Weak and Strong Normalization, K-redexes, and First-Order Logic

by Peter Møller Neergaard

27th August 1999

Preface

Avoiding infinite loops is one of the obstacles most computer scientists must fight. Therefore the study of infinite loops and their opposite, termination, is important for our understanding of programs. This thesis studies certain aspects of these notions in the λ-calculus. Being the foundation of many modern functional languages it is a good theoretical framework for the study. The notions studied in this thesis have also interest due to the correspondence (through the Curry-Howard isomorphism) to proof normalisation in the mathematical field of proof theory. Amongst other applications, proof normalisation is useful, if not crucial, for proofs of consistency, *ie*, that only true sentences can be derived.

In the λ-calculus, a program is called a *term*, the single steps used to evaluate a term are called *reductions*, and a term that cannot be reduced is said to be in a *normal form*. The latter corresponds to a value. This thesis begins with a study of the notions *conservation* and *uniform normalisation*. Conservation means that infinite reduction paths, *ie*, non-terminating ways of reducing the term, are preserved under reduction. Uniform normalisation means that the term can either not be reduced to a normal form, or that all ways of reducing the term will eventually lead to a normal form.

The classical *conservation theorem* for Λ^I—a restriction of the basic λ-calculus, called Λ^K, where all abstraction variables occur free in the body of the abstraction—has been formulated using either of the notions conservation and uniform normalisation. The reason for the indistinctness in the formulation of the conservation theorem is that Λ^I is closed under β-reduction, which in Λ^I does not erase subterms. In this situation uniform normalisation implies conservation, and conservation also implies uniform normalisation. However, when turning to erasing reductions the distinction becomes important as conservation no longer implies uniform normalisation. This insight is elaborated in a new technique for finding uniformly normalising subsets of a λ-calculus. The technique uses a combination of a syntactic and a semantic criterion and it is applied in several ways:

- It is used to give a simple proof of a slightly weaker version of Bergstra and Klop's characterisation of perpetual redexes of Λ^K, *ie*, redexes that in any context preserve infinite reduction paths under reduction.
- The technique is used to find a new uniformly normalising subset of the basic λ-calculus. This uniformly normalising subset includes some terms with erasing redexes, so-called $\mathbf{K}$-redexes. This appears to be the first known uniformly normalising subset of Λ^K that is not also a subset of Λ^I.
- The technique is also used in a typed setting to find a uniformly normalising subset of the typed λ-calculus Λ_{M} corresponding to minimal first-order logic through the Curry-Howard isomorphism. This calculus is a first-order functional language with pairs, sums, and a very restricted from of dependent types where types can depend on terms that cannot be reduced.
- This uniformly normalising subset is used to infer strong normalisation from weak normalisation for Λ_{M}. There have been some proofs of strong normalisation of Λ_{M}

in the literature. The proof presented here is an extension of techniques developed recently by Sørensen and Xi to infer strong normalisation from weak normalisation of the same notion of reduction. In fact, strong normalisation is inferred from weak normalisation of a simpler form of reduction. The extension is non-trivial as Λ_{M} contains reductions on pairs and sums, which are erasing, as well as some permutative reductions.

Furthermore, the thesis considers the type system of classical first-order logic, which allows typing of exception handling. An analysis concludes that it is not possible to make a direct extension of the technique by Sørensen and Xi to this system.

To the best of the author's knowledge, none of these results have appeared in the literature before.

The author proves strong normalisation from weak normalisation for the simply typed λ-calculus extended with pairs in a previous report [MN99]. The last item is thus a major extension of the work presented in that report. Implicitly, the proof in the previous report depends on the technique to find uniformly normalising subsets presented in this thesis. The proof in the previous report and the two first items above have been submitted for publication in Information and Computation [MNS99] as joint work with the supervisor. However, the work presented in this thesis has been carried out by the author.

The thesis constitutes the master's thesis of Peter Møller Neergaard. It is submitted as partial fulfilment of the requirements for the Danish master's degree (cand. scient.) at DIKU, the Department of Computer Science at the University of Copenhagen. It reports work done at DIKU in the period of February 1999 through August 1999. The supervisor is Morten Heine Sørensen.

The thesis addresses a skilled graduate student in computer science. The reader is expected to have some basic knowledge of the λ-calculus and logic, for instance as presented in the course notes [SU98]. The reader is expected to have a developed understanding of mathematical reason as most of the thesis consists of proofs. A more precise presentation of the prerequisites is given below. Concerning the notation: Proofs are concluded using $\Box$ to mark the end of the proofs. Definitions are ended by $\blacklozenge$. Relations, *eg*, $\rightarrow$ and $\equiv$, are often annotated with a reference above the relation to the theorem or the equation used to establish the relation. Within proofs, equations that are important for the specific proof are tagged using $*$, $\star$, $\spadesuit$, and $\heartsuit$. These tags are reused in several proofs.

The project description (Danish: "arbejdsbeskrivelse") can be found in appendix A on p. 114.

Acknowledgements

Many people deserve thanks for their fruitful contributions during the project period.

- First and foremost, Morten Heine Sørensen who as tutor provided brilliant ideas and valuable references. I found great encouragement in his ever optimistic attitude towards the *happy* situation of running into problems. Furthermore, Morten pointed out one of the contributions of this thesis: the experimental confirmation of Hofstadter's law: "It always takes longer than you expect, even when you take into account Hofstadter's Law" [Hof79, p. 152].

- Amr Sabry and Matthias Felleisen who provided valuable up-to-date pointers for Call-by-Name CPS-translations and for discussions placing the thesis in the right context.

- My fellow students Jakob Grue Simonsen and Sebastian Skalberg for help in every conceivable, and sometimes inconceivable, way. In particular I am grateful that they allowed me to spammail them with all sorts of questions and thoughts.
- Kirsten Marie Øveraas who proofread several drafts while she in the true Knuthian way [KLR88, §1.13] replaced all the mathematics by "blah". I would also thank Peter Makholm for proofreading part of the thesis.
- Mette Kiellerup for advice on the layout and for providing a "professional looking binding" [Wat95, p. 96].
- Nils Andersen, Gilles Barthe, Olivier Danvy, Andrzej Filinski, John Hatcliff, Zurab Khasidashvili, Jan-Willem Klop, Vincent van Oostrom, Femke van Raamsdonk, and Gunnar Stålmarck for discussions.

Contents

Chapter 1

Introduction

In the classical λ-calculus Λ^K, the subset Λ^I is defined syntactically as the subset where all abstraction variables occur free in the body of the abstraction. Thus all arguments of a function are needed when evaluating the function. Therefore the evaluation of a function cannot terminate, if the evaluation of an argument does not terminate. This was formalised by Church [Chu41] in the *conservation theorem* of Λ^I. Traditionally there are two formulations of this theorem. The first formulation,

$$M \to_\beta N \;\&\; M \in \infty_\beta \Longrightarrow N \in \infty_\beta \ , \tag{1}$$

considers the reduction of the Λ^I-term M to the term N. It states that if there is an infinite reduction path from the term before the reduction, then there is also an infinite reduction path from the reduced term. The second formulation,

$$M \in \mathrm{WN}_\beta \Longrightarrow M \in \mathrm{SN}_\beta \ , \tag{2}$$

considers a single Λ^I-term M. It says that if the term is weakly normalising, *ie*, it has a reduction path leading to a normal form then it is also strongly normalising, *ie*, all reduction paths lead to a normal form. Strong normalisation trivially implies weak normalisation, and (2) states that the converse also holds. For a term M that is either both weakly and strongly normalising or none of the two, (1) holds: if $M \in \mathrm{WN}_\beta \cap \mathrm{SN}_\beta$ then $M \notin \infty_\beta$ so (1) holds vacuously; and if $M \notin \mathrm{WN}_\beta \cup \mathrm{SN}_\beta$ and $M \to_\beta N$ then $N \in \infty_\beta$ (otherwise $M \in \mathrm{WN}_\beta$) so again (1) holds. Thus, (2) implies (1).

Furthermore, as Λ^I is closed under β-reduction, (1) also implies (2): Assume that M is weakly, but not strongly, normalising. Then M has an infinite reduction path. We can now consider its reduction path to normal form,

$$\begin{array}{ccccccc} M_0 & & M_1 & & M_2 & \cdots & M_n \in \mathrm{NF}_\beta \\ \vdots & & \vdots & & \vdots & & \vdots \\ \multicolumn{7}{l}{\to_\beta} \end{array}$$

Each of the terms $M_1, \ldots, M_n$ are Λ^I-terms because of the closure under β-reduction. Therefore each of the terms in the path has an infinite reduction path. In particular, the normal form has an infinite reduction path, contradicting it being a normal form. This demonstrates that in a subset that is closed under a notion of reduction, the two formulations of the conservation theorem (1) and (2) are equivalent.

If we consider a subset that is not closed under reduction, we can no longer derive (2) from (1). It then becomes important to distinguish between the two notions. We

will therefore in this thesis reserve the phrase *conservation* for results similar to (1), while results similar to (2) are called *uniform normalisation*, following terminology by Khasidashvili.

A reduction is called *erasing* if subterms are thrown away as a part of the reduction. In the classical λ-calculus, this is reduction of λ-abstractions that do not use their argument, so-called $\mathbf{K}$-redexes. In other calculi there can be other erasing reductions. For instance in the λ-calculus extended with pairs and projections, projection is erasing. When considering erasing reductions it turns out to be difficult to find subsets that are closed under reduction and where all terms preserve infinite reduction paths under reduction. When including erasing reductions in a λ-calculus, it is therefore difficult to find subsets of the λ-calculus where all the terms are uniformly normalising.

We will demonstrate a technique for doing this by characterising a new subset of Λ^K where all terms are uniformly normalising. In contrast with the situation for Λ^I, a syntactic condition is no longer sufficient. We therefore depend on a combination of a syntactic criterion and a semantic criterion.

A λ-calculus is said to be weakly normalising, if all its terms are weakly normalisation; likewise with uniform and strong normalisation. One of the most important applications of *uniform normalisation* is to prove strong normalisation from weak normalisation of typed λ-calculi. The classical proof of strong normalisation of β-reduction in the simply typed λ-calculus is due to Tait [Tai67]. It was generalised to second-order typed λ-calculus by Girard [Gir72], and subsequently simplified by Tait [Tai75]. For some notions of reductions in typed λ-calculi it is simpler to prove weak normalisation than strong normalisation. For instance for the simply typed λ-calculus, the proofs presented by Turing [Gan80] and Prawitz [Pra65] of weak normalisation are substantially simpler than Tait-Girard's proof of strong normalisation.

It is, in fact, a conjecture presented by Barendregt at the 1995 *Typed Lambda-Calculus and Applications* conference in Edinburgh that this can be done for every pure type system [Bar92]. In a certain sense, the conjecture restates the Milner-Tofte slogan that "well-typed programs do not go wrong": For programming languages—or fragments of programming languages—where the conjecture holds we can be liberal on the use of reduction strategy as all reduction strategies will eventually lead to the same result[1].

Most people expect it to hold for all languages with a reasonable type system, but naturally the conjecture is most interesting for systems satisfying weak normalisation; otherwise, it is vacuously true. In particular it is not interesting for languages with full general recursion as they are not weakly normalising. Pathological examples are the ML-program

$$\text{fun } \mathbf{infinite}\, n \Rightarrow \mathbf{infinite}\, n$$

or the λ-term $(\lambda x.x\ x)\ (\lambda x.x\ x)$. However, there are other type systems allowing more restricted forms of recursion and where every program is weakly normalising, see for instance [Hof99]. The conjecture has also been stated by Geuvers [Geu93] and, in less concrete form by Klop. We will therefore refer to it as the *Barendregt-Geuvers-Klop conjecture*.

Over the years techniques to infer strong normalisation from weak normalisation have been developed by de Groote [dG93], Karr [Kar85], Kfoury and Wells [KW95], Khasidashvili [Kha91], Klop [Klo80], Nederpelt [Ned73], Sørensen [Sør97], and Xi [Xi97]. Most of these proofs depend on either conservation or uniform normalisation of Λ^I. A partial explanation is that, by definition weak normalisation implies strong normalisation in a uniformly normalising subset. The proofs consider calculi where abstraction is the only

[1] Strictly speaking, the conjecture only states that if we have weak normalisation, the all reduction strategies lead to a normal form, *ie*, the evaluation terminates. However, to give the same result, the notions of reduction should have the Church-Rosser property: for every term with a normal form, the normal form is unique. It is underlying the field that we mainly study confluent reductions.

constructor, which makes the conservation theorem for Λ^I applicable. To extend these techniques to calculi with other constructors, it is necessary to develop a technique to find uniformly normalising subsets of the new calculi. We are then through, if we can map all terms into terms of the uniformly normalising subset in a way that, in a certain sense, preserves reductions.

We demonstrate that our technique is useful for this purpose by inferring strong normalisation from weak normalisation for a λ-calculus with a type system corresponding to first-order minimal logic. This is an extension of the technique developed by Sørensen [Sør97] and Xi [Xi97] and on my previous work on the simply typed λ-calculus with pairs [MN99]. We will conclude with an discussion on the problem arising when trying to extend the technique to the type system corresponding to classical first-order logic, *ie*, a logic where one can deduce ϕ from $\neg\neg\phi$.

1.1 Outline

In outline this thesis is organised as follows:

Chapter 2 gives a simpler version of the condition for perpetuality of redexes in Λ^K than the characterisation by Bergstra and Klop [BK82]. A perpetual redex preserves infinite reduction paths. The condition by Bergstra and Klop gives a precise characterisation, while we only find a necessary condition. However, the proof is simpler, which makes it easier to extend to other calculi.

Chapter 3 presents a new uniformly normalising subset of Λ^K where we allow some erasing reductions. The subset is defined through a syntactic and a semantic condition. This chapter is based on the simpler version of Bergstra-Klop's Theorem, which was presented in the previous chapter.

Chapter 4 relates a system used by Stålmarck [Stå91] to prove strong normalisation of proof normalisation in classical first-order logic with a typed λ-calculus. We restrict us to the calculus Λ_{M} with a type system corresponding to first-order minimal logic and establish some basic properties.

Chapter 5 extends the notion of leftmost reduction to the calculus Λ_{M}. We use a technique by Takahashi [Tak95] to prove that it is normalising, *ie*, that if a term has a normal form, then leftmost reduction will eventually find this normal form.

Chapter 6 extends the technique of Chapter 3 and presents a uniformly normalising subset of Λ_{M} exploiting that leftmost reduction is normalising as established in the previous chapter.

Chapter 7 gives us the fruit of the hard work in the previous chapters. In this chapter we prove that we can infer strong normalisation from weak normalisation of Λ_{M}. We end the chapter by discussing the problems in relation to full classical first-order logic.

1.2 Preliminaries

This thesis is written with a graduate level student in mind. My intention is to address a student with the same background knowledge as I had when starting on the thesis. The subject of the thesis is normalisation in the λ-calculus. Furthermore, some comparisons with natural deduction of logical language are done. Thus, the full understanding of the

thesis depends on many notions and propositions, which it is outside the scope of this thesis to present in full. These notions and propositions are outlined in this section with references to comprehensive introductions; the first subsection considers λ-calculi, the second natural deduction.

1.2.1 Preliminaries on the λ-calculus

In this subsection we recall the basic definitions and propositions of the λ-calculus. Readers interested in a thorough introduction are referred to [Bar84]. For this purpose, precise references to [Bar84] are included in the margin of this subsection.

We assume to have an infinite countable supply of variables $x \in \mathbb{V}$. The set Λ^K is the type-free λ-terms defined by the grammar 2.1.1

$$\Lambda^K \ni M ::= x \mid \lambda x.M \mid M\ M \ .$$

The second form is called an *abstraction* and the third an *application*. We will use parentheses to improve clarity and remove ambiguities. Some example terms are $I = \lambda x.x$, $\omega = \lambda x.x\,x$, and $\Omega = \omega\,\omega$. By *convention*, application associates to the left, and λ-abstractions bind as far as possible to the right. Thus $\lambda x.x\ y\ z$ stands for $\lambda x.((x\ y)\ z)$. We will generally denote arbitrary terms by capitalised roman letters $K, L, M, \ldots$ and variables by small roman letters $x, y, z, u, v, \ldots$. In both cases we will use primes or subscripts when it is convenient. 2.1.25 6.2.1 2.1.3

By $M \subseteq N$ we denote that M is a *subterm* of the term N. Formally we have: 2.1.8

$$\begin{array}{c} M \subseteq M \\ M \subseteq \lambda x.P, \quad \text{if } M \subseteq P \\ M \subseteq P\,Q, \quad \text{if } M \subseteq P \\ M \subseteq P\,Q, \quad \text{if } M \subseteq Q \ . \end{array} \tag{3}$$

We recall that the subterm relation is reflexive and transitive. We use $\subset$ to denote the *proper subterm relation* defined by $M \subset N$, if, and only if, $M \subseteq N$ and $M \not\equiv N$.

The set of *free variables* and the set of *bound variables* of the term M are denoted $\mathrm{FV}(M)$ and $\mathrm{BV}(M)$, resp. The sets are defined recursively by:

$$\begin{array}{lcllcl} \mathrm{FV}(x) & = & \{x\} & \mathrm{BV}(x) & = & \emptyset \\ \mathrm{FV}(\lambda x.P) & = & \mathrm{FV}(P) \setminus \{x\} & \mathrm{BV}(\lambda x.P) & = & \mathrm{BV}(P) \cup \{x\} \\ \mathrm{FV}(P\,Q) & = & \mathrm{FV}(P) \cup \mathrm{FV}(Q) & \mathrm{BV}(P\,Q) & = & \mathrm{BV}(P) \cup \mathrm{BV}(Q) \ . \end{array} \tag{4}$$

The subset $\Lambda^I \subseteq \Lambda^K$ contains all terms M, such that for every subterm $\lambda x.P$ of M, it holds that $x \in \mathrm{FV}(P)$.

Two terms M and N are *α-congruent*, written $M \equiv_\alpha N$, if N results from M by a series of changes of bound variables, *eg*, 2.1.11(ii)

$$(\lambda x.x)\ \lambda y.y \equiv_\alpha (\lambda y.y)\ \lambda x.x \ .$$

We identify α-congruent terms on a syntactic level. This means that for a given term M, all terms that are α-congruent to M are considered to be the same. This is opposed to an identification on a semantic level where the terms would be identified through some (semantic) interpretation, for instance by identifying all terms evaluating to the same value. A formal presentation of this identification can be found in [SU98]. It can also be achieved through de Bruijn notation [dB72]. The equivalence between de Bruijn notation and our style with explicit names for bound variables is established in [Bar84, App. C]. We use $\equiv$ for *syntactic equality* up to α-equivalence.

As it is custom, we use the *variable convention*: given any term M we can choose an 2.1.13

α-congruent term M' such that the bound variables of M' do not overlap with a given finite set of variables. In particular, if $M_1, \ldots, M_n$ occurs in a certain mathematical context (*eg*, a definition, a proof), then in these terms all bound variables are chosen to be different from the free variables of any of the terms.

A *substitution* is a finite set of pairs $(x, N) \subseteq \mathbb{V} \times \Lambda^K$, written

$$\{x_1 := N_1, \ldots, x_n := N_n\} \ ,$$

where $x_i \neq x_j$ for $i \neq j$. Arbitrary term substitutions are denoted by θ, ϕ, or σ.

Though not standard, the following notions are convenient shorthands:

Definition 1.1 For a substitution θ the *domain* $\operatorname{dom}\theta$, the *set of free variables* $\mathrm{FV}(\theta)$, the *subterm relation* $\subseteq$, the *restriction* $\theta_{|V}$ to the variables V, and the *exclusion* $\theta_{|V}$ of the variables V, resp., are defined as follows:

$$\begin{array}{lcl} \operatorname{dom}\theta & = & \{x \mid x := N \in \theta\} \\ \mathrm{FV}(\theta) & = & \bigcup\{\mathrm{FV}(N) \mid x := N \in \theta\} \\ M \subseteq \theta & \iff & M \subseteq N \text{ for some } x := N \in \theta \\ \theta_{|V} & = & \{x := N \mid x := N \in \theta, x \in V\} \\ \theta_{\setminus V} & = & \{x := N \mid x := N \in \theta, x \notin V\} \end{array}$$

We write $\theta_{|x}$ and $\theta_{\setminus x}$ when $V = \{x\}$. ♦

We also define capture avoiding substitution in a term.

Definition 1.2 *Simultaneous substitution* of a substitution θ in a term M, written $M\theta$, is defined recursively by the following equations: 2.4.8

$$\begin{array}{lcl} x\theta & = & \begin{cases} N & \text{if } x := N \in \theta \\ x & \text{if } x \notin \operatorname{dom}\theta \end{cases} \\ (\lambda x.P)\theta & = & \lambda x.P\theta \\ (P\,Q)\theta & = & (P\theta)\,(Q\theta) \end{array}$$

♦

It is important to recognise the rôle of the variable convention in this definition: it makes it unnecessary to assume, say, $x \notin (\mathrm{FV}(\theta) \cup \operatorname{dom}\theta)$ in the case for $\lambda x.P$! 2.1.15

The following is a generalisation of the substitution lemma [Bar84, 2.1.16].

Proposition 1.3 *Let θ and ϕ be substitutions and M a Λ^K-term. Then*

$$(M\theta)\phi \equiv M\sigma$$

where

$$\sigma = \{x := P\phi \mid x := P \in \theta\} \cup \{x := P \mid x := P \in \phi_{\setminus \operatorname{dom}\theta}\} \ .$$

This proposition is also used as the definition of the *composition* of two substitutions: for any two substitutions θ and ϕ, the composition $\theta\phi$ is the substitution σ described by the proposition.

We state in passing the following proposition:

Proposition 1.4 (Substitution generation lemma) *Let $M, N \in \Lambda^K$ be terms and θ a substitution. If $N \subseteq M\theta$ and $N \not\subseteq \theta$, then*

i. if $N \equiv \lambda x.P$, there exists $N' \equiv \lambda y.P' \subseteq M$ such that $N \equiv N'\theta$.

ii. if $N \equiv P\,Q$, there exists $N' \equiv P'\,Q' \subseteq M$ such that $N \equiv N'\theta$.

We recall that a *notion of reduction* is a binary relation on Λ^K. 3.1.2

Definition 1.5 We define the following notion of reduction on Λ^K (ordinary β*-reduction*): 3.1.3

$$(\lambda x.P)\,Q \quad \beta_\lambda \quad P\{x := Q\}\ .$$

The resulting term $P\{x := Q\}$ is the *contractum* of the redex $(\lambda x.P)\,Q$. The abstraction $\lambda x.P$ is called the *operator* of the redex. An abstraction $\lambda x.P$ is an I*-abstraction* and a β_λ-redex $(\lambda x.P)\,Q$ is an I*-redex*, resp., if, and only if, $x \in \mathrm{FV}(P)$. We call a β_λ-abstraction $\lambda x.P$ a $\mathbf{K}$*-abstraction* and a redex $(\lambda x.P)\,Q$ a $\mathbf{K}$*-redex*, resp., if, and only if, $x \notin \mathrm{FV}(P)$. In the latter cases we write $\mathbf{K}\,P$ and $\mathbf{K}\,P\,Q$, resp. ♦

When there is no confusion we will leave out λ from β_λ. We call application $P\,Q$ a *destructor* of β-redexes, and abstraction $\lambda x.P$ a *constructor* of β-redexes. As the notions of reductions under consideration should be clear from the context, we use Λ^K to refer to the *calculus* containing the terms of Λ^K and the notions of reductions on Λ^K.

