pages 495–515. Springer Verlag.

Barbanera, F., Berardi, S., and Schivalocchi, M. [1997]. "Classical" Programming-with-Proofs in λ^{sym}: An Analysis of Non-Confluence. In *Theoretical Aspects of Computer Software*, volume 1281 of *LNCS*, pages 365–390. Springer Verlag.

Barendregt, H. [1981]. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland.

Barendregt, H. and Ghilezan, S. [2000]. Theoretical Pearls: Lambda Terms for Natural Deduction, Sequent Calculus and Cut Elimination. *Journal of Functional Programming*, 10(1):121–134.

Benton, P. N. [1995]. Strong Normalisation for the Linear Term Calculus. *Journal of Functional Programming*, 5(1):65–80.

Benton, P. N., Bierman, G. M., de Paiva, V. C. V., and Hyland, J. M. E. [1993]. A Term Calculus for Intuitionistic Linear Logic. In *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 75–90. Springer-Verlag.

Berger, U., Buchholz, W., and Schwichtenberg, H. [2000]. Refined Program Extraction from Classical Proofs. *Annals of Pure and Applied Logic*. To appear.

Bierman, G. M. [1998]. A Computational Interpretation of the $\lambda\mu$-calculus. In *Mathematical Foundations of Computer Science*, volume 1450 of *LNCS*, pages 336–345. Springer-Verlag.

Bierman, G. M. [1999]. A Classical Linear λ-calculus. *Theoretical Computer Science*, 227(1–2):43–78.

Bittar, E. T. [1999]. Strong Normalisation Proofs for Cut-Elimination in Gentzen's Sequent Calculi. In *Logic, Algebra and Computer Science*, volume 46 of *Banach-Center Publications*, pages 179–225.

Bloo, R. [1997]. *Preservation of Termination for Explicit Substitution*. PhD thesis, Eindhoven University of Technology.

Bloo, R. and Geuvers, H. [1999]. Explicit Substitution: On the Edge of Strong Normalisation. *Theoretical Computer Science*, 211(1–2):375–395.

Boričić, B. R. [1985]. On Sequence-Conclusion Natural Deduction Systems. *Journal of Philosophical Logic*, 14(4):359–377.

Buss, S. R., editor [1998]. *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*. North-Holland.

Carlsson, M. [1984]. On Implementing Prolog in Functional Programming. *New Generation Computing*, 2(4):347–359.

Cellucci, C. [1992]. Existential Instantiation and Normalization in Sequent Natural Deduction. *Annals of Pure and Applied Logic*, 58:111–148.

Cichon, E. A., Rusinowitch, M., and Selhab, S. [1996]. Cut-Elimination and Rewriting: Termination Proofs. Technical Report.

Coquand, T. [1995]. A Semantics of Evidence of Classical Arithmetic. *Journal of Symbolic Logic*, 60(1):325–337.

Curry, H. B. and Feys, R. [1958]. *Combinatory Logic*, volume 1 of *Studies in Logic and the Foundations of Mathematics*. North-Holland.

Danos, V., Joinet, J.-B., and Schellinx, H. [1997]. A New Deconstructive Logic: Linear Logic. *Journal of Symbolic Logic*, 62(3):755–807.

Davey, B. A. and Priestley, H. A. [1990]. *Introduction to Lattices and Order*. Cambridge University Press.

de Bruijn, N. G. [1980]. A Survey of the Automath Project. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 580–606. Academic Press.

Dershowitz, N. [1982]. Orderings for Term Rewriting Systems. *Theoretical Computer Science*, 17:279–301.

Dragalin, A. G. [1988]. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society.

Dyckhoff, R. and Pinto, L. [1998]. Cut-Elimination and a Permutation-Free Sequent Calculus for Intuitionistic Logic. *Studia Logica*, 60(1):107–118.

Friedman, H. [1978]. Classically and Intuitionistically Provable Recursive Functions. In *Higher Set Theory*, volume 669 of *LNM*, pages 21–27. Springer-Verlag.

Gabbay, M. J. and Pitts, A. M. [1999]. A New Approach to Abstract Syntax Involving Binders. In *Logic in Computer Science*, pages 214–224. IEEE Computer Society Press.

Gallier, J. [1990]. On Girard's "Candidats de Reductibilité". In *Logic and Computer Science*, pages 123–203. Academic Press.

Gallier, J. [1993]. Constructive Logics. Part I: A Tutorial on Proof Systems and Typed λ-calculi. *Theoretical Computer Science*, 110(2):249–239.

Gentzen, G. [1935]. Untersuchungen über das logische Schließen I and II. *Mathematische Zeitschrift*, 39:176–210, 405–431.

Gentzen, G. [1936]. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112:493–565.

Gentzen, G. [1969]. *The Collected Papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North-Holland. English translation of Gentzen's papers edided by M. E. Szabo.

Girard, J.-Y. [1972]. *Interprétation Functionelle et Elimination des Coupures dans l'Arithmétique d'Ordre Supérieure*. PhD thesis, Université Paris VII.

Girard, J.-Y. [1987a]. Linear Logic. *Theoretical Computer Science*, 50(1):1–102.

Girard, J.-Y. [1987b]. *Proof Theory and Logical Complexity*. Studies in Proof Theory. Bibliopolis.

Girard, J.-Y. [1991]. A New Constructive Logic: Classical Logic. *Mathematical Structures in Computer Science*, 1(3):255–296.

Girard, J.-Y. [2001]. Locus Solum. *Mathematical Structures in Computer Science*, 11:301–506.

Girard, J.-Y., Lafont, Y., and Taylor, P. [1989]. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Griffin, T. [1990]. A Formulae-as-Types Notion of Control. In *Principles of Programming Languages*, pages 47–58. ACM Press.