A *context* C is a Λ^K-term with a hole $[]$ instead of a subterm. It is defined by the following grammar 2.1.18

$$C ::= [] \mid \lambda x.C \mid C\,M \mid M\,C\ .$$

The result of placing a term M in the hole of a context C is written $C[M]$. In this process free variables might become bound, but we will tacitly assume that this is not the case. We extend the *subterm relation* to contexts in the following way:

$$\begin{aligned} P \subseteq C\,M &\Longleftarrow P \subseteq M \\ P \subseteq M\,C &\Longleftarrow P \subseteq M \\ P \subseteq \lambda x.C &\Longleftarrow P \subseteq C \\ P \subseteq C\,M &\Longleftarrow P \subseteq C \\ P \subseteq M\,C &\Longleftarrow P \subseteq C\ . \end{aligned} \tag{5}$$

We recall that a binary relation $\mathbf{R}$ is *compatible*, if, and only if, 3.1.1

$$(M, N) \in \mathbf{R} \Longrightarrow (C[M], C[N]) \in \mathbf{R}$$

for all terms $M, N \in \Lambda^K$ and contexts C.

Definition 1.6 Let $\mathbf{R}$ be a notion of reduction. We define the following notions.

i. $\to_{\mathbf{R}}$ is the compatible closure, $\twoheadrightarrow_{\mathbf{R}}$ is the compatible, reflexive, transitive closure, $\twoheadrightarrow^+_{\mathbf{R}}$ is the compatible, transitive closure, and $=_{\mathbf{R}}$ is the equivalence relation induced by $\twoheadrightarrow_{\mathbf{R}}$. 3.1.5

ii. A finite or infinite sequence 3.1.7

$$M_0 \to_{\mathbf{R}} M_1 \to_{\mathbf{R}} \cdots$$

is called an $\mathbf{R}$*-reduction path* from M_0. We say that M_0 *has* this $\mathbf{R}$-reduction path. If the sequence is finite it *ends* in the last term M_n and has *length* n. 3.1.18

iii. We define the following sets: 3.1.22

$$\begin{aligned} \infty_{\mathbf{R}} &= \{M \mid M \text{ has an infinite } \mathbf{R}\text{-reduction path}\} \\ \mathrm{NF}_{\mathbf{R}} &= \{M \mid M \text{ has no } \mathbf{R}\text{-reduction path with length 1 or more}\} \\ \mathrm{SN}_{\mathbf{R}} &= \{M \mid M \text{ has no infinite } \mathbf{R}\text{-reduction path}\} \\ \mathrm{WN}_{\mathbf{R}} &= \{M \mid M \text{ has a finite } \mathbf{R}\text{-reduction path}\} \\ \mathrm{UN}_{\mathbf{R}} &= \{M \mid M \in \mathrm{WN}_{\mathbf{R}} \Rightarrow M \in \mathrm{SN}_{\mathbf{R}}\} \\ \mathrm{CON}_{\mathbf{R}} &= \{M \mid \forall N : (M \to_{\mathbf{R}} N \,\&\, M \in \infty_{\mathbf{R}}) \Rightarrow N \in \infty_{\mathbf{R}}\} \end{aligned}$$

The set $\infty_{\mathbf{R}}$ is the *infinite terms* under $\mathbf{R}$; the set $\mathrm{NF}_{\mathbf{R}}$ is the *normal forms* of $\mathbf{R}$; the set $\mathrm{SN}_{\mathbf{R}}$ is the *strongly normalising* terms under $\mathbf{R}$; the set $\mathrm{WN}_{\mathbf{R}}$ is the *weakly normalising* terms under $\mathbf{R}$; the set $\mathrm{UN}_{\mathbf{R}}$ is the *uniformly normalising* terms under $\mathbf{R}$; and the set $\mathrm{CON}_{\mathbf{R}}$ is the set of terms satisfying the *conservation property* under $\mathbf{R}$.

iv. An $\mathbf{R}$-*reduction strategy* is a map $F : \Lambda^K \to \Lambda^K$ satisfying $M \to_{\mathbf{R}} F(M)$ for all $M \in \Lambda^K \setminus \mathrm{NF}_{\mathbf{R}}$, and $F(M) = M$ for all $M \in \mathrm{NF}_{\mathbf{R}}$. ♦

It is seen that $M \in \mathrm{SN}_{\mathbf{R}}$, if, and only if, $M \notin \infty_{\mathbf{R}}$. Furthermore, we note that $M \in \mathrm{NF}_{\mathbf{R}}$ implies $M \in \mathrm{SN}_{\mathbf{R}}$ and $M \in \mathrm{SN}_{\mathbf{R}}$ implies $M \in \mathrm{WN}_{\mathbf{R}}$. We will omit "under $\mathbf{R}$" in the above notions most of the time as $\mathbf{R}$ will be clear from the context. *Par abus de langage*, we refer to a subset of $\mathrm{UN}_{\mathbf{R}}$, say, as a uniformly normalising subset.

The following lemma states the reasoning done in the start of this chapter.

Lemma 1.7 *Let $\mathbf{R}$ be a notion of reduction on a set L, and let $L' \subseteq L$.*

i. if $L' \subseteq \mathrm{UN}_{\mathbf{R}}$, then $L' \subseteq \mathrm{CON}_{\mathbf{R}}$.

ii. if L' is closed under $\mathbf{R}$ and $L' \subseteq \mathrm{CON}_{\mathbf{R}}$, then $L' \subseteq \mathrm{UN}_{\mathbf{R}}$.

We extend $\to_{\mathbf{R}}$, $\twoheadrightarrow_{\mathbf{R}}$, and $\twoheadrightarrow^+_{\mathbf{R}}$ to arbitrary contexts C and C' in the same way as we extended the subterm relation. Taking $\to_{\mathbf{R}}$ as an example we have

$$\begin{aligned} C\,M \to_{\mathbf{R}} C\,N &\Longleftarrow M \to_{\mathbf{R}} N \\ M\,C \to_{\mathbf{R}} N\,C &\Longleftarrow M \to_{\mathbf{R}} N \\ \lambda x.C \to_{\mathbf{R}} \lambda x.C' &\Longleftarrow C \to_{\mathbf{R}} C' \\ C\,M \to_{\mathbf{R}} C'\,M &\Longleftarrow C \to_{\mathbf{R}} C' \\ M\,C \to_{\mathbf{R}} M\,C' &\Longleftarrow C \to_{\mathbf{R}} C' \end{aligned} \quad .$$

From the definition of $\to_{\mathbf{R}}$ we see that $C \to_{\mathbf{R}} C'$ implies $C[M] \to_{\mathbf{R}} C'[M]$ for all contexts C and C' and all terms M. We say that a context C is infinite, written $C \in \infty_{\mathbf{R}}$, if $M \in \infty_{\mathbf{R}}$ for some subterm M of C.

The following three lemmata are used extensively:

Lemma 1.8 *Let M and N be Λ^K-terms such that $M \to_{\mathbf{R}} N$. Then* p. 54

$$K\{x := M\} \twoheadrightarrow_{\mathbf{R}} K\{x := N\}$$

for all terms $K \in \Lambda^K$ and all variables x.

Corollary 1.9 *Let M be an infinite term and θ any substitution. Then the term $M\theta$ is infinite.*

As this Corollary cannot be found in [Bar84] we include the proof.

Proof. The preceding lemma is trivially extended to an arbitrary substitution θ by induction on the number of pairs in the substitution. We thus have that $M \to_{\mathbf{R}} N$ implies $M\theta \to_{\mathbf{R}} N\theta$. If the term M is infinite, we have a reduction path

$$M \equiv M_0 \to_{\mathbf{R}} M_1 \to_{\mathbf{R}} \cdots M_n \to_{\mathbf{R}} \cdots \quad .$$

Using the lemma on each of these reductions we have the infinite reduction path

$$M\theta \equiv M_0\theta \to_{\mathbf{R}} M_1\theta \to_{\mathbf{R}} \cdots M_n\theta \to_{\mathbf{R}} \cdots \quad .$$

□

Lemma 1.10 *Let M and N be Λ^K-terms such that $M \to_\beta N$. Then* 3.1.16

$$M\{x := K\} \to_\beta N\{x := K\}$$

for all terms $K \in \Lambda^K$ and all variables x.

Lemma 1.11 *Let M be a Λ^K-term and θ a substitution. Then*

$$\mathrm{FV}(M\theta) = \big(\mathrm{FV}(M) \setminus (\mathrm{dom}\,\theta \cap \mathrm{FV}(M))\big) \cup \mathrm{FV}(\theta_{|\mathrm{FV}(M)})$$

Proof. By induction on the structure of M. □

The above notions can be carried over to the other calculi presented in the thesis, *mutatis mutandis*. We will only give the formal definitions and the proofs in case the extensions are non-trivial.

1.2.2 Preliminaries on Natural Deduction

Natural deduction is described comprehensively by Prawitz [Pra65], but the reader might also find it worth consulting [Pra71, GLT89, SU98]. In this subsection we recall the basic definitions needed to understand this thesis. Precise references to [Pra65] are included in the margin of this subsection.

Natural deduction formalises the natural way of reasoning in a language. The language determines a set of formulae denoted by A. The way of reasoning is described by inference rules of the form p. 13 p. 22–23

$$\frac{A_1 \quad \cdots \quad A_n}{A}$$

that states that the formula A can be inferred from the formulae $A_1, \ldots, A_n$. The formulae $A_1, \ldots, A_n$ are called *premises* and the formula A is denoted the *conclusion*. In some cases it is necessary to state some *side conditions*, *eg*, about variables in the formulae, restricting the use of the inference rule. An inference rule with its possible side conditions is called a *deduction rule*. The set of deduction rules describes the *atomic deductions* in a system of natural deduction.

In this thesis we work with classical first-order logic described by the symbol $\bot$, the *connectives* $\to$, $\wedge$, and $\vee$, the *quantifiers* $\forall$ and $\exists$, and the *relations* r on first-order terms t. The first-order terms are also called the *eigenterms*. We present all the deduction rules and discuss the rôle of the eigenterms in Section 4.2. For the moment, we will only consider the connectives $\to$ and $\forall$, as this is sufficient to establish the notation. The *formulae* built from these connectives are described by the grammar p. 13–15

$$\varphi, \psi ::= x \mid r \mid \varphi \to \varphi \mid \forall \alpha.\varphi$$

where α denotes variables drawn from a infinite, countable set. In the two last forms the connective $\to$ and the quantifier $\forall$, resp., is denoted the *principal sign* of the formula. p. 15

The deduction rules are used to build proofs of first-order formulae; intuitively, the theorems are the formulae where a proof can be found without doing any assumptions. To avoid indistinctness we will denote proofs deduced by deduction rules *deductions*. p. 24 We will denote arbitrary deductions by Σ or Π, possibly primed or indexed. A deduction starts from a set of formulae, called the assumptions. Other formulae are deduced from the assumptions using the deduction rules. This process might lead to the same formula occuring several places in the deduction; in particular, the same assumption might be used to start several subdeductions. We therefore distinguish between a formula and a *formula occurrence* where the latter is a specific instance of a given formula in a p. 25

deduction.

When stating the specific deduction rules we have to consider some additional problems. We first consider the connective $\rightarrow$ describing implication. The "natural" interpretation is that we prove $A \rightarrow B$ by assuming A and then prove B, and that we can deduce B from $A \rightarrow B$ by proving A. This is described in the deduction rules: p. 20

$$\frac{\begin{array}{c}(A)\\ B\end{array}}{A \rightarrow B}\ {\rightarrow}I \qquad \frac{A \rightarrow B \quad A}{B}\ {\rightarrow}E\ .$$

The notation (A) means that zero, one, or more occurrences of the assumption A in the deduction of B are *discharged* or *closed* by this instance of the deduction rule. It should be stressed that the inference rule does not necessarily discharge all occurrences of the assumption A. If all occurrences of an assumption have been discharged, the assumption is no longer considered *open*. When considering a specific deduction, we say that it *depends* on its open assumptions. p. 17–18 p. 24

We now turn to the quantifier $\forall$, corresponding to universal quantification. The "natural" interpretation is that we can deduce $\forall\alpha.A$, if we can prove the formula A (possibly containing the variable α) without referring to the specific value of α. Furthermore, if we know $\forall\alpha.A$, the formula A is in particular true when we replace all instances of α in A by an eigenterm t. This is described by the deduction rules: p. 20

$$\frac{A}{\forall\alpha.A}\ \forall I \qquad \frac{\forall\alpha.A}{A_t^\alpha}\ \forall E\ .$$

In these rules, A_t^α denotes the substitution of the eigenterm t for all free occurrences of the variable α in the formula A. p. 15

For both the connective and the quantifier we have rules *introducing* the connective or the quantifier, *ie*, the principal sign of the deduced formula is always the connective, and we have rules *eliminating* the connective or the quantifier, *ie*, the rules allow us to deduce other formulae from a formula with the connective or the quantifier as the principal sign. In the elimination rules, we call the premises without the connective or quantifier *minor*; all other premises are *major*. Furthermore, all premises of the introduction rules are also major. By convention, we write major premises as the leftmost premises. p. 19 p. 22

In Section 4.2 we present some reduction rules for deductions in natural deduction. In some of these reduction rules we will replace some of the occurrences of an assumption by a deduction of the assumption. As an example, we can consider the following reduction rule for $\rightarrow$: p. 37

$$\frac{\dfrac{\dfrac{\begin{array}{c}[A]\\ \Sigma_1\end{array}}{B}}{A \rightarrow B} \qquad \dfrac{\Sigma_2}{A}}{B} \quad \rightarrow \quad \frac{\begin{array}{c}\Sigma_2\\ [A]\\ \Sigma_1\end{array}}{B}\ .$$

The notation p. 26

$$\begin{array}{c}[A]\\ \Sigma_1\end{array}$$

denotes that there are zero, one, or more occurrences in the deduction Σ_1 of the formula A as an assumption discharged by the deduction of $A \rightarrow B$ above. The notation

$$\begin{array}{c}\Sigma_2\\ [A]\\ \Sigma_1\end{array}$$

means that each of the aforementioned occurrences of A are replaced by the deduction

$$\frac{\Sigma_2}{A} \ .$$

This concludes our quick warmup of the two main topics of this thesis. We are now ready for the actual dance.

Chapter 2

A simpler version of Bergstra-Klop's Theorem

A redex Δ is *perpetual* [BK82] if contraction of Δ to Δ' in any context C preserves infinite reductions, *ie*, for all contexts C

$$C[\Delta] \to_\beta C[\Delta'] \ \ \& \ \ C[\Delta] \in \infty_\beta \Longrightarrow C[\Delta'] \in \infty_\beta \ . \tag{6}$$

Examples of perpetual redexes are $\Delta \equiv I\,I$ and $\Delta \equiv \mathbf{K}\,\Omega\,I$, while $\Delta \equiv \mathbf{K}\,I\,\Omega$ is not perpetual. A perpetual redex is thus a redex where its contraction does not change the infinity of any term where the redex occur.

Note that if all redexes in a term are perpetual then the term has the conservation property (1):

$$M \to_\beta N \ \& \ M \in \infty_\beta \Longrightarrow N \in \infty_\beta \ .$$

The converse does not hold. For instance the term $x\,((\lambda y.z)\,\Omega)\,\Omega$ has the conservation property, but the redex $(\lambda y.z)\,\Omega$ is not perpetual. (To see this, note that with $C \equiv []$, we have $C[(\lambda y.z)\,\Omega] \in \infty_\beta$, but $C[z] \notin \infty_\beta$.) The reason is that for a term to satisfy the conservation property, it is required that contraction of any redex in the term preserves infinite reductions *from this term only*. In contrast, for a redex to be perpetual it must preserve infinite reductions when contracted inside *an arbitrary context*. Conditions for perpetuality of redexes can hence be a stepping stone for proving the conservation property of terms, which in turn can be used to characterise uniformly normalising terms.

Bergstra and Klop [BK82] characterised perpetual redexes in Λ^K. They proved that a perpetual redex in Λ^K is either an I-redex or a $\mathbf{K}$-redex $\mathbf{K}\,A\,B$ where A, in a certain sense, is at least as infinite as B. The proof was later simplified in [vRSSX99]. The drawback of Bergstra and Klop's characterisation of perpetual redexes is that the proof is quite complicated as they give a necessary and sufficient condition.

When proving conservation, we only need a sufficient condition for perpetuality of redexes: if we can prove that all redexes in a term are perpetual, the conservation property holds for the term. Similarly, sufficient conditions for perpetuality of redexes are useful for proving uniform normalisation of terms, as we shall see. In this chapter, we formulate and prove a sufficient condition for perpetuality of redexes in Λ^K. As the condition is not necessary, this gives a weaker version of Berstra-Klop's theorem. However, the condition is useful in the following section where we present a new characterisation of uniformly normalising terms in Λ^K containing terms not in Λ^I.

The proof that our condition implies perpetuality of a redex is simpler than the corresponding proof in [BK82]. Our condition is therefore easier to extend to other λ-calculi than Λ^K.

2.1 The Proof for the Simplification

As in the proof by Bergstra and Klop, and in the simplification [vRSSX99], we introduce the set of *underlined* or *labeled* λ*-terms* taken from [Bar84]. In this set some of the abstractions are labeled with an underline, *ie*, $\underline{\lambda}x.P$, and can hence be distinguished from the other abstractions. This allows us to keep track of an abstraction in a reduction path. This notion is a restriction of the general setting found in [Bar84, 11.1].

Definition 2.1

i. The set $\underline{\Lambda}^K$ ($\underline{\lambda}$-terms or labeled λ-terms) is defined as follows

$$\begin{array}{lcl} x \in \underline{\Lambda}^K & & \\ P \in \underline{\Lambda}^K & \Longrightarrow & \lambda x.P \in \underline{\Lambda}^K \\ P, Q \in \underline{\Lambda}^K & \Longrightarrow & P\,Q \in \underline{\Lambda}^K \\ P, Q \in \underline{\Lambda}^K & \Longrightarrow & (\underline{\lambda}x.P)\,Q \in \underline{\Lambda}^K \end{array}$$

In the last clause $(\underline{\lambda}x.P)\,Q$ is a *labeled redex*. A labeled or unlabeled abstraction is denoted λ^*, *ie*, $\lambda^* ::= \lambda \mid \underline{\lambda}$.

ii. The notions of reductions $\underline{\beta}$ and β on $\underline{\Lambda}^K$ are defined by

$$\begin{array}{lcl} (\underline{\lambda}x.P)\,Q & \underline{\beta} & P\{x := Q\} \\ (\lambda x.P)\,Q & \beta & P\{x := Q\} \end{array}$$

iii. The notion of reduction β^**-reduction* is defined by $\beta^* = \underline{\beta} \cup \beta$. ♦

It is important to notice that only the operators of a redex can be labeled.

Moving from a labeled term in $\underline{\Lambda}^K$ to an unlabeled term in Λ^K can be done by either *erasing* or *reducing* all labeled redexes. We define these two operations by the following functions $|\bullet|$ and φ, taken from [Bar84].

Definition 2.2 (Erasing of labels) For all $\underline{\Lambda}^K$-terms M we define the image $|M| \in \Lambda^K$ as follows.

$$\begin{array}{lcl} |x| & = & x \\ |\lambda x.P| & = & \lambda x.|P| \\ |P\,Q| & = & |P|\,|Q| \\ |(\underline{\lambda}x.P)\,Q| & = & (\lambda x.|P|)\,|Q| \end{array}$$

♦

Definition 2.3 (Reduction of labeled redexes) For all $\underline{\Lambda}^K$-terms M we define the image $\varphi(M) \in \Lambda^K$ as follows.

$$\begin{array}{lcl} \varphi(x) & = & x \\ \varphi(\lambda x.P) & = & \lambda x.\varphi(P) \\ \varphi(P\,Q) & = & \varphi(P)\,\varphi(Q) \\ \varphi((\underline{\lambda}x.P)\,Q) & = & \varphi(P)\{x := \varphi(Q)\} \end{array}$$

♦

The label reduction function erases all labeled redexes and finds the $\underline{\beta}$-normal form as new labeled redex cannot be created by $\underline{\beta}$-reduction.

The following fundamental lemma is also taken from [Bar84].

Lemma 2.4 ([Bar84, 11.1.7(i)]) *For all $M, N \in \underline{\Lambda}^K$ we have:*

$$\varphi(M\{x := N\}) = \varphi(M)\{x := \varphi(N)\}$$

The contractum of a $\mathbf{K}$-redex $\mathbf{K}\ A\ B$ is A—the second argument B is thrown away. For (6) to hold for a $\mathbf{K}$-redex, we should therefore ensure that we do not erase the only infinite reduction path from M by throwing away B. We cannot make any assumptions about the context C. We therefore have to require that if B is an infinite term, then A is also an infinite term.

Furthermore, we have to consider what might be substituted in the arguments A and B during the course of a reduction inside some context. Hence it is not sufficient to ensure that if B is infinite, then A is also infinite. We should look at all possible substitutions θ that can be performed on A and B. This motivates the following definition.

Definition 2.5 Let $(\Lambda, \mathbf{R}) = (\Lambda^K, \beta)$ or $(\Lambda, \mathbf{R}) = (\underline{\Lambda}^K, \beta^*)$. For all terms $P, Q \in \Lambda$, the term P is *at least as infinite* as Q, written $P \succeq_{\infty,\mathbf{R}} Q$, if, and only if, we for all substitutions θ have

$$Q\theta \in \infty_{\mathbf{R}} \Longrightarrow P\theta \in \infty_{\mathbf{R}}\ .$$

♦

We will omit $\mathbf{R}$ when it is clear from the context. A (labeled or unlabeled) $\mathbf{K}$-redex $\mathbf{K}\ A\ B$ or $\underline{\mathbf{K}}\ A\ B$ is *good*, if, and only if, $A \succeq_{\infty,\beta} B$. We will refer to the condition $P\theta \in \infty_{\mathbf{R}}$ as "P is *infinite under the substitution* θ".

In this section we prove the following.

Theorem 2.6 (Conservation of K-redexes) *The redex $\mathbf{K}\ A\ B$ is perpetual, if $\mathbf{K}\ A\ B$ is good.*

A slightly stronger theorem is proved by Bergstra and Klop [BK82]: in their definition of $\succeq_{\infty,\mathbf{R}}$ (let us call their variant $\succeq^{\text{SN}}_{\infty,\mathbf{R}}$) one does not consider all substitutions θ, but only all SN-*substitutions* (substitutions that substitute only strongly normalising terms). They then prove that $\mathbf{K}\ A\ B$ is perpetual, if, and only if, $A \succeq^{\text{SN}}_{\infty,\beta} B$.

The difference is non-trivial, as illustrated by

$$\mathbf{K}\ A\ B \equiv \mathbf{K}\ x\ y.$$

For all SN-substitution θ it holds that $A\theta \in \infty_\beta \Leftarrow B\theta \in \infty_\beta$, since y will only be replaced by SN-terms and thus $B\theta \notin \infty_\beta$ in all cases. On the other hand $A \not\succeq_\infty B$ by our definition, as illustrated by $\theta = \{y := \Omega\}$. However, for all $\mathbf{K}$-redexes $\mathbf{K}\ A\ B$ where $\text{FV}(A) = \text{FV}(B)$, our condition is equivalent to the one by Bergstra and Klop.

A drawback of using SN-substitutions is that they are not closed under composition: Consider for instance the substitutions $\theta = \{x := u\ u\}$ and $\theta' = \{u := \omega\}$. Clearly θ and θ' are SN-substitutions, but $\theta\,\theta' = \{u := \omega, x := \omega\,\omega\}$ is not a SN-substitution as $\Omega = \omega\,\omega \in \infty_\beta$. This complicates the proof of Theorem 2.6 in [BK82] as one has to ensure that two substitutions are never applied on the same subterm.

Our weaker definition avoids this problem and we get a simpler proof at the cost of a slightly weaker theorem. The proof follows the structure of the proof in [vRSSX99, sec. 7.4]. The reader is encouraged to compare our proof to this latter proof to see that our proof is indeed simpler.

The following proposition by van Raamsdonk et al. [vRSSX99, prop. 7.7] is pivotal for the proof.

Proposition 2.7 *Let $M \in \Lambda^K$ and $M \to_\beta N$. Then*

$$M \in \infty_\beta \Longrightarrow N \in \infty_\beta$$

if there is a subet S of $\underline{\Lambda}^K$ and a reduction strategy $F^* : \infty_{\beta^*} \to \infty_{\beta^*}$ with

i. $M \equiv C[(\lambda x.P)\,Q]$, and $N \equiv C[P\{x := Q\}]$, and $C[(\underline{\lambda} x.P)\,Q] \in S$ for some C, P and Q.

ii. $L \in S$ implies $F^*(L) \in S$

iii. For all $L \in S$ where $L \to_\beta F^*(L)$ we have $\varphi(L) \twoheadrightarrow_\beta^+ \varphi(F^*(L))$

Notice that the last condition only considers β-redexes not $\underline{\beta}$-redexes.

Using this proposition we can prove that a redex Δ of M is perpetual by looking at a corresponding labeled redex in a subset S of $\underline{\Lambda}^K$: The conditions in the proposition state that the subset S should be accompanied by a perpetual reduction strategy F^*. We should guaranty that a β-reduction by F^* is reduced to at least one β-reduction in Λ^K. When the conditions are met, we can construct an infinite reduction path from the $\underline{\Lambda}^K$-term where the redex Δ is labeled. This reduction path only contains finitely many $\underline{\beta}$-reductions (this is not used directly below, but can be found in [Bar84, chp. 11]). When using the reduction in Definition 2.3 of $\underline{\Lambda}^K$ to Λ^K, we then get an infinite reduction path from the reduced term N as exemplified by the following diagram:

$$\begin{array}{lllll} M_0 & & M_1 & & M_2 & & \cdots \\ \to_\beta & L & & F^*(L) & & (F^*)^2(L) & & \cdots \\ \twoheadrightarrow_{\underline{\beta}} & N_0 & & N_1 & & N_2 & & \cdots \end{array}$$

The major steps in the proof of Theorem 2.6 are to find a subset S and a perpetual reduction strategy F^* satisfying the conditions of Proposition 2.7.

We first define our subset.

Definition 2.8 The subset $\underline{\Lambda}^{\underline{K}}$ is the largest subset of $\underline{\Lambda}^K$ such that $(\underline{\lambda} x.P)\,Q \subseteq M$ implies $x \notin \mathrm{FV}(P)$ for all $M \in \underline{\Lambda}^{\underline{K}}$. ♦

Note that the only visual difference between $\underline{\Lambda}^K$ and $\underline{\Lambda}^{\underline{K}}$ is the label on K indicating that all labeled redexes are $\mathbf{K}$-redexes.

Lemma 2.9 *For all $M, N \in \underline{\Lambda}^K$ where $M \in \underline{\Lambda}^{\underline{K}}$ and $N \subseteq M$ we also have $N \in \underline{\Lambda}^{\underline{K}}$.*

Proof. Consider any $(\underline{\lambda} x.P)\,Q \subseteq N$. From the transitivity of $\subseteq$, we have $(\underline{\lambda} x.P)\,Q \subseteq M$ and hence $x \notin \mathrm{FV}(P)$. □

Definition 2.10 $M \in \underline{\Lambda}^K$ is *good*, if, and only if, for all $\underline{\mathbf{K}}\,A\,B \subseteq M$, it holds that $\underline{\mathbf{K}}\,A\,B$ is good. ♦

Lemma 2.11 *For all $M, N \in \underline{\Lambda}^K$ where M is good and $N \subseteq M$, the term N is also good.*

Proof. Consider any $\underline{\mathbf{K}}\,A\,B \subseteq N$. From the transitivity of $\subseteq$, we have $\underline{\mathbf{K}}\,A\,B \subseteq M$ and hence $\underline{\mathbf{K}}\,A\,B$ is good. □

Our subset of interest is $\underline{\Lambda}^K_{\text{good}} = \{M \in \underline{\Lambda}^K \mid M \text{ is good}\}$. We now turn to our perpetual strategy.

Definition 2.12 We define $F^* : \infty_{\beta^*} \to \underline{\Lambda}^K$ by:

$$\begin{array}{lcl}
F^*(x\,P_1\,\ldots\,P_m\,\ldots\,P_n) & = & x\,P_1\,\ldots\,F^*(P_m)\,\ldots\,P_n \\
 & & \quad \text{if } P_m \in \infty_{\beta^*},\, P_i \in \text{SN}_{\beta^*} \text{ for } i < m \\
F^*(\lambda x.P) & = & \lambda x.F^*(P_m) \\
F^*((\lambda^* x.P_0)\,P_1\,\ldots P_n) & = & P_0\{x := P_1\}\,P_2\,\ldots\,P_n \\
 & & \quad \text{if } P_0, P_1 \in \text{SN}_{\beta^*} \\
F^*((\lambda^* x.P_0)\,P_1\,\ldots\,P_n) & = & (\lambda^* x.P_0)\,F^*(P_1)\,\ldots\,P_n \\
 & & \quad \text{if } P_0 \in \text{SN}_{\beta^*}, P_1 \in \infty_{\beta^*} \\
F^*((\lambda^* x.P_0)\,P_1\,\ldots\,P_n) & = & (\lambda^* x.F^*(P_0))\,P_1\,P_2\,\ldots\,P_n \\
 & & \quad \text{if } P_0 \in \infty_{\beta^*}
\end{array}$$

♦

The following three lemmata establish that the set $\underline{\Lambda}^K_{\text{good}} \cap \infty_{\beta^*}$ is indeed closed under F^*. The first two are proved in [vRSSX99] and they state that F^* is perpetual and $\infty_{\beta^*} \cap \underline{\Lambda}^K$ is closed under F^*.

Lemma 2.13 ([vRSSX99, lem. 7.24]) *Let $M \in \infty_{\beta^*}$. Then $F^*(M) \in \infty_{\beta^*}$.*

Lemma 2.14 ([vRSSX99, lem. 7.25]) *Let $M \in \underline{\Lambda}^K \cap \infty_{\beta^*}$. Then $F^*(M) \in \underline{\Lambda}^K$.*

Lemma 2.15 *Let $M \in \underline{\Lambda}^K_{\text{good}} \cap \infty_{\beta^*}$. Then $F^*(M)$ is good.*

Proof. Let $\underline{\mathbf{K}}\,A\,B \subseteq F^*(M)$ be a labeled $\mathbf{K}$-redex and θ any substitution. We will show that $B\theta \in \infty_{\beta^*}$ implies $A\theta \in \infty_{\beta^*}$. We use induction on the structure of M exploiting the fact that as M is infinite there is at least one infinite subterm of M.

i. $M \equiv x\,P_1\,\ldots\,P_m\,\ldots\,P_n$ with $P_m \in \infty_{\beta^*}$ and $P_i \in \text{SN}_{\beta^*}$ for $i < m$. Using Definition 2.12, we have $F^*(M) \equiv x\,P_1\,\ldots\,F^*(P_m)\,\ldots\,P_n$. We divide into two subcases:

   (a) $\underline{\mathbf{K}}\,A\,B \subseteq P_i$ for some $i \neq m$. We then have $\underline{\mathbf{K}}\,A\,B \subseteq P_i \subseteq M$ and $A \succeq_{\infty,\beta^*} B$ as M is good.

   (b) $\underline{\mathbf{K}}\,A\,B \subseteq F^*(P_m)$. As $P_m \in \infty_{\beta^*}$ and $P_m \in \underline{\Lambda}^K_{\text{good}}$ by lemmata 2.9 and 2.11, we conclude, using the induction hypothesis, that $F^*(P_m)$ is good. We then have $A \succeq_{\infty,\beta^*} B$ from Definition 2.10.

ii. $M \equiv \lambda x.P$. We have $F^*(M) \equiv \lambda x.F^*(P)$. We can only have $\underline{\mathbf{K}}\,A\,B \subseteq \lambda x.F^*(P)$, if $\underline{\mathbf{K}}\,A\,B \subseteq F^*(P)$. Furthermore, as $(\lambda x.P) \in \infty_{\beta^*}$, we must necessarily have $P \in \infty_{\beta^*}$. Therefore, we have $A \succeq_{\infty,\beta^*} B$ by a reasoning similar to Case i(b).

iii. $M \equiv (\lambda^* x.P_0)\,P_1\,\ldots\,P_n$. We split into cases depending on whether $P_0, P_1 \in \text{SN}_{\beta^*}$.

   (a) $P_0, P_1 \in \text{SN}_{\beta^*}$. Then $F^*(M) \equiv P_0\{x := P_1\}\,P_2\,\ldots\,P_n$. We consider two cases

      i. $\underline{\mathbf{K}}\,A\,B \subseteq P_i$ where $i = 2, \ldots, n$. This is like Case i(a) above.

      ii. Otherwise, we have $\underline{\mathbf{K}}\,A\,B \subseteq P_0\{x := P_1\}$. By the substitution generation lemma we have a subterm $\underline{\mathbf{K}}\,A'\,B' \subseteq P_0$ such that

$$\underline{\mathbf{K}}\,A\,B \equiv (\underline{\mathbf{K}}\,A'\,B')\{x := P_1\} \equiv \underline{\mathbf{K}}\,(A'\{x := P_1\})\,(B'\{x := P_1\})\ .$$

      As $\underline{\mathbf{K}}\,A'\,B' \subseteq P_0 \subseteq M$ we have $A'\phi \in \infty_{\beta^*} \Leftarrow B'\phi \in \infty_{\beta^*}$ for all substitutions ϕ. In particular, for the substitution $\phi = \{x := P_1\}\theta$ we have

$$B\theta = B'\phi \in \infty_{\beta^*} \Longrightarrow A'\phi \in \infty_{\beta^*} = A\theta\ .$$

(b) $P_0 \in \mathrm{SN}_{\beta^*}$ and $P_1 \in \infty_{\beta^*}$. We cannot have $M \equiv (\underline{\lambda}x.P_0)\, P_1\, \ldots\, P_n \in \underline{\Lambda}^K_{\mathrm{good}}$. By definition 2.8, we would then have $(\underline{\lambda}x.P_0)\, P_1 \equiv \underline{\mathbf{K}}\, P_0\, P_1$. Using the empty substitution $\emptyset$ this would lead to $P_0\emptyset \in \mathrm{SN}_{\beta^*}$ and $P_1\emptyset \in \infty_{\beta^*}$, which contradicts $P_0 \succeq_{\infty,\beta^*} P_1$ as required by Definition 2.10. Therefore, we have $M \equiv \lambda x.P_0\, P_1\, \ldots\, P_n$ and $F^*(M) = (\lambda x.P_0)\, F^*(P_1)\, \ldots\, P_n$. This gives us two subcases:

i. $\underline{\mathbf{K}}\, A\, B \subseteq P_i$ for $i \neq 1$. This is similar to Case i(a).

ii. $\underline{\mathbf{K}}\, A\, B \subseteq F^*(P_1)$. The case is like Case i(b).

(c) $P_0 \in \infty_{\beta^*}$. Then $F^*(M) = (\lambda^* x.F^*(P_0))\, P_1\, \ldots\, P_n$. We consider three subcases

i. $\underline{\mathbf{K}}\, A\, B \subseteq F^*(P_0)$. As $P_0 \in \infty_{\beta^*}$, we can use the same reasoning as in Case i(b) to conclude that $A \succeq_{\infty,\beta^*} B$.

ii. $\underline{\mathbf{K}}\, A\, B \subseteq P_i$. This is similar to Case i(a).

iii. $\underline{\mathbf{K}}\, A\, B \equiv (\lambda^* x.F^*(P_0))\, P_1$. By Lemma 2.13, we have $F^*(P_0) \in \infty_{\beta^*}$. It trivially follows that $P_1\theta \in \infty_{\beta^*} \Rightarrow F^*(P_0)\, \theta \in \infty_{\beta^*}$.

This exhausts the possibilities and concludes the proof that goodness is preserved under F^*. □

Proposition 2.16 *Let M be a term in $\underline{\Lambda}^K_{\mathrm{good}} \cap \infty_{\beta^*}$, then $F^*(M)$ is also in $\underline{\Lambda}^K_{\mathrm{good}} \cap \infty_{\beta^*}$.*

Proof. By the three previous lemmata. □

We have now established that the subset $\underline{\Lambda}^K_{\mathrm{good}}$ of $\underline{\Lambda}^K$ with our reduction strategy F^* satisfies Condition ii of Proposition 2.7. The following proposition establishes that Condition iii is also satisfied.

Proposition 2.17 *Let M be a term in $\underline{\Lambda}^K_{\mathrm{good}} \cap \infty_{\beta^*}$. We then have*

$$M \to_\beta F^*(M) \implies \varphi(M) \twoheadrightarrow^+_\beta \varphi(F^*(M)) \ .$$

Proof. We assume that

$$M \to_\beta F^*(M) \ . \tag{$*$}$$

and use induction over the structure of M:

i. $M \equiv x\, P_1\, \ldots\, P_m\, \ldots\, P_n$ with $P_m \in \infty_{\beta^*}$, $P_i \in \mathrm{SN}_{\beta^*}$ for $i < m$. We then have

$$F^*(M) = x\, P_1\, \ldots\, F^*(P_m)\, \ldots\, P_n \ ,$$
$$\varphi(M) = x\, \varphi(P_1)\, \ldots\, \varphi(P_m)\, \ldots\, \varphi(P_n) \ ,$$

and

$$\varphi(F^*(M)) = x\, \varphi(P_1)\, \ldots\, \varphi(F^*(P_m))\, \ldots\, \varphi(P_n) \ .$$

We can only have $M \to F^*(M)$ by $P_m \to_\beta F^*(P_m)$. It follows from the induction hypothesis that $\varphi(P_m) \twoheadrightarrow^+_\beta \varphi(F^*(P_m))$ and then also that $\varphi(M) \twoheadrightarrow^+_\beta \varphi(F^*(M))$ by the compatibility.

ii. $M \equiv \lambda x.P$. This case is similar to Case i.

iii. $M \equiv (\lambda x.P_0)\ P_1\ \ldots\ P_n$. We have

$$\varphi(M) = (\lambda x.\varphi(P_0))\ \varphi(P_1)\ \varphi(P_2)\ \ldots\ \varphi(P_n)\ .$$

We split into three cases depending on whether $P_0, P_1 \in \mathrm{SN}_{\beta^*}$.

(a) $P_0, P_1 \in \mathrm{SN}_{\beta^*}$. We have

$$F^*(M) = P_0\{x := P_1\}\ P_2\ \ldots\ P_n$$

and

$$\varphi(F^*(M)) = \varphi(P_0\{x := P_1\})\ \varphi(P_2)\ \ldots\ \varphi(P_n)\ .$$

We can do the following reduction

$$\begin{aligned}\varphi(M) &= (\lambda x.\varphi(P_0))\ \varphi(P_1)\ \varphi(P_2)\ \ldots\ \varphi(P_n)\\ &\to_\beta \varphi(P_0)\{x := \varphi(P_1)\}\ \varphi(P_2)\ \ldots\ \varphi(P_n)\\ &\overset{2.4}{=} \varphi(P_0\{x := P_1\})\ \varphi(P_2)\ \ldots\ \varphi(P_n)\\ &= \varphi(F^*(M))\end{aligned}$$

(b) $P_0 \in \mathrm{SN}_{\beta^*}$ and $P_1 \in \infty_{\beta^*}$. We have

$$F^*(M) = (\lambda x.P_0)\ F^*(P_1)\ \ldots\ P_n$$

and

$$\varphi(F^*(M)) = (\lambda x.\varphi(P_0))\ \varphi(F^*(P_1))\ \ldots\ \varphi(P_n)\ .$$

It is easily seen that $M \to_\beta F^*(M)$, if, and only if, $P_1 \to_\beta F^*(P_1)$ so the case is similar to Case i.

(c) $P_0 \in \infty_{\beta^*}$. We then have

$$F^*(M) = (\lambda x.F^*(P_0))\ P_1\ \ldots\ P_n$$

and

$$\varphi(F^*(M)) = (\lambda x.\varphi(F^*(P_0)))\ \varphi(P_1)\ \ldots\ \varphi(P_n)\ .$$

It is clear that $M \to_\beta F^*(M)$, if, and only if, $P_0 \to_\beta F^*(P_0)$. This makes the case similar to Case i.

iv. $M \equiv (\underline{\lambda} x.P_0)\ P_1\ P_2\ \ldots\ P_n$ where $n \geq 1$. We have

$$\varphi(M) = \varphi(P_0)\{x := \varphi(P_1)\}\ \varphi(P_2)\ \ldots\ \varphi(P_n)\ .$$

We divide into three subcases

(a) $P_0, P_1 \in \mathrm{SN}_{\beta^*}$. In this case, we have

$$F^*(M) = P_0\{x := P_1\}\ P_2\ \ldots\ P_n\ .$$

As we have done a $\underline{\beta}$-reduction, there is no way to satisfy $(*)$. This makes the proposition trivially true.

(b) $P_0 \in \mathrm{SN}_{\beta^*}$ and $P_1 \in \infty_{\beta^*}$. This case is impossible by a reasoning similar to the one used in Case iii(b) in the proof of Lemma 2.15.

(c) $P_0 \in \infty_{\beta^*}$. We have

$$F^*(M) = (\underline{\lambda}x.F^*(P_0))\ P_1\ \ldots\ P_n$$

and

$$\varphi(F^*(M)) = \varphi(F^*(P_0))\{x := \varphi(P_1)\}\ \ldots\ \varphi(P_n)\ .$$

We clearly have $M \to_\beta F^*(M)$, if, and only if, $P_0 \to_\beta F^*(P_0)$, which makes the case similar to i.

This exhausts all the possibilities and we conclude that

$$M \to_\beta F^*(M) \Longrightarrow \varphi(M) \twoheadrightarrow_\beta^+ \varphi(F^*(M))\ .$$

□

Proof (of Theorem 2.6). Let there be given a redex $\mathbf{K}\ A\ B$ with $A \succeq_{\infty,\beta} B$. We must prove that for any context C, if $C[\mathbf{K}\ A\ B] \in \infty_\beta$ then also $C[A] \in \infty_\beta$. Let a context C with $C[\mathbf{K}\ A\ B] \in \infty_\beta$ be given. We use Proposition 2.7 with $S = \underline{\Lambda}^K_{\text{good}} \cap \infty_{\beta^*}$. Condition i is satisfied as $C[\underline{\mathbf{K}}\ A\ B] \in \underline{\Lambda}^K_{\text{good}}$. The conditions ii and iii are satisfied by Propositions 2.16 and 2.17, respectively. □

2.2 Conclusion

Besides the value of having a simpler version of the Bergstra-Klop Theorem, this chapter has illustrated a lot of the techniques we will use in the following. We have introduced ways to compare infinity of two terms. We have presented the notion of goodness to ensure conservation of infinite reduction paths under reduction. We have used a reduction strategy to support our reasoning about infinite reduction paths. All these notions will reappear in several forms in the rest of this thesis. Furthermore, we will use Theorem 2.6 in the next chapter to find a uniformly normalising subset of Λ^K.

Chapter 3

A Uniformly Normalising Subset of Λ^K

We recall that a term M is uniformly normalising if $M \in \text{WN}_\beta$ implies $M \in \text{SN}_\beta$, and that a set of terms is uniformly normalising if all its terms are uniformly normalising. The classical example of a uniformly normalising subset of Λ^K is Λ^I. The two important features of Λ^I are that

i. subterms are not deleted under β-reduction. In particular, infinite subterms are not deleted under β-reduction.

ii. the subset is closed under β-reduction. Thus a reduction path will never lead to a term, where infinite subterms can be deleted under β-reduction.

In contrast, erasing reductions delete subterms. As discussed in the introduction, we can still find subsets where all terms have the *conservation property* (1). In these subsets it does not matter if we delete an infinite subterm, as infinity is conserved. However, such subsets are not necessarily closed under reduction.

One such example is the subset of Λ^K of all terms having the conservation property. The subset is not closed under β-reduction as illustrated by

$$(\lambda y.\mathbf{K}\, x\, y)\, \Omega \quad \to_\beta \quad \mathbf{K}\, x\, \Omega \ .$$

In the left term, all redexes are perpetual: $\lambda y.\mathbf{K}\, x\, y$ and Ω are I-redexes and there is no SN-substitution making the y of $\mathbf{K}\, x\, y$ an infinite term. Hence, the term has the conservation property. However, the right term does not have the conservation property due to the redex $\mathbf{K}\, x\, \Omega$: we have $\mathbf{K}\, x\, \Omega \in \infty_\beta$, but $x \notin \infty_\beta$.

To find a uniformly normalising subset of a λ-calculus, the challenge is thus to find a subset that is both closed under reduction and has the conservation property. In this chapter, we will do this for Λ^K. This is also the first demonstration of our technique to find uniformly normalising subsets of a λ-calculus.

3.1 Proof of Uniform Normalisation

In proving uniform normalisation we will use two simple conditions: the term must be *good* and $\mathbf{K}$*-expanded*—both are defined formally below. In a good term all redexes are perpetual, so the term has the conservation property. Concerning closure under reduction, it is a problem that $\mathbf{K}$-redexes can be created in a reduction, as illustrate by

$$((\lambda x.\mathbf{K}\, x)\, \lambda y.y)\, \Omega \to_\beta \mathbf{K}\, (\lambda y.y)\, \Omega \ .$$

In the left term, the only redex is an I-redex and hence all redexes are perpetual, but in the right term the newly created $\mathbf{K}$-redex is not perpetual. This is avoided by requiring that all $\mathbf{K}$-abstractions have an argument—we say that the term is $\mathbf{K}$-expanded. In the left term above the $\mathbf{K}$-abstraction $\mathbf{K}\,x$ does not have an argument. Hence the left term is not $\mathbf{K}$-expanded.

For this chapter we will use the following definition of good.

Definition 3.1 A term is *good*, if, and only if, all $\mathbf{K}$-redexes are good. ⧫

The difference from the previous section is that we now consider all $\mathbf{K}$-redexes instead of only labeled $\mathbf{K}$-redexes.

The reader should be aware that there is a slight risk of confusion as a redex $(\lambda x.P)\,Q$ is also a term. However, it should be clear from the context whether we consider a redex as a term or as a redex.

Lemma 3.2 *Let $M, N \in \Lambda$ be terms with $N \subseteq M$. If M is good, then N is good.*

Proof. Consider any $\mathbf{K}$-redex $\mathbf{K}\,P\,Q \subseteq N$. We have $\mathbf{K}\,P\,Q \subseteq M$ using the transitivity of $\subseteq$. Hence, the redex $kredPQ$ is good. As all $\mathbf{K}$-redexes in N are good, N is good. □

Proposition 3.3 *Let M, N be terms where $M \to_\beta N$ and M is good. If $M \in \infty_\beta$ then $N \in \infty_\beta$.*

Proof. For some context C we have $M \equiv C[(\lambda x.P)\,Q] \to_\beta C[P\{x := Q\}] \equiv N$. If $(\lambda x.P)\,Q$ is a I-redex, then the proposition follows from [vRSSX99, 7.19]. If on the other hand $(\lambda x.P)\,Q$ is a $\mathbf{K}$-redex, the proposition follows from Theorem 2.6. □

Thus, a good term has the conservation property. We will now concentrate on finding a subset of the good terms that is closed under reduction. Essentially this is done by requiring that all $\mathbf{K}$-abstractions have an argument. However, there might be abstractions in a term that are never a subterm of a redex in any of the reduction paths from the term. Consider for instance $\lambda x.\ldots$ and $\lambda y.\ldots$ in the term

$$\lambda x.x\ (\lambda y.\mathbf{K}\ (\lambda u.u)\ (\lambda v.v))\ ((\lambda u.u)\ (\lambda v.v))\ .$$

These abstractions cannot become a part of newly created redex. By precluding these abstractions from the set of $\mathbf{K}$-abstractions that we require to have an argument, there are more terms fulfilling the conditions. We thus get a bigger subset that is uniformly normalising.