Hennessy, M. C. B. and Ashcroft, E. A. [1980]. A Mathematical Semantics for a Non-Deterministic Typed λ-calculus. *Theoretical Computer Science*, 11(3):227–245.

Herbelin, H. [1994]. A λ-calculus Structure Isomorphic to Sequent Calculus Structure. In *Computer Science Logic*, volume 933 of *LNCS*, pages 67–75. Springer Verlag.

Herbelin, H. [1995]. *Séquents qu'on Calcule: De l'Interprétation du Calcul des Séquents comme Calcul de λ-termes et comme Calcul de Stratégies Gagnantes*. PhD thesis, Université Paris VII.

Hindley, J. R. [1969]. The Principal Type Scheme of an Object in Combinatory Logic. *Transactions of the American Mathematical Society*, 146:29–40.

Howard, W. A. [1980]. The Formulae-as-Types Notion of Construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press (the paper was written in 1969).

Hyland, J. M. E. [2000]. Proof Theory in the Abstract. *Annals of Pure and Applied Logic*. To appear.

Joinet, J.-B., Schellinx, H., and Tortora de Falco, L. [1998]. SN and CR for Free-Style LK^{tq}: Linear Decorations and Simulation of Normalisation. Preprint No. 1067, Utrecht University, Department of Mathematics.

Kleene, S. C. [1952a]. *Introduction to Metamathematics*. North-Holland.

Kleene, S. C. [1952b]. Permutability of Inferences in Gentzen's Calculi LK and LJ. *Memoirs of the American Mathematical Society*, 10:1–26.

Kreisel, G. [1958]. Mathematical Significance of Consistency Proofs. *Journal of Symbolic Logic*, 23(2):155–182.

Kreisel, G. [1971]. A Survey of Proof Theory II. In *Proceedings of the 2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 109–170. North-Holland.

Lambek, J. and Scott, P. J. [1988]. *Introduction to Higher Order Categorical Logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press.

Leivant, D. [1979]. Assumption Classes in Natural Deduction. *Zeitschrift für mathematische Logik*, 25:1–4.

McDowell, R. and Miller, D. [2000]. Cut-Elimination for a Logic with Definitions and Induction. *Theoretical Computer Science*, 232(1–2):91–119.

Melliès, P. A. [1995]. Typed Lambda Calculi with Explicit Substitutions May Not Terminate. In *Typed Lambda Calculi and Applications*, volume 902 of *LNCS*, pages 328–334. Springer Verlag.

Milner, R. [1978]. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17:348–375.

Mitchell, J. C. [1996]. *Foundations for Programming Languages*. Foundations of Computing. MIT Press.

Murthy, C. R. [1990]. *Extracting Constructive Content from Classical Proofs*. PhD thesis, Cornell University.

Parigot, M. [1991]. Free Deduction: An Analysis of "Computation" in Classical Logic. In *Conference on Logic Programming*, volume 592 of *LNCS*, pages 361–380. Springer Verlag.

Parigot, M. [1992]. $\lambda\mu$-calculus: An Algorithmic Interpretation of Classical Logic. In *Logic Programming and Automated Deduction*, volume 624 of *LNCS*, pages 190–201. Springer Verlag.

Pfenning, F. [1995]. Structural Cut-Elimination. In *Logic in Computer Science*, pages 156–166. IEEE Computer Society.

Pitts, A. M. and Gabbay, M. J. [2000]. A Metalanguage for Programming with Bound Names Modulo Renaming. In *Mathematics of Program Construction*, volume 1837 of *LNCS*, pages 230–255. Springer Verlag.

Pottinger, G. [1977]. Normalisation as Homomorphic Image of Cut-Elimination. *Annals of Mathematical Logic*, 12:323–357.

Prawitz, D. [1965]. *Natural Deduction: A Proof-Theoretical Study*. Almquist and Wiksell.

Prawitz, D. [1971]. Ideas and Results of Proof Theory. In *Proceedings of the 2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 235–307. North-Holland.

Reynolds, J. C. [1974]. Towards a Theory of Type Structure. In *Colloque sûr la Programmation*, volume 19 of *LNCS*, pages 408–425. Springer Verlag.

Rose, K. H. [1996]. Explicit Substitution: Tutorial & Survey. Technical report, BRICS, Department of Computer Science, University of Aarhus.

Schellinx, H. [1994]. *The Noble Art of Linear Decorating*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam. ILLC dissertation series.

Schütte, K. [1960]. *Beweistheorie*, volume 103 of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen*. Springer Verlag.

Schwichtenberg, H. [1999]. Termination of Permutative Conversions in Intuitionistic Gentzen Calculi. *Theoretical Computer Science*, 212(1–2):247–260.

Shoesmith, D. and Smiley, T. [1978]. *Multiple-Conclusion Logic*. Cambridge University Press.

Takeuti, G. [1975]. *Proof Theory*, volume 81 of *Studies in Logic and the Foundations of Mathematics*. North-Holland.

Troelstra, A. S. and Schwichtenberg, H. [1996]. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Ungar, A. M. [1992]. *Normalisation, Cut-Elimination and the Theory of Proofs*, volume 28 of *CLSI Lecture Notes*. Center for the Study of Language and Information.

Urban, C. [1998]. Implementation of Proof Search in the Imperative Programming Language Pizza. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1397 of *LNAI*, pages 313–319. Springer-Verlag.

Wadler, P. [1990]. Linear Types can Change the World! In *Working Conference on Programming Concepts and Methods*, pages 561–581. North-Holland.

Zucker, J. [1974]. The Correspondence Between Cut-Elimination and Normalisation. *Annals of Mathematical Logic*, 7:1–112.

Index