We therefore add the syntactic criterion of only considering *relevant abstractions*. The relevant abstractions are abstractions that might occur as the operator of a redex in a reduction path of M. Note that the following definition—naturally—includes too many abstractions, *eg*, the second abstraction in $(\lambda x.x)(\lambda x.x)$. However, to define the set exactly one would have to consider all possible reduction paths: a semantic criterion.

Definition 3.4 The set of relevant abstractions of a term M is defined as follows.

$$\begin{array}{lcll} \mathrm{rel}(x\ P_1\ \ldots\ P_n) & = & \bigcup_{i=1}^{n} \mathrm{rel}(P_i) & n \geq 0 \\ \mathrm{rel}(\lambda x.P) & = & \mathrm{rel}(P) & \\ \mathrm{rel}((\lambda x.P_0)\ P_1\ P_2\ \ldots\ P_n) & = & \{\lambda y.Q \subseteq P_i \mid 0 \leq i \leq n\} & n \geq 1. \end{array}$$

⧫

Definition 3.5 $\mathbf{K}$*-expanded*

i. A $\mathbf{K}$-abstraction is $\mathbf{K}$*-expanded* in a term, if, and only if, the abstraction is the operator of a redex in the term.

ii. A term is $\mathbf{K}$-expanded, if, and only if, all its relevant $\mathbf{K}$-abstractions are $\mathbf{K}$-expanded. ⧫

Thus the term $\mathbf{K}\ (\mathbf{K}\ (\lambda x.x)\ (\lambda y.y))$ is $\mathbf{K}$-expanded but $\mathbf{K}\ (\mathbf{K}\ (\lambda x.x))\ (\lambda y.y)$ is not.

Lemma 3.6 *Let $M, N \in \Lambda^K$ be terms with $N \subseteq M$. Then* $\mathrm{rel}(N) \subseteq \mathrm{rel}(M)$.

Corollary 3.7 *Let $M, N \in \Lambda^K$ be terms with $N \subseteq M$. If M is $\mathbf{K}$-expanded, then N is $\mathbf{K}$-expanded.*

Proof (Lemma 3.6). The lemma trivially holds when $N \equiv M$. Hence, we assume that $N \subset M$ and prove that $\mathrm{rel}(N) \subseteq \mathrm{rel}(M)$. It then follows that N is $\mathbf{K}$-expanded as all relevant abstractions of N are $\mathbf{K}$-expanded. We use induction on the structure of M:

i. $M \equiv x\ P_1\ \ldots\ P_n$ where $n \geq 0$. Then $\mathrm{rel}(M) = \bigcup_{i=1}^{n} \mathrm{rel}(P_i)$. We split into two cases:

   (a) $N \equiv P_i$ for some i where $1 \leq i \leq n$. We clearly have $\mathrm{rel}(N) \subseteq \mathrm{rel}(M)$.

   (b) $N \equiv x\ P_1\ \ldots\ P_m$ for some m where $0 \leq m < n$. In this case we have $\mathrm{rel}(N) = \bigcup_{i=1}^{m} \mathrm{rel}(P_i) \subseteq \mathrm{rel}(M)$.

ii. $M \equiv \lambda x.P$. We have $N \subseteq P$ and it follows, by the induction hypothesis, that $\mathrm{rel}(N) \subseteq \mathrm{rel}(P) = \mathrm{rel}(M)$.

iii. $M \equiv (\lambda x.P_0)\ P_1\ P_2\ \ldots\ P_n$ for some n where $n \geq 1$. It follows that we have $\mathrm{rel}(M) = \{\lambda z.S \subseteq P_i \mid 0 \leq i \leq n\}$. We split into the following cases:

   (a) $N \subseteq \lambda x.P_0$ or $N \subseteq P_i$ for some i where $0 \leq i \leq m$. In the former case, we have $\mathrm{rel}(\lambda x.P_0) = \mathrm{rel}(P_0)$, so in all cases we have, by the induction hypothesis, $\mathrm{rel}(N) \subseteq \mathrm{rel}(P_i) \subseteq \mathrm{rel}(M)$ for some i where $0 \leq i \leq m$.

   (b) $N \equiv (\lambda x.P_0)\ P_1\ P_2\ \ldots\ P_m$ for some m where $1 \leq m < n$. Therefore, we have $\mathrm{rel}(N) = \{\lambda z.S \subseteq P_i \mid 0 \leq i \leq m\} \subseteq \mathrm{rel}(M)$.

This exhausts the possibilities and concludes the proof that the relevant abstractions of a subterm are part of the relevant abstractions of a term. □

The main theorem we shall prove is

Theorem 3.8 *Let $M \in \Lambda$ be a good and $\mathbf{K}$-expanded term. Then $M \in \mathrm{UN}_\beta$.*

It is important to notice that even in combination with the $\mathbf{K}$-expanded criterion, the Bergstra-Klop criterion, which only considers SN-substitutions in the definition of good, does not give a subset that is closed under reduction. Consider again the reduction

$$(\lambda y.\mathbf{K}\ x\ y)\ \Omega \quad \to_\beta \quad \mathbf{K}\ x\ \Omega\ .$$

The left term is $\mathbf{K}$-expanded and it would have been good if we only considered SN-substitutions. The right term is $\mathbf{K}$-expanded, but not good, even if we only considered SN-substitutions. As we would expect, neither of the terms are good by the definition above where all possible substitutions are considered. This illustrates that considering

all possible substitutions seems crucial in finding a uniformly normalising subset of Λ^K under erasing β-reductions.

The idea of the proof of Theorem 3.8 is captured by the following diagram:

$$\begin{array}{ll} \Lambda^- \ni M & \cdots \\ \\ \Lambda^- \ni F(M) & \cdots \\ \\ \Lambda^- \ni F^2(M) & \cdots \\ \\ \vdots & \\ \\ \mathrm{NF}_\beta \cap \Lambda^- \ni F^n(M) & \cdots \twoheadrightarrow_\beta \end{array}$$

F is a normalising reduction strategy, *ie*, if $M \in \mathrm{WN}_\beta$ then repeated application of F on M will eventually result in a normal form. $\Lambda^- \subseteq \Lambda^K$ is closed under F and is such that if $M \in \Lambda^-$ then all redexes in M are perpetual. The diagram illustrates that $M \in \mathrm{WN}_\beta$ and $M \in \infty_\beta$ would contradict $F^n(M)$ being a normal form for some n. From this we conclude that $M \in \mathrm{UN}_\beta$.

We use *leftmost reduction* as our normalising strategy [Bar84, CF58].

Definition 3.9 The leftmost reduction strategy F_l on Λ^K is defined as follows. When $M \in \Lambda^K \setminus \mathrm{NF}_\beta$, we have

$$\begin{array}{lcl} F_l(x\ P_1\ \ldots\ P_n) & = & x\ P_1\ \ldots\ F_l(P_m)\ \ldots\ P_n \\ & & \quad \text{if } P_i \in \mathrm{NF}_\beta \text{ for } i < m \text{ and } P_m \notin \mathrm{NF}_\beta \\ F_l(\lambda x.P) & = & \lambda x.F_l(P) \\ F_l((\lambda x.P_0)\ P_1\ P_2\ \ldots\ P_n) & = & P_0\{x := P_1\}\ P_2\ \ldots\ P_n. \end{array}$$

For $M \in \mathrm{NF}_\beta$ we have $F_l(M) = M$. ♦

It is clear that $M \to_\beta F_l(M)$ using the compatibility of $\to_\beta$.

Proposition 3.10 *Let $M \in \Lambda^K$ be a* $\mathbf{K}$*-expanded term. Then $F_l(M)$ is a* $\mathbf{K}$*-expanded term.*

Proof. If $M \in \mathrm{NF}_\beta$, then $F_l(M) = M$ and is thus $\mathbf{K}$-expanded by assumption. We therefore assume $M \notin \mathrm{NF}_\beta$ and proceed by induction on the structure of M.

i. $M \equiv x\ P_1\ \ldots\ P_n$. In this case we have $F_l(M) \equiv x\ P_1\ \ldots\ F_l(P_m)\ \ldots\ P_n$ and $\mathrm{rel}(F_l(M)) = \mathrm{rel}(F_l(P_m)) \cup \bigcup_{i=1, i\neq m}^{n} \mathrm{rel}(P_i)$. We consider any $\mathbf{K}$-abstraction $\mathbf{K}\ S \in \mathrm{rel}(F_l(M))$ and divide into two cases

   (a) $\mathbf{K}\ S \in \mathrm{rel}(P_i)$, $i \neq m$. Then $\mathbf{K}\ S \in \mathrm{rel}(M)$ and is $\mathbf{K}$-expanded by assumption.

   (b) $\mathbf{K}\ S \in \mathrm{rel}(F_l(P_m))$. By Corollary 3.7 and the induction hypothesis, we note $F_l(P_m)$ is $\mathbf{K}$-expanded. We conclude that $\mathbf{K}\ S$ is $\mathbf{K}$-expanded.

   As all relevant abstractions are $\mathbf{K}$-expanded, $F_l(M)$ is $\mathbf{K}$-expanded.

ii. $M \equiv \lambda x.P$. Then $F_l(M) \equiv \lambda x.F_l(P)$. Using Corollary 3.7 and the induction hypothesis, we note that $F_l(P)$ is $\mathbf{K}$-expanded. Thus, any $\mathbf{K}$-abstraction $\mathbf{K}\ S \in \mathrm{rel}(F_l(M)) = \mathrm{rel}(F_l(P))$ is $\mathbf{K}$-expanded. We conclude that $F_l(M)$ is $\mathbf{K}$-expanded.

iii. $M \equiv (\lambda x.P_0)\, P_1\, P_2\, \ldots\, P_n$. Then $F_l(M) \equiv P_0\{x := P_1\}\, P_2\, \ldots\, P_n$. Furthermore $\mathrm{rel}(M) = \{\lambda y.Q \subseteq P_i \mid 0 \leq i \leq n\}$. We consider any $\mathbf{K}$-abstraction $\mathbf{K}\, S \subseteq F_l(M)$. We split into two cases:

(a) $\mathbf{K}\, S \subseteq P_0\{x := P_1\}$ where $\mathbf{K}\, S \not\subseteq P_1$. By the substitution generation lemma, we have a subterm $\mathbf{K}\, S' \subseteq P_0$ such that $(\mathbf{K}\, S')\{x := P_1\} \equiv \mathbf{K}\, S$. The subterm $\mathbf{K}\, S'$ is $\mathbf{K}$-expanded as it is in $\mathrm{rel}(M)$ and hence part of the redex $\mathbf{K}\, S'\, T' \subseteq P_0$. By the definition of simultaneous substitution, we conclude that

$$\begin{aligned}\mathbf{K}\, S\, (T'\{x := P_1\}) &\equiv ((\mathbf{K}\, S')\{x := P_1\})\, (T'\{x := P_1\}) \\ &= (\mathbf{K}\, S'\, T')\{x := P_1\} \\ &\subseteq P_0\{x := P_1\}\ .\end{aligned}$$

Thus $\mathbf{K}\, S$ is the abstraction of a $\mathbf{K}$-redex and is thus $\mathbf{K}$-expanded.

(b) $\mathbf{K}\, S \subseteq P_i$ with $i \geq 1$. We have $\mathbf{K}\, S \in \mathrm{rel}(M)$ and therefore $\mathbf{K}\, S$ is $\mathbf{K}$-expanded by assumption.

As all $\mathbf{K}$-abstractions are $\mathbf{K}$-expanded, in particular all relevant $\mathbf{K}$-abstractions are $\mathbf{K}$-expanded, and we conclude that $F_l(M)$ is $\mathbf{K}$-expanded.

This exhaust the possibilities and concludes the proof that $\mathbf{K}$-expandedness is conserved under leftmost reduction. □

Proposition 3.11 *Let $M \in \Lambda$ be a $\mathbf{K}$-expanded and good term. Then $F_l(M)$ is a good term.*

Proof. If $M \in \mathrm{NF}_\beta$, then $F_l(M) = M$ and is therefore good by assumption. We assume $M \notin \mathrm{NF}_\beta$ and proceed by induction on the structure of M.

i. $M \equiv x\, P_1\, \ldots\, P_n$. Then $F_l(M) \equiv x\, P_1\, \ldots\, F_l(P_m)\, \ldots\, P_n$. Consider a $\mathbf{K}$-redex $\Delta \equiv \mathbf{K}\, S\, T \subseteq F_l(M)$.

(a) $\Delta \subseteq P_i$ for some i where $i \neq m$. As $\Delta \subseteq P_i \subseteq M$ we see that it is good by assumption.

(b) $\Delta \subseteq F_l(P_m)$. By Lemma 3.2 and the induction hypothesis, $F_l(P_m)$ is good. We conclude that Δ is good.

As all $\mathbf{K}$-redexes are good, M is good.

ii. $M \equiv \lambda x.P$. We have $F_l(M) \equiv \lambda x.F_l(P)$. We consider any redex $\Delta \equiv \mathbf{K}\, S\, T$ such that $\Delta \subseteq F_l(M)$. It must be the case that $\Delta \subseteq F_l(P)$. Using Lemma 3.2 and the induction hypothesis, Δ is good as $F_l(P)$ is good. Since all $\mathbf{K}$-redexes of $F_l(M)$ are good, $F_l(M)$ is good.

iii. $M \equiv (\lambda x.P_0)\, P_1\, P_2\, \ldots\, P_n$. We have $F_l(M) \equiv P_0\{x := P_1\}\, P_2\, \ldots\, P_n$. We consider any redex $\Delta \equiv \mathbf{K}\, S\, T \subseteq F_l(M)$ and split into three cases:

(a) $\Delta \subseteq P_0\{x := P_1\}$ where $\Delta \not\subseteq P_1$. We should prove that $S \succeq_\infty T$. By the substitution generation lemma, we have a subterm $\Delta' \equiv \mathbf{K}\, S'\, T' \subseteq P_0$ such that $\Delta'\{x := P_1\} = \Delta$. As M is good, we have $S' \succeq_\infty T'$. Hence, we have, for any substitution θ, that

$$\infty_\beta \ni S\theta = (S'\{x := P_1\})\theta \Longleftarrow (T'\{x := P_1\})\theta = T\theta \in \infty_\beta$$

and we conclude that Δ is good.

(b) $\Delta \subseteq P_i$ for some i where $i \geq 1$. We have $\Delta \subseteq M$ and it is thus good by Definition 3.2.

(c) $\Delta \equiv (P_0\{x := P_1\})\, P_2 \equiv (\lambda y.Q)\, P_2$. By *reductio ad absurdum* we note that Δ cannot be a $\mathbf{K}$-redex: suppose that Δ was the $\mathbf{K}$-redex $\mathbf{K}\, Q\, P_2$. Then, by the substitution generation lemma, either $P_0 \equiv \mathbf{K}\, Q'$ where $Q \equiv Q'\{x := P_1\}$ or $P_0 \equiv x$ and $P_1 \equiv \mathbf{K}\, Q$, which in both cases would give a $\mathbf{K}$-unexpanded abstraction in M. As all abstractions in M are relevant this contradicts M being $\mathbf{K}$-expanded.

As all $\mathbf{K}$-redexes of $F_l(M)$ are good, $F_l(M)$ is good.

This exhausts the possibilities and concludes the proof that for a $\mathbf{K}$-expanded term, goodness is conserved under leftmost reduction. □

Note that as we considered arbitrary substitutions, we could compose substitutions freely in the proof above. We are now able to prove Theorem 3.8:

Proof (Theorem 3.8). Let $M \in \Lambda$ be a good and $\mathbf{K}$-expanded term. We suppose that $M \in \mathrm{WN}_\beta$. As $F_l(M)$ is normalising, we have the following finite reduction path

$$M \to_\beta F_l(M) \to_\beta F_l^2(M) \to_\beta \cdots \to_\beta F_l^n(M) \in \mathrm{NF}_\beta$$

for some $n \in \mathbb{N}$.

Using *reductio ad absurdum* we conclude that $M \in \mathrm{SN}_\beta$: We suppose that $M \in \infty_\beta$. By induction over $i \geq 0$ using the propositions 3.3, 3.10, and 3.11 we see that $F_l^i(M)$ is infinite, $\mathbf{K}$-expanded and good. In particular $F_l^n(M) \in \infty_\beta$ contradicting $F_l^n(M) \in \mathrm{NF}_\beta$. We conclude that $M \in \mathrm{SN}_\beta$. □

3.2 Conclusion

In this chapter, we have presented a technique to find a uniformly normalising subset of a λ-calculus. As seen from the proof of Proposition 3.3, our simpler version of Bergstra-Klop's Theorem came in handy to prove the conservation property. The overall idea of the proof is to find the relevant abstractions: the abstractions that could eventually become the operator of a $\mathbf{K}$-redex. We require that these abstractions are $\mathbf{K}$-expanded and thereby ensure that there is not created new $\mathbf{K}$-redex in a reduction path. In this way, a $\mathbf{K}$-redex appearing in a term at any point in the reduction point is a copy of a redex in the original term. It is therefore sufficient to require that all $\mathbf{K}$-redexes in the original term are good to ensure that a $\mathbf{K}$-redexes at any point in the reduction path is good. By the definition of goodness, it is not possible for $\mathbf{K}$-redexes to erase infinite reduction paths. This gives us the uniform normalisation.

In Chapter 6 we will give a more complicated proof of uniform normalisation for a typed λ-calculus. The proof presented here is useful as it illustrates the basic reasoning involved without all the technical details in the later proof. In this way, this chapter gives a much clearer exposition of our technique.

Chapter 4

A Calculus Corresponding to Classical First-Order Logic

"As the name suggests, proof theory studies proofs. In other words, it studies not only theorems of a theory, *what* we know in the theory, but also *how* we know the theorems." [Pra71]. One of the most surprising facts about proof theory is that through the Curry-Howard isomorphism [CF58, How80] (or see [Gal93] for a recent account) it is related to the study of the λ-calculus. An overly brief description of the isomorphism is that proofs derived in a proof theoretical system are, in a certain sense, isomorphic to type derivations in a λ-calculus. Many other notions in the two theories are related in a similar way; for instance, reduction of proofs is isomorphic to reduction in a λ-calculus. The two disciplines were developed independently, but since the "discovery" of the isomorphism the two worlds have in many ways merged.

We have several formalisations of mathematical reasoning in proof theory: Hilbert Style [Neu27, HB34], Natural Deduction [Pra65, Gen34], and Sequent Calculus [Gen34] (see [TS96] for a recent account). Sørensen and Urzycyn [SU98] introduce the different styles and their relationship. In proof theory, proofs are referred to as deductions, and theorems being proved (or rather deduced) are called formulae. One aim of proof theory[1] is to find proofs characterised by having a certain simple form, so-called normal proofs. Knowing that all deductions can be rewritten in a simple form simplifies the reasoning when analysing the structure of proofs. Proofs in normal forms are therefore useful (if not crucial) in proofs of consistency. The process of rewriting a deduction towards a normal form is called normalisation and is described by reduction rules, also called contraction rules.

In the formulation of reduction rules there is a difference in the starting point between proof theory and λ-calculus. Reduction rules in proof theory are introduced to mechanise the quest for the normal form of a deduction of a given formula. As there are absolute characterisations of normal forms, reduction rules are judged by their ability to bring a deduction closer to a normal form. On the other hand, λ-calculus is developed to describe the computational behaviour of programs. Therefore, reduction rules are introduced to model the evaluation of programs.

In the following we will consider natural deductions of first-order logic. First-order logic is described in different forms by Prawitz [Pra65, Pra71]: minimal, intuitionistic, and classical. For classical logic he describes a system where the connectives for *disjunction* (logical or: $\vee$) and *existential quantification* ($\exists$) are derived from the other connectives. He introduces different reduction rules to simplify the proof: removing detours in the deduction, restructuring the deduction in certain ways, and removing re-

[1] There are, naturally, many motivations for proof theory. Some of these are discussed in the introductions of [Pra71] and [TS96]. In the following we will only consider the proof normalisation aspect of proof theory.

dundancy, *ie*, unused parts of the deduction. He proves weak normalisation [Pra65] and strong normalisation [Pra71] of natural deduction for first-order logic. Stålmarck [Stå91] continues Prawitz' work and proves strong normalisation for a system where disjunction and existential quantification are primitives instead of being derived.

Starting with this chapter we will change our focus from untyped λ-calculus to typed λ-calculus. In this chapter we reformulate Stålmarck's system as a λ-calculus. In the two next chapters we establish some important properties for the λ-calculus. This builds up to Chapter 7 where we prove that the question of strong normalisation can be reduced to the question of weak normalisation for this system.

The first half of this chapter provides background knowledge useful for the reasoning in the following. Section 4.1 clarifies the distinction between the different forms of logic mentioned above. Section 4.2 gives a brief and informal presentation of the systems used by Prawitz and Stålmarck. The section only includes the notions that are needed for our further work and is included to give background for the formal presentation. We relate the systems presented in Section 4.2 to a λ-calculus in Section 4.3. This section contains a formal presentation of the typed λ-calculus $\Lambda_{\mathrm{M}\Delta}$ with a type system corresponding to first-order classical logic. We also introduce the restriction of $\Lambda_{\mathrm{M}\Delta}$ to the calculus Λ_{M} with a type system corresponding to first-order minimal logic. These two calculus will be our subject for the rest of the thesis. Section 4.4 demonstrates how to carry over the notion of Λ^I to Λ_{M}.

4.1 Classical vs. Intuitionistic Logic

In the field of mathematical logic we formalise human ways of reasoning, in particular the way of reasoning used in mathematics. As we in everyday life have several different ways of constructing an argument, it is not that surprising that the formalisation can be done in several ways. While the basic connectives, *implication* ($\rightarrow$), *conjunction* (logical *and*: $\wedge$), *disjunction* (logical *or*: $\vee$) are fairly straightforward, the interpretation of *negation* (logical not: $\neg$) has been much debated. On an abstract level, the question is whether we can conclude positive information from negative. For instance, from the knowledge of a person not being female we can conclude that the person is male. However, this way of reasoning is not correct in all settings. Consider the statement "He that is not with me is against me" [Matt 12:30]: people, not being with him, could simply be unaware of him and hence not directly against him.

In a formal setting, the principle we are discussing is whether we from knowing that it is false that the proposition ϕ does not hold, *ie* $\neg\neg\phi$, can conclude that ϕ holds. This reasoning principle is called the *double negation elimination*. This reasoning principle is included in the so-called *classical logic*. Classical logic is also sometimes called a closed world view as all the possibilities are considered known. In *intuitionistic logic* [van86, Tv88] we only accept the principle of "ex falso sequitur quod libet", from falsehood follows whatever you like. In *minimal logic* [Pra65] none of these principles are accepted so we cannot conclude anything from an absurdity.

Clearly, a definitive answer to which interpretation resembles the world the best cannot be given. However, the double negation elimination is generally used in mathematics though some mathematicians are reluctant to use *reductio ad absurdum*.

4.2 Natural Deduction for Classical First-Order Logic

When formalising logic the first step is propositional logic, where we reason about formulae that are true or false (hereby not said that all formulae are either true or false,

cf. the discussion on intuitionistic logic above). In propositional logic we only know the value of the formula, but not the meaning of the formula. The meaning is included in the next step where we allow propositions about objects. We then have the possibility of saying that a statement is true for some but not all objects in a universe. This is called *first-order logic*. A further generalisation is to include propositions about other propositions; this is called higher-order logic. In this thesis, we concentrate on first-order logic.

To use first-order logic we have to settle on the objects of the universe first. This is done by using a finite, first-order signature consisting of *functions* f with a fixed arity and *relations* r with a fixed arity. Nullary functions are called *constants*. We can then describe the first-order formulae ϕ by the grammar:

$$\begin{array}{rcl} t & ::= & x \mid f(t_1, \ldots, t_{n_f}) \\ \psi, \phi & ::= & \bot \mid r(t_1, \ldots, t_{n_r}) \mid \phi \to \psi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall \alpha.\psi \mid \exists \alpha.\psi \end{array}$$

where n_f is the arity of f and n_r is the arity of r. The *negation* $\neg\phi$ of a formula ϕ is defined as a shorthand for $\phi \to \bot$. The *first-order terms* t with no variables, also called *closed terms*, represent the objects of the universe. The formulae $r(t_1, \ldots, r_{n_r})$ are called *atomic*; all other formulae except $\bot$ are called *compound*. The variable α ranges over the objects described by the signature.

As an example, we can describe the natural numbers $\mathbb{N}$ with addition by the following signature:

Constant symbols : $\mathsf{0}$
Functions : $\mathsf{S}(\bullet), +(\bullet, \bullet)$
Relations : $=(\bullet, \bullet)$

Arbitrary numbers are built using the successor function S and the constant $\mathsf{0}$ for zero.

The basic properties of equality and addition can now be expressed in first-order logic by the following rules:

$$\begin{array}{l} \forall x.=(x, x) \\ \forall x.\forall y.=(x, y) \to =(y, x) \\ \forall x.\forall y\forall z.(=(x, y) \wedge =(y, z)) \to =(x, z) \\ \forall x.=\big(+(x, \mathsf{0}), x\big) \\ \forall x.\forall y.=\big(+(x, \mathsf{S}(y)), +(\mathsf{S}(x), y)\big) \\ \forall x.\forall y\forall z.=(x, y) \to =\big(+(x, z), +(y, z)\big) \end{array}$$

Taking these formulae as axioms we could derive the properties of addition, *eg*, associativity and commutativity, for concrete instances of the rules, *eg*, $1 + (2 + 3) = (1 + 2) + 3$ and $1 + 3 = 3 + 1$, using the reasoning rules presented below.[2]

In his monograph Prawitz introduces deduction rules for classical first-order logic, we have included them in Figure 4.1 for easy reference. We note that elimination rules are more complicated than the examples used in the preliminaries. In the elimination rules $\vee E$ and $\exists E$, the deduced formula is not a subformula of the major premise. The same applies to the rule $\bot$. Prawitz proves that for every deduction there is a corresponding normal deduction. This was originally proved by Gentzen [Gen34] for the sequent calculus and is now known as Gentzen's Hauptsatz.

As said in the introduction, Prawitz' and Stålmarck's proofs of normalisation builds on reduction rules for the deductions. The first of these reduction rules are based on the fact that the rules, except for $\bot$, can be divided into *introduction rules* and *elimination*

[2]The axioms presented here cannot prove associativity and commutativity in general for all natural numbers. This requires the *induction scheme* that formalises mathematical induction, but requires either an infinite number of axioms or second-order logic as it abstracts over all formulae. The induction scheme is included in Peano's axioms [Men97], which is an often used way of formalising the natural numbers.

Figure 4.1 Natural deduction for classical first-order logic. The rules are as presented by Prawitz [Pra65] except for minor syntactic changes and simplifications. In the rules $\forall I$ and $\exists E$, some restrictions apply to ensure that α is not free in an undischarged assumption.

$$\frac{A \quad B}{A \wedge B}\ \wedge I \qquad \frac{A \wedge B}{A}\ \wedge E \qquad \frac{A \wedge B}{B}\ \wedge E$$

$$\frac{A}{A \vee B}\ \vee I \qquad \frac{B}{A \vee B}\ \vee I \qquad \frac{A \vee B \qquad \begin{matrix}(A)\\ C\end{matrix} \qquad \begin{matrix}(B)\\ C\end{matrix}}{C}\ \vee E$$

$$\frac{\begin{matrix}(A)\\ B\end{matrix}}{A \to B}\ \to I \qquad \frac{A \to B \quad A}{B}\ \to E$$

$$\frac{A}{\forall \alpha.A}\ \forall I \qquad \frac{\forall \alpha.A}{A_t^\alpha}\ \forall E$$

$$\frac{A_t^\alpha}{\exists \alpha.A}\ \exists I \qquad \frac{\exists \alpha.A \qquad \begin{matrix}(A)\\ B\end{matrix}}{B}\ \exists E$$

$$\frac{\begin{matrix}(\neg A)\\ \bot\end{matrix}}{A}\ \bot$$

Figure 4.2 Rewriting rules for deductions in classical first-order logic. The figure follows the presentation in [Stå91] of Prawitz' rules [Pra65, pp. 36–38]. The motivation of the rules are found in the latter reference. The rules can be applied on any subdeduction of a deduction.

$$\dfrac{\dfrac{\dfrac{\Sigma_1}{A_1}\quad\dfrac{\Sigma_2}{A_2}}{A_1 \wedge A_2}}{A_i} \quad\rightarrow\quad \dfrac{\Sigma_i}{A_i} \qquad i = 1, 2$$

$$\dfrac{\dfrac{\dfrac{\Sigma}{A_i}}{A_1 \vee A_2}\quad\dfrac{\begin{matrix}[A_1]\\ \Sigma_1\end{matrix}}{B}\quad\dfrac{\begin{matrix}[A_2]\\ \Sigma_2\end{matrix}}{B}}{B} \quad\rightarrow\quad \dfrac{\begin{matrix}\Sigma\\ [A_i]\\ \Sigma_i\end{matrix}}{B} \qquad i = 1, 2$$

$$\dfrac{\dfrac{\dfrac{\begin{matrix}[A]\\ \Sigma_1\end{matrix}}{B}}{A \rightarrow B}\quad\dfrac{\Sigma_2}{A}}{B} \quad\rightarrow\quad \dfrac{\begin{matrix}\Sigma_2\\ [A]\\ \Sigma_1\end{matrix}}{B}$$

$$\dfrac{\dfrac{\dfrac{\Sigma_1(\alpha)}{A}}{\forall\alpha.A}}{A^\alpha_t} \quad\rightarrow\quad \dfrac{\Sigma_1(t)}{A^\alpha_t}$$

$$\dfrac{\dfrac{\dfrac{\Sigma_1}{A^\alpha_t}}{\exists\alpha.A}\quad\dfrac{\begin{matrix}[A]\\ \Sigma_2(\alpha)\end{matrix}}{B}}{B} \quad\rightarrow\quad \dfrac{\begin{matrix}\Sigma_1\\ [A^\alpha_t]\\ \Sigma_2(t)\end{matrix}}{B}$$

rules. In some cases when an introduction rule appears as a premise to an elimination rule, the rule can be simplified; the reduction rules for doing so are presented in Figure 4.2.

In classical natural deduction systems without the connective $\vee$ and the quantifier $\exists$, the $\bot$-rule can be restricted to atomic formulae without loss of reasoning power [Pra65, pp. 39–40]. In our system, containing this connective and this quantifier, this no longer holds [Stå91, p. 131]. However, following an idea used by Statman [Sta78, Sta74], we can rewrite a compound formula to an atomic formula if the compound formula is at the same time the conclusion of an application of the $\bot$-rule and a major premise of an elimination rule. This is encompassed in the rules of Figure 4.3 taken from [Pra65, Stå91].

In the proof of strong normalisation Stålmarck [Stå91] encounters the problem of so-called maximum segments [GLT89, p. 78]. In short, the problem is that in the elimination rules of disjunction and existential quantification, the consequence is not a subformula of the formula containing the disjunction and the existential quantification, resp. To circumvent this we introduce the *permutative reductions* of Figure 4.4.[3] These rules are called commuting conversions by Girard et al.. [GLT89]. Furthermore, Stålmarck introduces reduction rules for *redundant applications* of the negation rule, Figure 4.5. These play a crucial rôle in his proof of strong normalisation: At certain points in the proof, reductions to make the consequence of a negation rule atomic are followed by these reductions. This makes the structural induction in his proof work. However, as we shall discuss in Section 4.3.2, their computational value is limited.

Stålmarck proves that the reduction rules presented in this section are strongly normalising.

4.3 Translation to the λ-Calculus

In this subsection we define $\Lambda_{\mathrm{M}\Delta}$, the λ-calculus corresponding to classical first-order logic. There are many examples [TS96, GLT89, SU98] of term assignments of first-order classical logic in the literature. However, the full system is rarely used as the connectives $\wedge$, $\vee$, and $\exists$ are definable from $\rightarrow$, $\forall$, and $\bot$. Hence, there is no agreement on how to denote the rarely used terms like pairs and injections. We will follow this line and introduce yet another way of denoting the terms.

Before giving the formal definition, it is appropriate to discuss the intuitive meaning of the types and the corresponding terms. Overall, the language presented constitutes a higher-order typed functional language. Many important features of "real" languages, like for instance ML [Pau91], are thus described:

- Implication corresponds to the type for functions. We only have nameless function abstraction $\lambda x.P$ constructing a function, and application $P\,Q$ destroying a function.
- *Universal quantification* corresponds to types quantified by a term and thus corresponds to so-called *dependent types*. The name stems from the fact that a type *depends* on a term. As a concrete example[4], we can consider a program manipulating n-ary vectors. Vectors of different lengths are incompatible, and it is therefore reasonable to let the type of such a vector depend on the length n, *eg*, $\mathsf{vector}(n)$.

[3] When comparing the reduction rules in Figure 4.4 with the presentation in [Stå91, p. 136], there is the additional restriction in the presentation here that B should be the major premise. Private communication with Stålmarck confirms that with this restriction the system of reduction rules is still strong enough to normalise any deduction. The restriction is therefore "a natural one". Furthermore, as discussed on Page 39, the restriction is necessary to avoid infinite reduction paths.

[4] This example is taken from [Mit96].

Figure 4.3 Rewriting rules to make the consequences of $\bot$-rule atomic. The rules are taken from [Stå91]. Note that the deduction on the form

$$\frac{\bot}{[\neg(\cdots)]}$$

in each of the right hand sides is an application of the atomic deduction $\rightarrow I$.

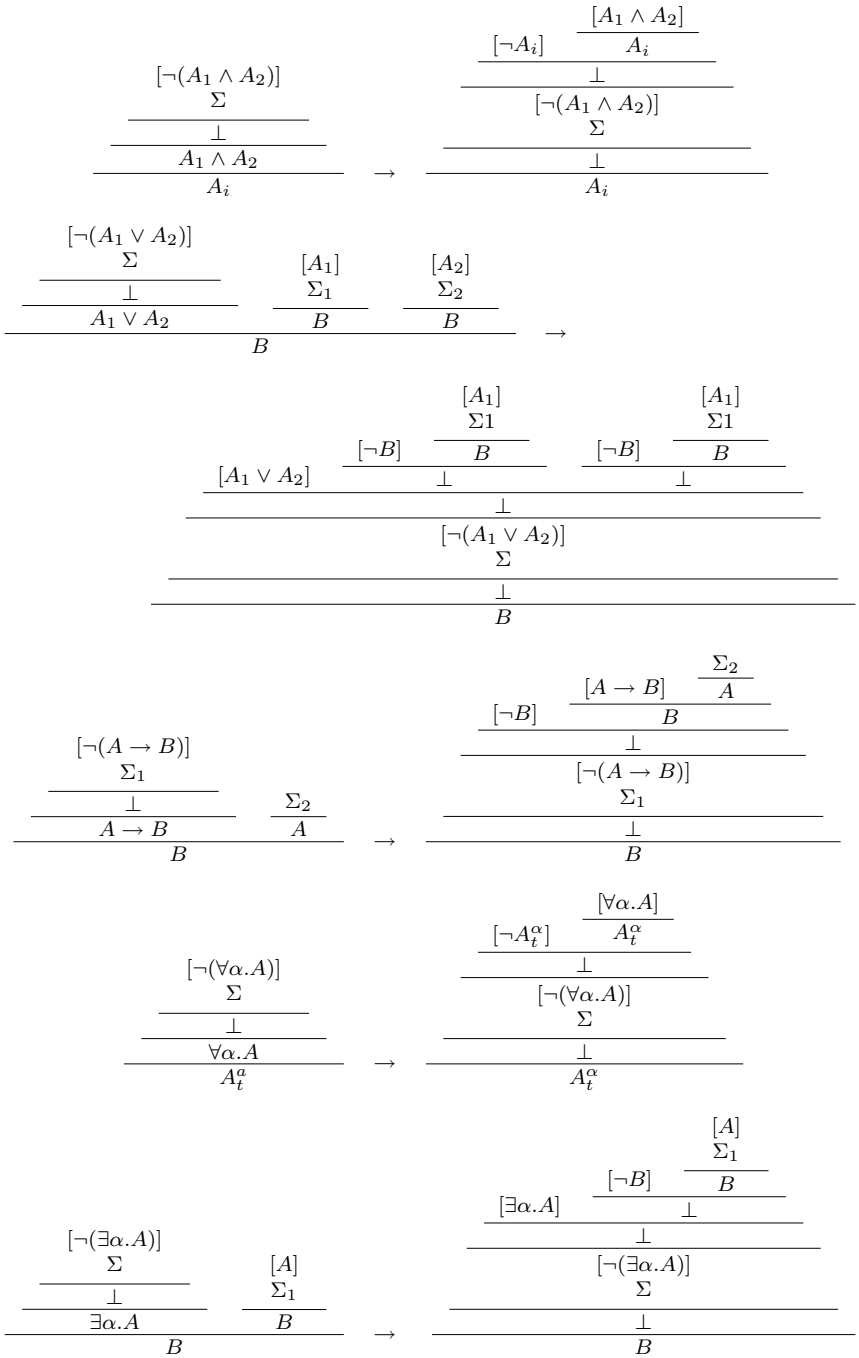


Figure 4.4 Permutative reductions as presented by Stålmarck [Stå91] as permutative contractions. We have left out some restrictions on the application of the rules as they are not important for our further development. The rules use a meta-notation where R stands for any elimination rule with $\Pi_1, \ldots, \Pi_n$ representing the minor premises. See also footnote on Page 30.

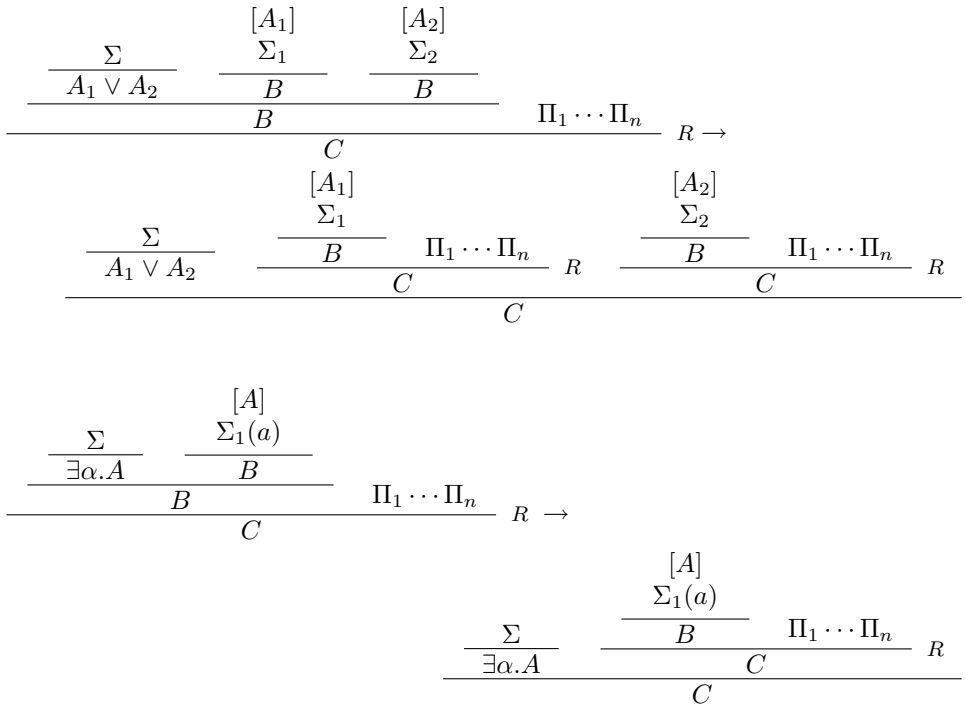
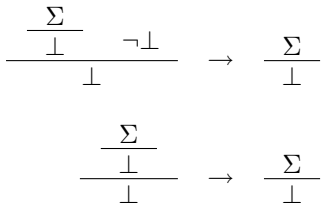


Figure 4.5 Reduction of redundant applications of the negation rule [Stå91] (called contraction of redundant applications of the negation rule there). In the last reduction rule, no assumption is discharged by the last application of the negation rule in the left deduction.



Using the universal quantification type $\forall n : \mathsf{int.vector}(n)$, we can define a function returning the zero vector of any length n. Only few languages include dependent types in this form where the type depends on computations in the program; object oriented languages like C++ [Str97], where the constructor of the object can be provided with a calculated value, seem to be the exception.

General dependent types are formalised by the system λP [Bar92]. For our purpose, general dependent types complicate matters, as types and terms are intertwined. Fortunately, we shall only need dependent types in a more restricted form where we abstract over relations of first-order logic. This gives us types depending on constants. This is comparable to arrays in Pascal where the array size cannot be evaluated at run-time, but should be specified at compile-time. We will use $\Lambda\alpha.M$ for abstractions over first-order terms and Mt for application of such an abstraction.

- The type corresponding to *existential quantification* $\exists\alpha.\tau$ seems more artificial. With inspiration from the constructions used in intuitionistic logic, an object of this type is a packet of a first-order term t to be substituted for α and a term P of the resulting type $\tau(t)$. It is thus a special form of pairs, which we will denote by $[\![P, t]\!]$. The destructor, which we also call unpacking, of these pairs is the term $\mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$ where P is of existential type.

- The type corresponding to *conjunction* $\tau \wedge \sigma$ gathers two separate types. It is seen that this corresponds to a type for tuples. We have pairs $\langle P, Q\rangle$ constructing a tuple, and projections $\pi_i(P)$ destructing them.

- The type corresponding to *disjunction* $\tau \vee \sigma$ is also a type gathering of two separate types. However, we can only access the one *or* the other. It thus corresponds to datatype-declaration defining a type consisting of several distinct classes; this is also known as sums. It is not possible to construct abstract data types, as the type system corresponding to first-order logic does not contain abstraction over types. To describe abstract data types, a type system corresponding to second-order logic is needed [Mit96, pp. 679 ff].

  Injections $\mathrm{in}_i(P)$ construct an instance of a disjunction type and with P being of type $\tau \vee \sigma$ the case-expression $\mathrm{case}(P,\, x.Q,\, x.R)$ destructs it again.

- The type corresponding to $\bot$ does not in itself have an associated term. In classical logic any formula can, given the right assumptions, be derived from this type through the negation rule $\bot$. If a $\Lambda_{\mathrm{M}\Delta}$-term types to $\bot$ we will be able to derive a term of any type. We denote the derived term as $\Delta x.M$ which, as discussed by Griffin [Gri90], corresponds to exception handling.

Let us now turn to the formal definitions inspired by [RS93, TS96, GLT89]. We have not translated the reductions of redundant applications of the negation rule found in Figure 4.5. As we will discuss in detail below, after the formal definitions, their inclusion would require a Church-style calculus, in which the types of the abstraction variables are explicit. Finally, it is questionable whether our technique would be able to handle these reductions.

Definition 4.1 (Terms and types of $\Lambda_{\mathrm{M}\Delta}$)

i. Let $\mathbb{V}_{\mathbb{T}}$ be an infinite, countable alphabet of *type variables* ranged over by α and β, and let there for each $n \geq 0$ be given an infinite, countable set of n-ary function symbols; the symbols are denoted f^n . The set of *eigenterms* $\mathbb{T}$ is defined by the following grammar:

$$\mathbb{T} \ni t ::= \alpha \mid f^n(t_1, \ldots, t_n) \ .$$

Arbitrary eigenterms are denoted by t and s, possibly primed or indexed.

ii. Let there be given an infinite, countable set of n-ary relation symbols; the relation symbols are written r^n. The set of *types* $L_{\mathrm{M}\Delta}$ on $\Lambda_{\mathrm{M}\Delta}$ is defined by the following grammar:

$$L_{\mathrm{M}\Delta} \ni \tau ::= \bot \mid r^n(t_1, \ldots, t_n) \mid (\tau \to \tau) \mid (\tau \wedge \tau) \mid (\tau \vee \tau) \mid \forall\alpha.\tau \mid \exists\alpha.\tau \ .$$

Arbitrary types are denoted by τ, σ, or ϑ, possibly primed or indexed. Outermost parentheses are omitted, $\to$ associates to the right, and $\neg\phi \stackrel{\text{def}}{\equiv} \phi \to \bot$.

iii. Let $\mathbb{V}_{\Lambda}$ be an infinite, countable alphabet of *term variables* ranged over by x, y, z, u, and v, all possibly primed or indexed. The *terms* of $\Lambda_{\mathrm{M}\Delta}$ are generated by the following grammar

$$\begin{array}{rcl} \Lambda_{\mathrm{M}\Delta} \ni M & ::= & x \\ & \mid & \lambda x.M \mid M\ M \\ & \mid & \langle M, M\rangle \mid \pi_i(M) \\ & \mid & \mathrm{in}_i(M) \mid \mathrm{case}(M,\, x.M,\, x.M) \\ & \mid & \Lambda\alpha.M \mid Mt \\ & \mid & [\![M, t]\!] \mid \mathrm{let}\ [\![x, \alpha]\!] = M\ \mathrm{in}\ M \\ & \mid & \Delta x.M \end{array}$$

where $i = 1, 2$.

Arbitrary terms are denoted M, N, P, Q, and R; all of these possibly indexed or primed. The *subterm relation* $\subseteq$ is defined in the obvious manner from the grammar. ♦

The terms $\lambda x.M$, and $\langle M, M\rangle$, and $\mathrm{in}_i(M)$, and $\Lambda\alpha.M$, and $[\![M, t]\!]$ are the *constructors* of β-redexes, while $M\ M$, and $\pi_i(M)$, and $\mathrm{case}(M,\, x.M,\, x.M)$, and $\mathrm{let}\ [\![x, \alpha]\!] = M\ \mathrm{in}\ M$, and Mt are *destructors* of β-redexes.

The term assignment above follows Rehof and Sørensen [RS93] closely. They use $\lambda^{\forall}\alpha.C$ instead of $\Lambda\alpha.M$, which we have chosen to use to stress the relationship with λP. Felleisen et al. [FH92] introduce the control operator $\mathcal{C}$, and Griffin [Gri90] shows how to type it. Troelstra et al. [TS96] use $\mathbf{p}(t, s)$ for pairs of both conjunction and existential quantification type, $\mathbf{p}_j(t)$ for projections, $\mathbf{k}_j(t)$ for injections, $E^{\vee}_{u,v}(t, s, s')$ for case expressions where u and s describe the first case and v and s' the second, and $E^{\exists}_{u,v}(t, s)$ to unpack. Girard et al. [GLT89] write injections ι^1 and ι^2, resp., and case-expressions as $\delta\ x.u\ y.v\ t$ where t is the condition and $x.u$ and $y.v$ are the two cases. As seen from the last two references, different abstraction variables are normally used in the case-expressions. We have used the variable convention to assume the equality of the two variables as this will decrease the number of special cases below.

We adapt the notion from classical first-order logic and use *minor* about the subterms Q in the application $P\,Q$, the case-expression $\mathrm{case}(P,\, x.Q,\, x.Q')$, and the unpacking $\mathrm{let}\ [\![x, \alpha]\!] = P\ \mathrm{in}\ Q$. All other subterms are *major*. In a redex, the *operator* is the major subterm of the destructor of the redex.

We continue the development, and define the sets of free variables. When finding the free variables of a term we only consider the free *term* variables.

Definition 4.2 (Free variables in $\Lambda_{\mathrm{M}\Delta}$ terms and types)

i. The set of *free variables* of a term is defined inductively by

$$\begin{array}{ll}
\mathrm{FV}(x) & = \{x\} \\
\mathrm{FV}(\lambda x.P) = \mathrm{FV}(\Delta x.P) & = \mathrm{FV}(P) \setminus \{x\} \\
\mathrm{FV}(P\,Q) = \mathrm{FV}(\langle P, Q\rangle) & = \mathrm{FV}(P) \cup \mathrm{FV}(Q) \\
\mathrm{FV}(\pi_i(P)) = \mathrm{FV}(\mathrm{in}_i(P)) = \mathrm{FV}(\Lambda\alpha.P) & = \\
\quad \mathrm{FV}(P\,t) = \mathrm{FV}(\llbracket P, t\rrbracket) & = \mathrm{FV}(P) \\
\mathrm{FV}(\mathrm{case}(P,\, x.Q,\, x.R)) & = \mathrm{FV}(P) \cup \big((\mathrm{FV}(Q) \cup \mathrm{FV}(R)) \setminus \{x\}\big) \\
\mathrm{FV}(\mathrm{let}\,\llbracket x, \alpha\rrbracket = P\,\mathrm{in}\,Q) & = \mathrm{FV}(P) \cup (\mathrm{FV}(Q) \setminus \{x\})
\end{array}$$

ii. The set of free variables of a type is defined by:

$$\begin{array}{ll}
\mathrm{FV}(\alpha) & = \{\alpha\} \\
\mathrm{FV}(f^n(t_1, \ldots, t_n)) = \mathrm{FV}(r^n(t_1, \ldots, t_n)) & = \mathrm{FV}(t_1) \cup \cdots \cup \mathrm{FV}(t_n) \\
\mathrm{FV}(\bot) & = \emptyset \\
\mathrm{FV}(\phi \to \tau') = \mathrm{FV}(\phi \wedge \tau) = \mathrm{FV}(\phi \vee \tau) & = \mathrm{FV}(\phi) \cup \mathrm{FV}(\tau) \\
\mathrm{FV}(\forall\alpha.\tau) = \mathrm{FV}(\exists\alpha.\tau) & = \mathrm{FV}(\tau) \setminus \{\alpha\}
\end{array}$$

♦

When defining *substitution in terms* it is important to notice that we do not substitute terms in eigenterms. We thus have

$$\begin{array}{lcl}
\llbracket P, t\rrbracket\theta & = & \llbracket P\theta, t\rrbracket \\
(\mathrm{let}\,\llbracket x, \alpha\rrbracket = Q\,\mathrm{in}\,P)\theta & = & \mathrm{let}\,\llbracket x, \alpha\rrbracket = Q\theta\,\mathrm{in}\,P\theta\ ,
\end{array}$$

while the rest of the cases are straightforward extensions of Definition 1.2.

Definition 4.3 (Type substitution in $\Lambda_{\mathrm{M}\Delta}$) *Substitution of a type $\sigma \in L_{\mathrm{M}\Delta}$* for a type variable α in a type $\tau \in L_{\mathrm{M}\Delta}$, written $\tau\{\alpha := \sigma\}$, is defined by the following equations:

$$\begin{array}{lcl}
\beta\{\alpha := \sigma\} & = & \begin{cases} \sigma & \text{if } \alpha = \beta \\ \beta & \text{otherwise} \end{cases} \\
(f^n(t_1, \ldots, t_n))\{\alpha := \sigma\} & = & f^n(t_1\{\alpha := \sigma\}, \ldots, t_n\{\alpha := \sigma\}) \\
\bot\{\alpha := \sigma\} & = & \bot \\
(\tau \to \tau')\{\alpha := \sigma\} & = & \tau\{\alpha := \sigma\} \to \tau'\{\alpha := \sigma\} \\
(\tau \wedge \tau')\{\alpha := \sigma\} & = & \tau\{\alpha := \sigma\} \wedge \tau'\{\alpha := \sigma\} \\
(\tau \vee \tau')\{\alpha := \sigma\} & = & \tau\{\alpha := \sigma\} \vee \tau'\{\alpha := \sigma\} \\
(\forall\beta.\tau)\{\alpha := \sigma\} & = & \forall\beta.\tau\{\alpha := \sigma\} \\
\exists\beta.\tau\{\alpha := \sigma\} & = & \exists\beta.\tau\{\alpha := \sigma\}
\end{array}$$

♦

The next step is to define the typing relation on $\Lambda_{\mathrm{M}\Delta}$. For this purpose we need to introduce some auxiliary notation.

Definition 4.4 (Type contexts)

i. The set $\Gamma_{\mathrm{M}\Delta}$ of *contexts of types* on $\Lambda_{\mathrm{M}\Delta}$ is the set of all pairs $(x, \tau) \subseteq \mathbb{V}_{\mathbb{T}} \times L_{\mathrm{M}\Delta}$ of the form

$$\{x_1 : \tau_1, \ldots, x_n : \tau_n\}\ ,$$

where $x_i \neq x_j$ for $i \neq j$, and $\tau_i \in L_{\mathrm{M}\Delta}$.

Arbitrary contexts are denoted Γ, Δ, or Ξ, possibly primed or subscripted. We write $x : \tau$ for $\{x : \tau\}$ and Γ, Δ for $\Gamma \cup \Delta$ whenever $\mathrm{dom}\,\Gamma \cap \mathrm{dom}\,\Delta = \emptyset$.

ii. For a context $\Gamma \in \Gamma_{\mathrm{M}\Delta}$ of types there are the following operations: the *domain* is denoted $\operatorname{dom}\Gamma$, the *range* is denoted $|\Gamma|$, the *substitution of a type* σ for a type variable α is denoted $\Gamma\{\alpha := \sigma\}$, and the set of free variables is denoted $\mathrm{FV}(\Gamma)$. The operations are defined by the following equations:

$$\begin{array}{lcl} \operatorname{dom}\Gamma & = & \{x \mid x : \tau \in \Gamma\} \\ |\Gamma| & = & \{\tau \mid x : \tau \in \Gamma\} \\ \Gamma\{\alpha := \sigma\} & = & \{x : \tau\{\alpha := \sigma\} \mid x : \tau \in \Gamma\} \\ \mathrm{FV}(\Gamma) & = & \bigcup\{\mathrm{FV}(\tau) \mid x : \tau \in \Gamma\} \end{array}$$

♦

With all the technical machinery established we can now define the typing relation.

Definition 4.5 The *typability relation* $\vdash \subseteq \Gamma_{\mathrm{M}\Delta} \times \Lambda_{\mathrm{M}\Delta} \times L_{\mathrm{M}\Delta}$ on $\Lambda_{\mathrm{M}\Delta}$ is defined by

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ \mathrm{var}$$

$$\frac{\Gamma, x : \tau \vdash P : \sigma}{\Gamma \vdash \lambda x.P : \tau \to \sigma}\ {\to}I$$

$$\frac{\Gamma \vdash P : \tau \to \sigma \quad \Gamma \vdash Q : \tau}{\Gamma \vdash P\,Q : \sigma}\ {\to}E$$

$$\frac{\Gamma \vdash P : \tau \quad \Gamma \vdash Q : \sigma}{\Gamma \vdash \langle P, Q\rangle : \tau \wedge \sigma}\ {\wedge}I$$

$$\frac{\Gamma \vdash P : \tau_1 \wedge \tau_2}{\Gamma \vdash \pi_i(P) : \tau_i}\ {\wedge}E \qquad i = 1, 2$$

$$\frac{\Gamma \vdash P : \tau_i}{\Gamma \vdash \mathrm{in}_i(P) : \tau_1 \vee \tau_2}\ {\vee}I \qquad i = 1, 2$$

$$\frac{\Gamma \vdash P : \sigma \vee \vartheta \quad \Gamma, x : \sigma \vdash Q : \tau \quad \Gamma, x : \vartheta \vdash R : \tau}{\Gamma \vdash \mathrm{case}(P,\, x.Q,\, x.R) : \tau}\ {\vee}E$$

$$\frac{\Gamma \vdash P : \tau}{\Gamma \vdash \Lambda\alpha.P : \forall\alpha.\tau}\ \forall I \qquad \alpha \notin \mathrm{FV}(\Gamma)$$

$$\frac{\Gamma \vdash P : \forall\alpha.\tau}{\Gamma \vdash P\,t : \tau\{\alpha := t\}}\ \forall E$$

$$\frac{\Gamma \vdash P : \tau\{\alpha := t\}}{\Gamma \vdash [\![P, t]\!] : \exists\alpha.\tau}\ \exists I$$

$$\frac{\Gamma \vdash P : \exists\alpha.\sigma \quad \Gamma, x : \sigma \vdash Q : \tau}{\Gamma \vdash \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q : \tau}\ \exists E \qquad \alpha \notin \mathrm{FV}(\tau) \cup \mathrm{FV}(\Gamma)$$

$$\frac{\Gamma, x : \tau \to \bot \vdash P : \bot}{\Gamma \vdash \Delta x.P : \tau}\ \bot$$

A term $M \in \Lambda_{\mathrm{M}\Delta}$ is *typable*, if, and only if, $\Gamma \in \Gamma_{\mathrm{M}\Delta}$ and there exists $\tau \in L_{\mathrm{M}\Delta}$ such that $\Gamma \vdash M : \tau$. ♦

The typability relation follows the natural deduction rules. We note from Definition 4.4.i of Γ, Δ that $x \notin \operatorname{dom}\Gamma$ in the rules *var*, ${\to}I$, $\vee E$, $\exists E$, and $\bot$.

Lemma 4.6 (Weakening) *Let $M \in \Lambda_{\mathrm{M}\Delta}$ be a typable term having the type derivation $\Gamma \vdash M : \tau$. We then have*

$$\Gamma, x : \sigma \vdash M : \tau$$

for any any term variable $x \notin \operatorname{dom}\Gamma$ and any type $\sigma \in L_{\mathrm{M}\Delta}$.

Proof. The lemma is intuitively easy to understand from the typability definition: the type context is only extended with bound variables, which by the variable convention are different from x. Formally the proof is done by induction on the type derivation. We omit this. □

The language $\Lambda_{\mathrm{M}\Delta}$ has many of the flavours of a real functional language. However, it is important to notice that it is not polymorphic. In particular, we cannot type the term $\omega \equiv \lambda x.xx$: to derive a type for ω we should derive a type for each of the x's, but if we derive, say, type α for the second x, then the first x should have $\alpha \to \beta$ for some β, and $\alpha \to \beta \neq \alpha$. This can be solved by for instance polymorphic type inference where x, intuitively, can be given type $\alpha \to \alpha$. When typing the first x, type $\alpha \to \alpha$ can be instantiated to $(\alpha \to \alpha) \to (\alpha \to \alpha)$. Polymorphic type inference is formalised in the so-called polymorphic λ-calculus $\lambda 2$ [Rey74, Gir86, GLT89]. It should be noted that the polymorphic type systems found in for instance ML is a restriction of the typing system $\lambda 2$, and therefore ω cannot be typed there either. A discussion of the differences can be found in [Mit96]. However, still in $\lambda 2$ the infinite term Ω cannot be typed. The typing of Ω requires recursive types [CC91], but in these systems not all terms are weakly normalising and then the Barendregt-Geuvers-Klop conjecture is trivially true.

4.3.1 Reduction Rules

The next step is to translate Stålmarck's reductions to $\Lambda_{\mathrm{M}\Delta}$. The overall idea is to exploit the Curry-Howard isomorphism and consider each of the two sides of the rules as a type derivation for a $\Lambda_{\mathrm{M}\Delta}$-term. In spelling out the reduction rules, we then consider which $\Lambda_{\mathrm{M}\Delta}$-terms can have these type derivations. As an example, we consider the permutative reduction of existential quantification and disjunction. The left-hand side corresponds to the following type derivation:

$$\dfrac{\dfrac{\dfrac{\Sigma}{P : \exists\alpha A} \qquad \dfrac{\begin{matrix}x : [A] \\ \Sigma_1(\alpha)\end{matrix}}{Q : B_1 \vee B_2}}{\operatorname{let}\llbracket x, \alpha\rrbracket = P \operatorname{in} Q : B_1 \vee B_2} \qquad \dfrac{\begin{matrix}y : [B_1] \\ \Pi_1\end{matrix}}{R_1 : C} \qquad \dfrac{\begin{matrix}y : [B_2] \\ \Pi_2\end{matrix}}{R_2 : C}}{\operatorname{case}(\operatorname{let}\llbracket x, \alpha\rrbracket = P \operatorname{in} Q,\ y.R_1,\ y.R_2) : C}$$

where a term variable used in an assumption means that the variable is assumed to have the specified type. Similarly, the right-hand side corresponds to the type derivation:

$$\dfrac{\dfrac{\Sigma}{P : \exists\alpha.A} \qquad \dfrac{\dfrac{\begin{matrix}x : [A] \\ \Sigma_1(\alpha)\end{matrix}}{Q : B_1 \vee B_2} \qquad \dfrac{\begin{matrix}y : [B_1] \\ \Pi_1\end{matrix}}{R_1 : C} \qquad \dfrac{\begin{matrix}y : [B_2] \\ \Pi_2\end{matrix}}{R_2 : C}}{\operatorname{case}(Q,\ y.R_1,\ y.R_2) : C}}{\operatorname{let}\llbracket x, \alpha\rrbracket = P \operatorname{in} \operatorname{case}(Q,\ y.R_1,\ y.R_2) : C}$$

From these two derivations, we get the reduction rule

$$\operatorname{case}(\operatorname{let}\llbracket x, \alpha\rrbracket = P \operatorname{in} Q,\ y.R_1,\ y.R_2) \to \operatorname{let}\llbracket x, \alpha\rrbracket = P \operatorname{in} \operatorname{case}(Q,\ y.R_1,\ y.R_2)$$

corresponding to one of the permutative reductions.

For the sake of the reductions to atomise a $\bot$-conclusion (Figure 4.3) and the permutative reductions (Figure 4.4), we introduce the one-level destructor contexts inspired by [RS93].

Definition 4.7 (One-level Destructor Contexts)

i. The set of *one-level destructor contexts* $\mathcal{D}_1$ is defined by the following grammar:

$$\mathcal{D}_1 \ni D_1 ::= [] \; P \mid \pi_i([]) \mid \mathrm{case}([],\, x.P,\, x.Q) \mid [] \; t \mid \mathrm{let}\, [\![x, \alpha]\!] = [] \,\mathrm{in}\, P \;\; .$$

ii. The Δ-transformation of one-level destructor context is the mapping $(\bullet)^{\Delta}(\bullet)$ from $\mathcal{D}_1 \times \mathbb{V}_\Lambda$ to the contexts of $\Lambda_{\mathrm{M}\Delta}$ defined by:

$$\begin{array}{rcl}
([] \; P)^{\Delta}(u) & \longmapsto & u \; ([] \; P) \\
(\pi_i([]))^{\Delta}(u) & \longmapsto & u \; (\pi_i([])) \\
(\mathrm{case}([],\, x.P,\, x.Q))^{\Delta}(u) & \longmapsto & \mathrm{case}([],\, x.u\, P,\, x.u\, Q) \\
([] \; t)^{\Delta}(u) & \longmapsto & u \; ([] \; t) \\
(\mathrm{let}\, [\![x, \alpha]\!] = [] \,\mathrm{in}\, P)^{\Delta}(u) & \longmapsto & \mathrm{let}\, [\![x, \alpha]\!] = u \; [] \,\mathrm{in}\, P \qquad .
\end{array}$$

We define $D_1^{\Delta}[M, u]$ to be a shorthand for $((D_1)^{\Delta}(u))[M]$. ⧫

As usual, we will extend the notion of subterms and reductions to these contexts. The name Δ-transformation is introduced, as we will use the transformation to describe the Δ-reduction rules introduced below.

Definition 4.8 (Notions of reductions in $\Lambda_{\mathrm{M}\Delta}$) There are the following notions of reduction in $\Lambda_{\mathrm{M}\Delta}$:

i. The *β-reductions*:

$$\begin{array}{lll}
(\lambda x.P)\, Q & \beta_\lambda & P\{x := Q\} \\
\pi_i(\langle P_1, P_2 \rangle) & \beta_{\langle\rangle} & P_i \\
\mathrm{case}(\mathrm{in}_i(P),\, x.Q_1,\, x.Q_2) & \beta_{\mathrm{in}} & Q_i\{x := P\} \\
(\Lambda\alpha.P)\, t & \beta_\Lambda & P\{\alpha := t\} \\
\mathrm{let}\, [\![x, \alpha]\!] = [\![P, t]\!] \,\mathrm{in}\, Q & \beta_{[\![]\!]} & Q\{\alpha := t, x := P\}
\end{array}$$

The combined reduction $\beta_\lambda \cup \beta_{\langle\rangle} \cup \beta_{\mathrm{in}} \cup \beta_\Lambda \cup \beta_{[\![]\!]}$ is denoted by β.

ii. The *Δ-reductions*:

$$D_1[\Delta x.P] \quad \Delta \quad \Delta u.P\{x := \lambda y.D_1^{\Delta}[y, u]\}$$

iii. The *permutative reductions*:

$$D_1[\mathrm{case}(P,\, x.Q,\, x.R)] \quad \mathrm{p}_{\mathrm{in}} \quad \mathrm{case}(P,\, x.D_1[Q],\, x.D_1[R])$$

$$D_1[\mathrm{let}\, [\![x, \alpha]\!] = P \,\mathrm{in}\, Q] \quad \mathrm{p}_{[\![]\!]} \quad \mathrm{let}\, [\![x, \alpha]\!] = P \,\mathrm{in}\, D_1[Q]$$

The combined reduction $\mathrm{p}_{\mathrm{in}} \cup \mathrm{p}_{[\![]\!]}$ is denoted by p. ⧫

The β and Δ reductions are inspired by [RS93]. Our calculus is in fact more "aggressive" than Stålmarck's reduction rules, as we have no side conditions on the permutative reductions (see also the discussion on Page 39).

We see that a $\Lambda_{\mathrm{M}\Delta}$-term M will have one of the following forms:

$$M ::= x \mid \lambda x.P \mid \langle P, Q\rangle \mid \mathrm{in}_i(P) \mid \Lambda\alpha.P \mid [\![P, t]\!] \mid \Delta x.P \mid D_1[P] \tag{7}$$

where P and Q are any terms. In all but the last case a $\beta\Delta\mathrm{p}$-reduction can only occur in a subterm. This simple characterisation comes in handy when proving properties of $\beta\Delta\mathrm{p}\pi$-reduction. As the reduction rules are derived from the reduction rules for deductions, it is evident that they have the subject reduction property. We therefore state the following:

Proposition 4.9 (Subject reduction) *Let $M, N \in \Lambda_{\mathrm{M}\Delta}$ be such that $M \to_{\beta\Delta\mathrm{p}} N$. If M is typable with type context Γ and type τ, then N is typable with the same type context and type.*

Without the use of one-level Δ-transformation of the contexts, the Δ-reductions read

$$\begin{array}{lll}
(\Delta x.P)Q & \Delta_\lambda & \Delta u.P\{x := \lambda y.u\,(y\,Q)\} \\
\pi_i(\Delta x.P) & \Delta_{\langle\rangle} & \Delta u.P\{x := \lambda y.u\,\pi_i(y)\} \\
\mathrm{case}(\Delta x.P,\, z.Q,\, z.R) & \Delta_{\mathrm{in}} & \Delta u.P\{x := \lambda y.\mathrm{case}(y,\, z.u\,Q,\, z.u\,R)\} \\
(\Delta x.P)\,t & \Delta_\Lambda & \Delta u.P\{x := \lambda y.u\,(y\,t)\} \\
\mathrm{let}\,[\![z, \alpha]\!] = \Delta x.P\,\mathrm{in}\,Q & \Delta_{[\![]\!]} & \Delta u.P\{x := \lambda y.\mathrm{let}\,[\![z, \alpha]\!] = y\,\mathrm{in}\,u\,Q\}
\end{array}$$

Similarly the p_{in}-reductions read

$$\begin{array}{lll}
\mathrm{case}(P,\, x.Q_1,\, x.Q_2)R & p_{\mathrm{in}_\lambda} & \mathrm{case}(P,\, x.Q_1\,R,\, x.Q_2\,R) \\
\pi_i(\mathrm{case}(P,\, x.Q_1,\, x.Q_2)) & p_{\mathrm{in}_{\langle\rangle}} & \mathrm{case}(P,\, x.\pi_i(Q_1),\, x.\pi_i(Q_2)) \\
\mathrm{case}(\mathrm{case}(P,\, x.Q_1,\, x.Q_2),\, y.R_1,\, y.R_2) & p_{\mathrm{in}_{\mathrm{in}}} & \mathrm{case}(P,\, x.\mathrm{case}(Q_1,\, y.R_1,\, y.R_2), \\
 & & \quad x.\mathrm{case}(Q_2,\, y.R_1,\, y.R_2)) \\
\mathrm{case}(P,\, x.Q_1,\, x.Q_2)\,t & p_{\mathrm{in}_\Lambda} & \mathrm{case}(P,\, x.Q_1\,t,\, x.Q_2\,t) \\
\mathrm{let}\,[\![y, \alpha]\!] = \mathrm{case}(P,\, x.Q_1,\, x.Q_2)\,\mathrm{in}\,R & p_{\mathrm{in}_{[\![]\!]}} & \mathrm{case}(P,\, x.\mathrm{let}\,[\![y, \alpha]\!] = Q_1\,\mathrm{in}\,R, \\
 & & \quad x.\mathrm{let}\,[\![y, \alpha]\!] = Q_2\,\mathrm{in}\,R)
\end{array} \tag{8}$$

The effect of these rules is that a redex that is "blocked" by a unreduced case-expression. By moving the argument past the construction, we can continue the reductions in each of the two branches as if the case-expression had been reduced. It is useful to note the representation of the rules as trees, as depicted on Figure 4.6. The rules for $\mathrm{p}_{[\![]\!]}$-reductions are similar.

Our rules for permutative reductions are restricted compared to [RS94], such that they only apply to major subterms. Without this restriction we would allow infinite reductions. [RS94] defines an eliminative context as

$$E ::= \cdots \mid \mathrm{case}([\,],\, y.Q,\, y.R) \mid \mathrm{case}(P,\, y.[\,],\, y.R) \mid \mathrm{case}(P,\, y.Q,\, y.[\,])$$

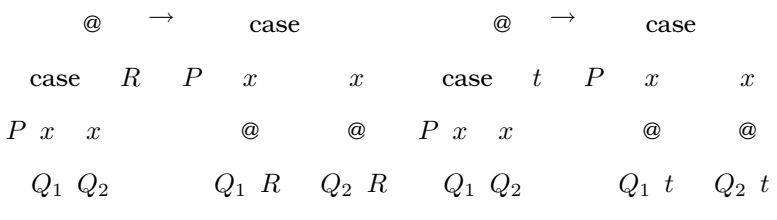
and the permutative reduction (P) by

$$E[\mathrm{case}(L,\, x.M,\, x.N)] \to \mathrm{case}(L,\, x.E[M],\, x.E[N])$$

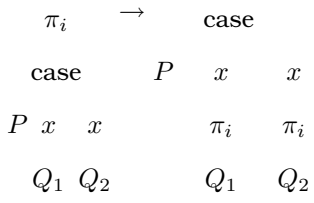
with the restriction that the subterm $\mathrm{case}(L,\, x.M,\, x.N)$ is a normal form wrt. all the other reduction rules. However, this definition allows infinite terms as seen from the term $\mathrm{case}(u,\, x.\mathrm{case}(u,\, x.v,\, x.v),\, x.v)$. This term has the reduction (where we underline the eliminative context)

$$\underline{\mathrm{case}(u,\, x.}\mathrm{case}(u,\, x.v,\, x.v)\underline{,\, x.v)} \to \mathrm{case}(u,\, x.\underline{\mathrm{case}(u,\, x.v,\, x.v)},\, x.\underline{\mathrm{case}(u,\, x.v,\, x.v)})\ . \tag{9}$$

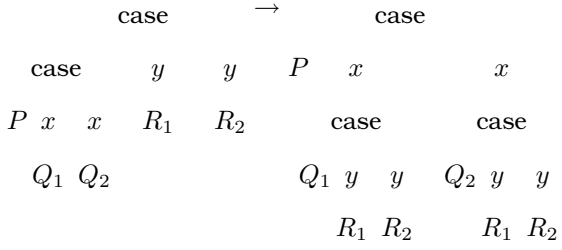
Figure 4.6 Tree representations of the permutative reductions



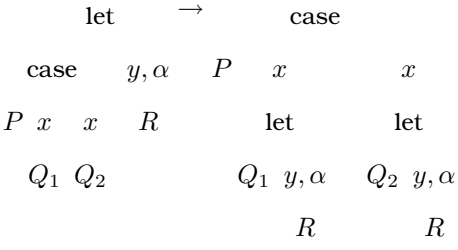
(a) $\mathrm{p_{in}}$ on application (b) $\mathrm{p_{in}}$ on type application



(c) $\mathrm{p_{in}}$ on projection



(d) $\mathrm{p_{in}}$ on case



(e) $\mathrm{p_{in}}$ on packing

In this term we have created two new redexes; one of them is similar to first:

$$\underline{\mathrm{case}(u,\, x.\mathrm{case}(u,\, x.v,\, x.v),\, x.\mathrm{case}(u,\, x.v,\, x.v))} \to \\ \mathrm{case}(u,\, x.\underline{\mathrm{case}(u,\, x.v,\, x.\mathrm{case}(u,\, x.v,\, x.v))},\, x.\underline{\mathrm{case}(u,\, x.v,\, x.\mathrm{case}(u,\, x.v,\, x.v))}) \ . \quad (10)$$

In this term, we have doubled the number of redexes and we can continue in this way. Hence, it is essential to consider only the major subterm if we hope to prove strong normalisation.

4.3.2 On Leaving out the Reductions of Redundant Applications

Let us at this point return to the reason why the rules for reduction of redundant applications of the negation rule, found in Figure 4.5, would require a Church-style calculus. The problem with these rules is that the corresponding reductions depend on the subterms having certain types. To illustrate this, consider the term $M \equiv \lambda x.\lambda y.y\ \Delta z.z\ x$ of type $\tau \to (\tau \to \vartheta) \to \vartheta$ as seen by the type derivation:

$$\dfrac{\dfrac{\Xi \vdash y : \tau \to \vartheta \qquad \dfrac{\dfrac{\Xi' \vdash z : \neg\tau \quad \Xi' \vdash x : \tau}{\Xi' = \Xi, z : \neg\tau \vdash z\ x : \bot}}{\Xi \vdash \Delta z.z\ x : \tau}}{\Xi = x : \tau, y : \tau \to \vartheta \vdash y\ \Delta z.z\ x : \vartheta}}{\dfrac{x : \tau \vdash \lambda y.y\ \Delta z.z\ x : (\tau \to \vartheta) \to \vartheta}{\vdash \lambda x.\lambda y.y\ \Delta z.z\ x : \tau \to (\tau \to \vartheta) \to \vartheta}} \ . \quad (11)$$

Thus, the term encodes a proof for *modus ponens*. Seen as a deduction in classical first-order logic, the deduction cannot be contracted further using Stålmarck's rules of fig 4.2–4.5. The term M should therefore be a normal form under the reduction rules in our λ-calculus. We can now choose $\tau = \vartheta = \bot$ and redo a type deduction for $M : \bot \to (\bot \to \bot) \to \bot$:

$$\dfrac{\dfrac{\Xi \vdash y : \bot \to \bot \qquad \dfrac{\dfrac{\Xi' \vdash z : \neg\bot \quad \Xi' \vdash x : \bot}{\Xi' = \Xi, z : \neg\bot \vdash z\ x : \bot}}{\Xi \vdash \Delta z.z\ x : \bot}}{\Xi = x : \bot, y : \bot \to \bot \vdash y\ \Delta z.z\ x : \bot}}{\dfrac{x : \bot \vdash \lambda y.y\ \Delta z.z\ x : (\bot \to \bot) \to \bot}{\vdash \lambda x.\lambda y.y\ \Delta z.z\ x : \bot \to (\bot \to \bot) \to \bot}} \ .$$

M now represents a tautology: it is immediate that we can conclude $\bot$ when we assume $\bot$. This is reflected in the reduction rules of Stålmarck: using the reductions of redundant applications of the negation rule thrice, we can remove the unnecessary derivation of $\bot$ from $\bot \to \bot$ and $\bot$:

$$\dfrac{\dfrac{x : \bot, y : \bot \to \bot \vdash x : \bot}{x : \bot \vdash \lambda y.x : (\bot \to \bot) \to \bot}}{\vdash \lambda x.\lambda y.x : \bot \to (\bot \to \bot) \to \bot} \ .$$

We see that with the typing $\bot \to (\bot \to \bot) \to \bot$ the term M reduces to $\lambda x.\lambda y.x$ in three steps. As it is only the types of the variables that are different in the two cases, it is necessary to include the type of the variables in the term. If we had done this, it would have been sufficient to annotate the abstraction variables[5] with their types as this determines the type of a term in full.

[5]Here, and in the following, we use abstraction variable in a broad sense referring to all the places where a variable becomes bound, *ie*, in abstractions, in case-expressions, in unpacking, and by Δ-constructors.

We will informally discuss the computational consequences of leaving out the term reductions corresponding to reductions of redundant applications of the negation rule. These reductions reduce a term of type $\bot$ to another term of type $\bot$. As no terms of type $\bot$ can be the operator of a β, p, or Δ-redex, this does not create new β, p, or Δ-redexes. Furthermore, by the same reasoning, a term of type $\bot$ might be copied under β, p, or Δ-redex, but only to a position where it is not the operator of any of these redexes, *ie*, if it is copied into a redex, then it is copied into a minor subterm of the redex. From this perspective the terms of type $\bot$ only work as a token we pass around, and the reductions only change the representation of the token. Hence, leaving out the rules should not have much effect on what we can compute.

However, when considering normalisation, the situation is different. Looking at the rules more specifically, the first rule reads

$$\Delta x^{\bot\to\bot} P^{\bot} \longrightarrow P^{\bot}$$

provided that $x \notin \mathrm{FV}(P)$. (As the calculus should be Church-style, we have annotated the type of terms using superscript.) As discussed above, the left hand side could not be the operator of a Δ-redex. Infinite reductions in the left hand side are conserved as they can only arise in P. However, the second rule

$$x^{\bot\to\bot}\, P^{\bot} \longrightarrow P^{\bot}$$

might change the normalisation properties of the term that the redex is part of: x might, in a later β or Δ-reduction, be substituted by an infinite term. This infinite term will be lost after contraction of the redex above, if the x in the redex is the only free occurrence of x. To conclude, we change the normalisation properties by leaving out these reductions.

To conclude the discussion on the reductions of redundant application of the negation rule, we note that Rehof and Sørensen [RS94] have a reduction rule that is to some extent similar to the first of these reduction: $\Delta x : \neg\tau.x\, M \to M$ with the proviso that $x \notin \mathrm{FV}(M)$. It corresponds to a reduction

$$\dfrac{\dfrac{\dfrac{\Sigma}{\tau} \quad \neg\tau}{\bot}}{\tau} \to \dfrac{\Sigma}{\tau} \ .$$

With this rule, we could simplify (11) to

$$\dfrac{\dfrac{\dfrac{x : \tau, y : \tau \to \vartheta \vdash y : \tau \to \vartheta \quad x : \tau, y : \tau \to \vartheta \vdash x : \tau}{x : \tau, y : \tau \to \vartheta \vdash y\, x : \vartheta}}{x : \tau \vdash \lambda y.y\, x : (\tau \to \vartheta) \to \vartheta}}{\vdash \lambda x.\lambda y.y\, x : \tau \to (\tau \to \vartheta) \to \vartheta} \ .$$

4.3.3 Simpler First-Order Logics

Prawitz [Pra65] discusses natural deduction of two other first-order logics: minimal logic and intuitionistic logic, which we will now discuss shortly. In fact, we mainly restrict ourself to the calculus with a type system corresponding to minimal logic in the following, *ie*, we leave out the Δ-operator. More precisely, *minimal logic* is first-order logic without absurdity. The inference rules are thus the same as in Figure 4.1 with the negation rule $\bot$ removed. This makes the reduction rules in Figure 4.3 to make consequences of the negation rule atomic superfluous. We can therefore remove the Δ-operator and its reductions from our language. We define the corresponding calculus as follows:

Definition 4.10 (Λ_{M})

i. The terms of the calculus Λ_{M} are the $\Lambda_{\mathrm{M}\Delta}$-terms without no subterms of the form $\Delta x P$.

ii. The notions of reduction on Λ_{M} are the notions of reduction β and p of $\Lambda_{\mathrm{M}\Delta}$.

From our previous discussion, we see that this is a straight-forward first-order language with all the usual features

Another possibility is to replace the classical negation rule with a rule for *ex falso sequitur quod libet*:

$$\frac{\bot}{A}\ \bot_I\ .$$

This gives us intuitionistic first-order logic. One can make term assignments for this rule by adding a new constructor ε to our language. This gives us the following typing rule:

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \varepsilon(M) : \phi}\ \bot_I\ .$$

It is then sufficient to introduce the permutative reduction rule [GLT89, p. 78]

$$D_1[\varepsilon(M)] \to \varepsilon(M)\ ,$$

which is not computational interesting: We can never remove a ε-constructor so reduction on ε can only bubble the ε-constructor to a more outermost position in the term. In this process subterms are erased so we get a strictly smaller term. It is therefore strongly normalising in itself. In combination with β-reduction it cannot create new β-redexes, but, as it is erasing, it can change the normalisation properties.

4.4 An Extension of Λ^I

For the sake of the reasoning in the following chapters, we conclude this chapter with the definition of Λ^I_{M}, an extension of Λ^I. In the classical Λ^I all abstraction variables are required to occur free in the abstracted terms. This is sufficient to ensure that β-reduction does not erase subterms. When moving to Λ_{M} we can no longer expect a syntactically condition to suffice, as we have reductions that are erasing by nature. However, the condition that a abstraction variable should occur free in the subterm where it is bound is still useful. It now ensures that reductions built on substitution, *ie*, the β_λ, $\beta_{[]}$, and, partially, β_{in}-reductions, do not erase subterms. The pairing constructions $\langle Q, R\rangle$ and $\mathrm{case}(P,\, x.Q,\, x.R)$ need some extra considerations, as either of the subterms Q or R will be reduced by $\beta_{\langle\rangle}$ and β_{in}-reduction, resp. Therefore, to minimise erasing, it seems natural to ensure that the term substituted in a β_λ, β_{in}, or $\beta_{[]}$-reduction is substituted into the two halves of the pairing construction. We will therefore require that the two parts of a pairing construction have the same free variables.

As we shall see, the technique for inferring strong normalisation from weak normalisation relies on the introduction of some free variables, which are named y in Chapter 7. Each of these variables occur only once in the terms under consideration and can hence not be subject to the condition that the two halves of a pairing construction have the same free variables. We denote the set of all these variables by $\mathbb{Y}$ and exclude this set from our considerations. As we, by the variable convention, can assume that the variables in $\mathbb{Y}$ are not used as abstraction variables, the exclusion of $\mathbb{Y}$ does not change our objective of preventing reductions from erasing subterms.

Definition 4.11 Let some fixed set of variables $\mathbb{Y}$ be given. The subset of terms $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ contains the Λ_{M} terms M satisfying the following conditions:

i. $\lambda x.P \subseteq M \Longrightarrow x \in \mathrm{FV}(P)$

ii. $\langle P, Q\rangle \subseteq M \Longrightarrow \mathrm{FV}(P) \setminus \mathbb{Y} = \mathrm{FV}(Q) \setminus \mathbb{Y}$,

iii. $\mathrm{case}(P,\, x.Q,\, x.R) \subseteq M \Longrightarrow \mathrm{FV}(Q) \setminus \mathbb{Y} = \mathrm{FV}(R) \setminus \mathbb{Y}$ & $x \in \mathrm{FV}(Q)$, and

iv. $\mathrm{let}\,[\![x, \alpha]\!] = P \,\mathrm{in}\, Q \subseteq M \Longrightarrow x \in \mathrm{FV}(Q)$. ♦

We note that in Case iii we have implicitly $x \in \mathrm{FV}(Q)$ too: x is an abstraction variable and therefore by the variable convention $x \notin \mathbb{Y}$. As $\mathrm{FV}(P) \setminus \mathbb{Y} = \mathrm{FV}(Q) \setminus \mathbb{Y}$ and $x \in \mathrm{FV}(P)$ we must also have $x \in \mathrm{FV}(Q)$. We will implicitly consider the set $\mathbb{Y}$ for given in the following. It should be noted that this definition only deals with term variables. As no computations are done in the eigenterms, it is not relevant to consider these.

Having defined $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, we will prove that it is closed under β-reduction. To this end, we need the following lemmata. The first states that free variables are preserved under β-reduction, while the second states that $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ is closed under substitution of $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms.

Lemma 4.12 *For all terms $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ and terms $N \in \Lambda_{\mathrm{M}}$ where $M \to_\beta N$ we have $\mathrm{FV}(M) \setminus \mathbb{Y} = \mathrm{FV}(N) \setminus \mathbb{Y}$.*

Proof. We shall prove that

$$M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}} \;\&\; M \to_\beta N \Longrightarrow \big(\mathrm{FV}(N) \setminus \mathbb{Y} = \mathrm{FV}(M) \setminus \mathbb{Y}\big)\ , \tag{$*$}$$

using induction on the structure of M:

i. $M \equiv x$. There is no N such that $x \to_\beta N$ and therefore ($*$) is trivially true.

ii. $M \equiv \lambda x.P$. We have $M \equiv \lambda x.P \to_\beta \lambda x.Q \equiv N$ for some Q where $P \to_\beta Q$. By the induction hypothesis, we conclude that

$$\mathrm{FV}(N) \setminus \mathbb{Y} = \mathrm{FV}(\lambda x.Q) \setminus \mathbb{Y} = \mathrm{FV}(Q) \setminus \{x\} \setminus \mathbb{Y} \overset{\text{IH}}{=} \mathrm{FV}(P) \setminus \{x\} \setminus \mathbb{Y} = \mathrm{FV}(M) \setminus \mathbb{Y}\ .$$

iii. $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. The cases are similar only varying with the number of subcases we split into. We take $M \equiv \langle P, Q\rangle$ as an example. Either $N \equiv \langle R, Q\rangle$ where $P \to_\beta R$ or $N \equiv \langle P, R\rangle$ where $Q \to_\beta R$. In the former case we have

$$\mathrm{FV}(N) \setminus \mathbb{Y} = \mathrm{FV}(\langle R, Q\rangle) \setminus \mathbb{Y} = \big(\mathrm{FV}(R) \setminus \mathbb{Y}\big) \cup \big(\mathrm{FV}(Q) \setminus \mathbb{Y}\big) \overset{\text{IH}}{=} \big(\mathrm{FV}(P) \setminus \mathbb{Y}\big) \cup \big(\mathrm{FV}(Q) \setminus \mathbb{Y}\big) = \mathrm{FV}(M) \setminus \mathbb{Y}\ .$$

The case where $N \equiv \langle P, R\rangle$ is similar.

iv. $M \equiv P\,Q$. We consider the following subcases where we split on the form of N.

(a) $N \equiv P'\{x := Q\}$ and $P \equiv \lambda x.P'$, *ie*, M is the β-redex being reduced. As $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, we have from Definition 4.11 that $x \in \mathrm{FV}(P)$ and therefore:

$$\begin{aligned}\mathrm{FV}(N) \setminus \mathbb{Y} &= \mathrm{FV}(P'\{x := Q\}) \setminus \mathbb{Y} \\ &\overset{1.11}{=} \big((\mathrm{FV}(P') \setminus \{x\}) \cup \mathrm{FV}(Q)\big) \setminus \mathbb{Y} \\ &= \big(\mathrm{FV}(\lambda x.P') \cup \mathrm{FV}(Q)\big) \setminus \mathbb{Y} \\ &= \mathrm{FV}(M) \setminus \mathbb{Y}\ .\end{aligned}$$

(b) $N \equiv R\,Q$ or $N \equiv P\,R$ where $P \to_\beta R$ or $Q \to_\beta R$, resp. In these cases the reduction is done in a subterm and the cases are similar to Case iii above.

v. $M \equiv \pi_i(P)$. We split into subcases depending on the form of N:

(a) $N \equiv P_i$ where $P \equiv \langle P_1, P_2\rangle$, *ie*, M is the β-redex being reduced. As we have $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, we know from Definition 4.11 that $\mathrm{FV}(P_1) \setminus \mathbb{Y} = \mathrm{FV}(P_2) \setminus \mathbb{Y}$ and therefore

$$\mathrm{FV}(N) \setminus \mathbb{Y} = \mathrm{FV}(P_i) \setminus \mathbb{Y} = \big(\mathrm{FV}(P_1) \cup \mathrm{FV}(P_2)\big) \setminus \mathbb{Y} = \mathrm{FV}(M) \setminus \mathbb{Y}\ .$$

(b) $N \equiv \pi_i(Q)$ where $P \to_\beta Q$, *ie*, the reduction is done under the projection. The case is similar to Case iii above.

vi. $M \equiv \mathrm{case}(P,\, x.Q_1,\, x.Q_2)$. We divide into subcases depending on the form of N:

(a) $N \equiv Q_i\{x := P'\}$ where $P \equiv \mathrm{in}_i(P')$. It follows from $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ and Definition 4.11 that $x \in \mathrm{FV}(Q_i)$ and $\mathrm{FV}(Q_1) \setminus \mathbb{Y} = \mathrm{FV}(Q_2) \setminus \mathbb{Y}$. We conclude

$$\begin{aligned}\mathrm{FV}(N) \setminus \mathbb{Y} &= \big(\mathrm{FV}(Q_i\{x := P\})\big) \setminus \mathbb{Y}\\ &\overset{1.11}{=} \big((\mathrm{FV}(Q_i) \setminus \{x\}) \setminus \mathbb{Y}\big) \cup \big(\mathrm{FV}(P) \setminus \mathbb{Y}\big)\\ &= \big(((\mathrm{FV}(Q_1) \cup \mathrm{FV}(Q_2)) \setminus \{x\}) \setminus \mathbb{Y}\big) \cup (\mathrm{FV}(P) \setminus \mathbb{Y})\\ &= \mathrm{FV}(M) \setminus \mathbb{Y}\ .\end{aligned}$$

(b) $N \equiv \mathrm{case}(R,\, x.Q_1,\, x.Q_2)$ where $P \to_\beta R$, or $N \equiv \mathrm{case}(P,\, x.R,\, x.Q_2)$ with $Q_1 \to_\beta R$, or $N \equiv \mathrm{case}(P,\, x.Q_1,\, x.R)$ where $Q_2 \to_\beta R$. In all three cases the reduction is done in the subcase and they are similar to Case iii above.

vii. $M \equiv P\,t$. We consider the following two subcases:

(a) $N = P'\{\alpha := t\}$ where $P \equiv \Lambda\alpha.P'$, *ie*, M is the β_Λ-redex. In this case we have

$$\mathrm{FV}(N) \setminus \mathbb{Y} = \big(\mathrm{FV}(P\{\alpha := t\})\big) \setminus \mathbb{Y} = \mathrm{FV}(P) \setminus \mathbb{Y} = \mathrm{FV}(M) \setminus \mathbb{Y}\ .$$

(b) $N \equiv Q\,t$ where $P \to_\beta Q$, *ie*, the reduction is done in a subterm. The case is similar to Case iii above.

viii. $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$. In this case we split into subcases depending on the form of N:

(a) $N \equiv P'\{x := Q, \alpha := t\}$ where $P \equiv [\![P', t]\!]$, *ie*, M is the reduced β-redex. Definition 4.11 states that $x \in \mathrm{FV}(Q)$ since $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$. We have

$$\begin{aligned}\mathrm{FV}(N) \setminus \mathbb{Y} &= \big(\mathrm{FV}(P'\{x := Q, \alpha := t\})\big) \setminus \mathbb{Y}\\ &\overset{1.11}{=} \big(\mathrm{FV}(P) \cup (\mathrm{FV}(Q) \setminus \{x\})\big) \setminus \mathbb{Y}\\ &= \mathrm{FV}(M) \setminus \mathbb{Y}\ .\end{aligned}$$

(b) $N \equiv \mathrm{let}\,[\![x, \alpha]\!] = R\,\mathrm{in}\,Q$ or $N \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,R$ where $P \to_\beta R$ or $Q \to_\beta R$, resp. In this case the reduction is done in a subterm and the situation is similar to Case iii above.

This exhausts the possible forms of M and completes the proof that the free variables in a $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term are preserved under reduction. □

Lemma 4.13 *Let $M, N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ be $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms and t be a eigenterm. Then*

i. $M\{x := N\} \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ *and*

ii. $M\{\alpha := t\} \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$

where x is any term variable and α is any type variable.

Proof. If $x \notin \mathrm{FV}(M)$ then the lemma is trivially true as $M\{x := N\} \equiv M$, which is an $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term by assumption. We hence assume that $x \in \mathrm{FV}(M)$ and consider the different forms of the subterms $S \subseteq M\{x := N\}$ of Definition 4.11. If $S \subseteq M$ or $S \subseteq N$, then S fulfils the conditions by assumption. We therefore consider the situation where $S \not\subseteq M$ and $S \not\subseteq N$:

i. $S \equiv \lambda y.P$. By the substitution generation lemma we have a subterm $\lambda y.P' \subseteq M$ such that $(\lambda y.P')\{x := N\} \equiv \lambda y.P$. Using the variable convention, we have $x \neq y$, and from this we conclude

$$y \in \big(\mathrm{FV}(P') \setminus \{x\}\big) \cup \mathrm{FV}(N) \overset{1.11}{=} \mathrm{FV}(P) \ .$$

ii. $S \equiv \langle P, Q\rangle$. In this case, we use the substitution generation lemma to find a subterm $\langle P', Q'\rangle \subseteq M$ such that $\langle P', Q'\rangle\{x := N\} \equiv \langle P, Q\rangle$. From the assumption that $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, we know that $\mathrm{FV}(P') \setminus \mathbb{Y} = \mathrm{FV}(Q') \setminus \mathbb{Y}$. Using Lemma 1.11 twice we conclude:

$$\begin{aligned}
\mathrm{FV}(P) \setminus \mathbb{Y} &= \mathrm{FV}(P'\{x := N\}) \setminus \mathbb{Y} \\
&\overset{1.11}{=} \big((\mathrm{FV}(P') \setminus \{x\}) \cup \mathrm{FV}(N)\big) \setminus \mathbb{Y} \\
&= \big((\mathrm{FV}(Q') \setminus \{x\}) \cup \mathrm{FV}(N)\big) \setminus \mathbb{Y} \\
&\overset{1.11}{=} \mathrm{FV}(Q'\{x := N\}) \setminus \mathbb{Y} \\
&= \mathrm{FV}(Q) \setminus \mathbb{Y}
\end{aligned}$$

iii. $S \equiv \mathrm{case}(P,\, y.Q,\, y.R)$. We have a subterm $\mathrm{case}(P',\, y.Q',\, y.R') \subseteq M$ by the substitution generation lemma. By the lemma, this subterm satisfies the condition that $\mathrm{case}(P',\, y.Q',\, y.R')\{x := N\} \equiv \mathrm{case}(P,\, y.Q,\, y.R)$. Using the same reasoning as in Case i and Case ii we conclude that $y \in \mathrm{FV}(Q)$ and that $\mathrm{FV}(Q)\setminus\mathbb{Y} = \mathrm{FV}(R)\setminus\mathbb{Y}$.

iv. $S \equiv \mathrm{let}\,[\![y, \alpha]\!] = P \,\mathrm{in}\, Q$. We find a subterm $\mathrm{let}\,[\![y, \alpha]\!] = P' \,\mathrm{in}\, Q' \subseteq M$ using the substitution generation lemma. From the lemma, we know the subterm is subject to the condition $\mathrm{let}\,[\![y, \alpha]\!] = P' \,\mathrm{in}\, Q'\{x := N\} \equiv \mathrm{let}\,[\![y, \alpha]\!] = P \,\mathrm{in}\, Q$. By the same reasoning as in Case i we conclude that $x \in \mathrm{FV}(Q)$.

This exhausts the possible forms of S and concludes the proof that $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ is closed under term substitution. As the substitution generation lemma also holds for type substitution, the proof in the case of type substitution is similar with α substituted for x and t for N. The proof gets somewhat simpler, as we do not have to deal with the exclusion of the set $\mathbb{Y}$. □

Now follows the proposition stating that $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ is closed under β-reduction.

Proposition 4.14 *Let M and N be Λ_{M}-terms with $M \to_\beta N$. If M is a $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term, then N is $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term.*

Proof. We shall prove that

$$M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}} \,\&\, M \to_\beta N \Longrightarrow N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}} \quad . \tag{$*$}$$

To ease the reasoning we say that a term M is *I-friendly* if, and only if, one of the following holds:

i. $M \equiv \lambda x.P \Longrightarrow x \in \mathrm{FV}(P)$,

ii. $M \equiv \langle P, Q\rangle \Longrightarrow \mathrm{FV}(P) \setminus \mathbb{Y} = \mathrm{FV}(Q) \setminus \mathbb{Y}$,

iii. $M \equiv \mathrm{case}(P,\, x.Q,\, x.R) \Longrightarrow \mathrm{FV}(Q) \setminus \mathbb{Y} = \mathrm{FV}(R) \setminus \mathbb{Y} \,\&\, x \in \mathrm{FV}(Q)$, and

iv. $M \equiv \mathrm{let}\, [\![x, \alpha]\!] = P \,\mathrm{in}\, Q \Longrightarrow x \in \mathrm{FV}(Q)$,

I-friendly restates the conditions in Definition 4.11 on subterms of $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms. The advantage of I-friendliness is that it is defined for all terms. It should be clear that a term M is a $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term if, and only if, all its subterms are I-friendly.

We proceed by induction on the structure of M:

i. $M \equiv x$. There is no N such that $x \to_\beta N$ and therefore ($*$) is trivially true.

ii. $M \equiv \lambda x.P$. In this case the reduction is done in a subterm, *ie*, $N \equiv \lambda x.Q$ for some Q with $P \to_\beta Q$. From the induction hypothesis we have that $Q \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, and we hence know that all subterms $S \subseteq Q$ are I-friendly. Furthermore, it follows from Lemma 4.12 that $x \in \mathrm{FV}(Q) \setminus \mathbb{Y} = \mathrm{FV}(P) \setminus \mathbb{Y}$ and therefore that $\lambda x.Q$ is I-friendly. We conclude that N is a $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term.

iii. $M \equiv \langle P, Q\rangle$. The reduction is done in a subterm and we have $N \equiv \langle R, Q\rangle$ or $N \equiv \langle P, R\rangle$ where $P \to_\beta R$ or $Q \to_\beta R$, resp. We consider the former case. It follows from the induction hypothesis that $R \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$. We consider any subterm $S \subseteq N$ and should prove that it is I-friendly. We have two possibilities:

   (a) $S \equiv \langle R, Q\rangle$. By Lemma 4.12 we know that $\mathrm{FV}(R) \setminus \mathbb{Y} = \mathrm{FV}(Q) \setminus \mathbb{Y} = \mathrm{FV}(P) \setminus \mathbb{Y}$ where the last equation follows from $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

   (b) $S \subseteq R$ or $S \subseteq Q$. As S is a subterm of a $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term, it is I-friendly.

   We conclude that all subterms of N are I-friendly and therefore that N is a $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term. The case where $N \equiv \langle P, R\rangle$ is similar.

iv. $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. In these three cases the reduction is done in a subterm, *ie*, $N \equiv \mathrm{in}_i(Q)$, or $N \equiv \Lambda\alpha.Q$, or $N \equiv [\![Q, t]\!]$ where $P \to_\beta Q$. Using the induction hypothesis we find that $Q \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ and therefore that any subterm of Q is I-friendly. Furthermore, as the term N is I-friendly, we conclude that $N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

v. $M \equiv P\,Q$. We do the following case split on N:

   (a) $N \equiv P'\{x := Q\}$ and $P \equiv \lambda x.P'$, *ie* M is the reduced β-redex. As P' and Q are $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms, we conclude from Lemma 4.13 that $N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

   (b) $N \equiv R\,Q$ or $N \equiv P\,R$ where $P \to_\beta R$ or $Q \to_\beta R$, resp. In these cases the reduction is done in a subterm and the cases are similar to Case iii above.

vi. $M \equiv \pi_i(P)$. We split into subcases depending on the form of N:

   (a) $N \equiv P_i$. As $P_i \subseteq M$, we have trivially that $N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

(b) $N \equiv \pi_i(Q)$ where $P \to_\beta Q$, *ie*, the reduction is done under the projection. The case is similar to Case iii above.

vii. $M \equiv \mathrm{case}(P, x.Q_1, x.Q_2)$. We divide into subcases depending on the form of N:

(a) $N \equiv Q_i\{x := P'\}$ where $P \equiv \mathrm{in}_i(P')$. As $P' \subseteq M$ and $Q_i \subseteq M$ we know that the two terms are $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms. Using Lemma 4.13 we conclude that $N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

(b) $N \equiv \mathrm{case}(R, x.Q_1, x.Q_2)$ where $P \to_\beta R$, or $N \equiv \mathrm{case}(P, x.R, x.Q_2)$ with $Q_1 \to_\beta R$, or $N \equiv \mathrm{case}(P, x.Q_1, x.R)$ with $Q_2 \to_\beta R$. In all three cases the reduction is done in a subterm and they are similar to Case iii above.

viii. $M \equiv P\,t$. We consider the following two subcases:

(a) $N = P'\{\alpha := t\}$ where $P \equiv \Lambda\alpha.P'$, *ie*, M is the reduced β-redex. As $P' \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ it follows from Lemma 4.13 that $N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

(b) $N \equiv Q\,t$ where $P \to_\beta Q$, *ie*, the reduction is done in a subterm. The case is similar to Case iii above.

ix. $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$. In this case we split into subcases depending on the form of N:

(a) $N \equiv Q\{x := P', \alpha := t\}$ where $P \equiv [\![P', t]\!]$, *ie*, M is the $\beta_{[\!]}$-redex. From a double application of Lemma 4.13 we conclude that $N \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

(b) $N \equiv \mathrm{let}\,[\![x, \alpha]\!] = R\,\mathrm{in}\,Q$ or $N \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,R$ where $P \to_\beta R$ or $Q \to_\beta R$, resp. In this case the reduction is done in a subterm and the situation is similar to Case iii above.

This exhausts the possible forms of M and completes the proof that $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ is closed under β-reduction.

□

4.5 Conclusion

This chapter has brought us far about. We started out discussing the different forms of first-order logic in the first half of the chapter. We then found the typed λ-calculus $\Lambda_{\mathrm{M}\Delta}$ corresponding to classical first-order logic in the middle part the chapter. As we saw, this calculus has several notions of reductions: β-reduction forms the basic notion of reduction and is intended for the main part of the reduction. Furthermore, we have the permutative reduction and the Δ-reductions with a similar structure. The permutative reductions are somewhat administrative as they ensure that β-redexes are not blocked. The Δ-reductions are more complicated and, as discussed in [RS94], they correspond to exception handling. We restricted $\Lambda_{\mathrm{M}\Delta}$ to the calculus Λ_{M} with a type system corresponding to a minimal logic. In Λ_{M} we do not have the Δ-constructor and therefore we also leave out the Δ-reductions from Λ_{M}.

Our motivation for introducing $\Lambda_{\mathrm{M}\Delta}$ and Λ_{M} is to infer strong normalisation from weak normalisation. We do this in Chapter 7 where we consider Λ_{M} and prove that WN_β implies $\mathrm{SN}_{\beta\mathrm{p}}$. To do this, we need some more machinery of the calculus Λ_{M}. In the following chapter, we establish that leftmost reduction is normalising. In Chapter 6 we use this to find a uniformly normalising subset along the same lines as in Chapter 3.

Chapter 5

Leftmost Reduction—A Normalising Reduction

It is well-known [Bar84] that leftmost reduction in Λ^K is normalising. As a normalising reduction strategy is central in our proof of uniform normalisation, we will in this chapter extend the notion of leftmost reduction to Λ_{M}. The original proof that leftmost reduction is normalising is given by Curry and Feys [CF58]. They prove it using standard reductions, which are reductions where the redexes are reduced from left to right. They establish that if M reduces to N, then there is a standard reduction from M to N. From this it is easy to conclude that leftmost reduction is normalising, see [Bar84, 13.2.2]. Their proof depends on developments: Labeled redexes (as in Chapter 2) are introduced. A development is a reduction path where only labelled redexes are reduced. The full proof is rather complicated and depends on a lot of auxiliary notation.

We will therefore use an alternative technique due to Takahashi [Tak95] with inspiration from Martin-Löf and Tait. The fundamental notion in this proof is parallel reduction, which is simultaneous reduction of several, possibly overlapping, β-redexes. In the first section we define the leftmost reduction strategy F_l. In the second section we introduce the notation used in the proof that it is normalising and do the proof.

5.1 The Leftmost Reduction Strategy

We define the leftmost reduction strategy F_l in this section. We start out with a definition of a context used to identify the leftmost redex.

Definition 5.1 The set of *leftmost evaluation contexts* $\mathcal{C}_{\mathrm{lm}}$ in Λ_{M} is defined by the following grammar

$$\mathcal{C}_{\mathrm{lm}} \ni C ::= [] \mid C\ M \mid \pi_i(C) \mid \mathrm{case}(C,\, x.M,\, x.M) \mid C\ t \mid \mathrm{let}\ \langle\!\langle x, \alpha \rangle\!\rangle = C\ \mathrm{in}\ M$$

where M is any Λ_{M}-term.

We note that the leftmost evaluation context could be considered the generalisation of the one-level destructor contexts as $\mathcal{C}_{\mathrm{lm}} \equiv D_1^1[D_1^2[\ldots D_1^n[\ldots]]]$ for some $n \geq 1$ and one-level destructor contexts $D_1^1, \ldots, D_1^n$. The hole $\mathcal{C}_{\mathrm{lm}}$ is found in D_1^n.

We start out by doing the simple, but useful observation that for any leftmost evaluation context C, the filled context $C[x]$ will never reduce to a constructor. This insight is important for the understanding the rôle of leftmost evaluation contexts to pick redexes.

Lemma 5.2 *For all leftmost evaluation contexts $C \in \mathcal{C}_{\mathrm{lm}}$ and all term variables x, there is no reduction path such that*

i. $C[x] \twoheadrightarrow_\beta \lambda z.S$,

ii. $C[x] \twoheadrightarrow_\beta \langle S, T\rangle$,

iii. $C[x] \twoheadrightarrow_\beta \mathrm{in}_i(S)$,

iv. $C[x] \twoheadrightarrow_\beta \Lambda\alpha.S$, *or*

v. $C[x] \twoheadrightarrow_\beta [\![S, t]\!]$.

Proof. We use induction on the structure of C:

i. $C = []$. In this case $C[x] = x \in \mathrm{NF}_\beta$.

ii. $C \equiv C'\,Q$, or $C \equiv \pi_i(C')$, or $C \equiv \mathrm{case}(C', x.Q, x.R)$, or $C \equiv \mathrm{let}\,[\![x, \alpha]\!] = C'\,\mathrm{in}\,Q$, or $C \equiv C'\,t$. The cases are similar and we take $C \equiv C'\,P$ as illustration. We consider each of the five forms mentioned in the lemma and use *reductio ad absurdum*.

   Suppose we have a reduction path $C'[x]\,P \twoheadrightarrow_\beta \lambda z.S$. It must have the form

$$C'[x]\,Q \twoheadrightarrow_\beta (\lambda u.P)\,Q' \to_\beta P\{u := Q'\} \twoheadrightarrow_\beta \lambda z.S$$

   where $C'[x] \twoheadrightarrow_\beta \lambda u.P$, but this contradicts the induction hypothesis. Similar arguments apply to the other reductions mentioned in the lemma.

We have thus proved that $C[x]$ never reduces to a constructor. □

Next, we prove that all Λ_{M}-terms have a unique *decomposition* into a leftmost evaluation context and a term.

Proposition 5.3 (Decomposition of a term) *Let $M \in \Lambda_{\mathrm{M}}$ be a typable term. There is a unique decomposition of the term into a context $C \in \mathcal{C}_{\mathrm{lm}}$ and a term $R \in \Lambda_{\mathrm{M}}$ such that $C[R] \equiv M$ and exactly one of the following holds*

i. $R \equiv x$

ii. $R \equiv \lambda x.P$ *and* $C = []$

iii. $R \equiv \langle P, Q\rangle$ *and* $C = []$

iv. $R \equiv \mathrm{in}_i(P)$ *and* $C = []$

v. $R \equiv \Lambda\alpha.P$ *and* $C = []$

vi. $R \equiv [\![P, t]\!]$ *and* $C = []$

vii. $R \equiv (\lambda x.P)\,Q$

viii. $R \equiv \pi_i(\langle P, Q\rangle)$

ix. $R \equiv \mathrm{case}(\mathrm{in}_i(P), x.Q, x.Q')$

x. $R \equiv (\Lambda\alpha.P)\,t$

xi. $R \equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![P, t]\!]\,\mathrm{in}\,Q$ ♦

Intuitively, the proposition states that a term has one of the following three forms:

i. It has an outermost constructor (Case ii–vi).

ii. It has an outermost destructor that will never be reduced (Case i—Lemma 5.2 shows that the major subterm of the outermost destructor never reduces to a constructor).

iii. It has an outermost destructor that might might be reduced later.

Proof (Proposition 5.3). The proof is in two parts. In the first part we prove that all terms can be decomposed, and in the second part we prove that a decomposition is unique. For the first part we use induction on the structure of M:

i. $M \equiv x$. We choose $C = []$ and $R \equiv x$ as our decomposition; this is the decomposition in Case i.

ii. $M \equiv \lambda x.P$. We choose $C = []$ and $R \equiv \lambda x.P$, which fulfils the conditions in Case ii.

iii. $M \equiv \langle P, Q\rangle$. We choose the decomposition $C = []$ and $R \equiv \langle P, Q\rangle$; this fulfils the conditions in Case iii.

iv. $M \equiv \mathrm{in}_i(P)$. We choose $C = []$ and $R \equiv \mathrm{in}_i(P)$ fulfilling the conditions in Case iv.

v. $M \equiv \Lambda\alpha.P$. We choose $C = []$ and $R \equiv \Lambda\alpha.P$; this is is the conditions of Case v.

vi. $M \equiv [\![P, t]\!]$. We choose the decomposition $C = []$ and $R \equiv [\![P, t]\!]$ that satisfies the conditions in Case vi.

vii. $M \equiv P\,Q$, or $M \equiv \pi_i(P)$, or $M \equiv \mathrm{case}(P,\, x.Q,\, x.Q')$, or $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$, or $M \equiv P\,t$. The cases are very similar and we choose $M \equiv P\,Q$ as illustration. We split into subcases on the structure of P:

   (a) $P \equiv \lambda y.S$. We choose $C = []$ and $R \equiv (\lambda y.S)\,Q$ as the decomposition; this fulfils Case vii.

   (b) $P \equiv \langle S, T\rangle$, $P \equiv \mathrm{in}_i(S)$, or $P \equiv \Lambda\alpha.P$, or $P \equiv [\![P, t]\!]$. All these cases are impossible as P is not of type $\tau \to \sigma$. This makes M untypable as the major part of an application should have type $\tau \to \sigma$.

   (c) $P \equiv x$, or $P \equiv S\,T$, or $P \equiv \pi_i(P)$, or $P \equiv \mathrm{case}(S,\, y.T,\, y.U)$, or $P \equiv S\,t$, or $P \equiv \mathrm{let}\,[\![x, \alpha]\!] = S\,\mathrm{in}\,T$. All these cases are similar and we choose $P \equiv S\,T$ as an example. We first note that P is typable as it is a subterm of the typable term M. By the induction hypothesis, we can find a leftmost evaluation context C' and a term R' such that $P \equiv C'[R']$. The Cases ii–vi can not hold for $C'[R]$ as we would then have $P \equiv R$ and P is not on any of the forms in the Cases ii–vi. Hence, the context C' and the term R' satisfy the conditions in Case i or one of the Cases vii–xi of the proposition. We can therefore choose $C \equiv C'\,Q$ and $R \equiv R'$ and fulfil the conditions in Case i or one of the Cases vii–xi of the proposition.

   This exhausts the possible forms of P.

   The other cases of M are handled similarly. The only difference is that we should have another constructor P in Case vii(a) for the typable constructor.

This exhausts the possibilities and concludes the proof that all typable Λ_{M}-terms can be decomposed into one of the forms i–xi in Proposition 5.3.

To prove the uniqueness of the decomposition, we consider a term M with two decompositions, *ie*, $M \equiv C[S]$ and $M \equiv D[T]$, both satisfying one of the conditions in Proposition 5.3. We shall prove that $C \equiv D$ and $S \equiv T$; we use induction on the structure of the leftmost evaluation context C.

i. $C = []$. We split into cases on the structure of D:

   (a) $D = []$: We have $C = D$ and also

   $$S = C[S] \equiv M \equiv D[T] = T \ .$$

   (b) $D \equiv D'\,Q$, or $D \equiv \pi_i(D')$, or $D \equiv D'\,t$, or $D \equiv \mathrm{let}\,\langle\!|x, \alpha|\!\rangle = D'\,\mathrm{in}\,Q$, or $D \equiv \mathrm{case}(D', x.Q, x.R)$. All these cases are similar so we use the case where $D \equiv \mathrm{case}(D', x.Q, x.R)$ as an example and proceed using *reductio ad absurdum*. We have

   $$S = C[S] \equiv M \equiv D[T] \equiv \mathrm{case}(D'[T], x.Q, x.R) \ .$$

   This can only be achieved if the context C and the term S satisfies the conditions in Case ix of the proposition, *ie*, $S \equiv \mathrm{case}(\mathrm{in}_i(P), x.Q, x.R)$. From this we conclude that $D'[T] \equiv \mathrm{in}_i(P)$. This is only possible if D' and T fulfil the conditions in Case iv of the proposition, *ie*, $D' = []$ and $T \equiv \mathrm{in}_i(P)$. But the context $D \equiv \mathrm{case}([], x.Q, x.R)$ and the term $T \equiv \mathrm{in}_i(P)$ do not satisfy any of the forms in Case i–xi of the proposition. This contradicts $D[T]$ being a decomposition of M, and we conclude that $D \equiv \mathrm{case}(D', x.Q, x.R)$ is not a possible form of D.

   The same reasoning applies for other forms of D; only the form of S and T differs.

   This exhausts the possible forms of the leftmost evaluation context D.

ii. $C \equiv C'\,Q$, or $C \equiv \pi_i(C')$, or $C \equiv \mathrm{case}(C', y.Q, y.R)$, or $C \equiv \mathrm{let}\,\langle\!|x, \alpha|\!\rangle = C'\,\mathrm{in}\,Q$, or $C \equiv C'\,t$. These cases are all very similar; we illustrate their proof using the case $C \equiv \mathrm{case}(C', y.Q, y.R)$ as an illustration. We split on the possible forms of D:

   (a) $D = []$. This is Case i(b) above with C and D interchanged.

   (b) $D \equiv \mathrm{case}(D', y.Q', y.R')$. Without loss of generality, we can assume that $x = y$. We have

   $$\mathrm{case}(C'[S], x.Q, x.R) \equiv C[S] \equiv M \equiv D[T] \equiv \mathrm{case}(D'[T], x.Q', x.R') \ .$$

   We see from this that we must have $Q \equiv Q'$, $R \equiv R'$, and $C'[S] \equiv D'[T]$. As the decomposition $C[S]$ fulfils one of the conditions in Case i or one of the Cases vii–xi of the proposition, the context C' and the term S fulfil the same condition. As C' is furthermore a part of C, we can apply the induction hypothesis to conclude that the context C' and the term S are the unique decomposition. Hence, $C' \equiv D'$ and $S \equiv T$.

   (c) $D \equiv D'\,Q$, or $D \equiv \pi_i(D')$, or $D \equiv D'\,t$, or $D \equiv \mathrm{let}\,\langle\!|x, \alpha|\!\rangle = D'\,\mathrm{in}\,Q$. All these cases are impossible as we cannot have a different outermost destructor in C and D as $C[S] \equiv M \equiv D[T]$.

   This exhausts the possible forms of the context D.

   The cases for the other forms of C are similar except for another leftmost context D in Case ii(b).

This exhausts the possibilities and we conclude that the decomposition is unique. □

As noted, we left out untypable cases like $C[\mathrm{let}\,\langle\!|x, \alpha|\!\rangle = \lambda x.P\,\mathrm{in}\,Q]$ and $C[\mathrm{in}_i(P)t]$. It should be noted that the decomposition could easily be extended to all the terms in Λ_{M}, but this would only serve to lengthen the proof above. As the decomposition is unique and applies to all typable terms, we can use it for function definitions and to divide into cases in proofs (as long as we only consider typable terms).

Definition 5.4 For any typable term, $M \in \Lambda_{\mathrm{M}}$ the *leftmost reduction* strategy F_l is defined as

$$
\begin{array}{lcl}
F_l(x) & = & x \\
F_l(C[x]\ P) & = & \begin{cases} F_l(C[x])\ P & \text{if } C[x] \notin \mathrm{NF}_\beta \\ C[x]\ F_l(P) & \text{otherwise} \end{cases} \\
F_l(\pi_i(C[x])) & = & \pi_i(F_l(C[x])) \\
F_l(\mathrm{case}(C[x],\ y.P,\ y.Q)) & = & \begin{cases} \mathrm{case}(F_l(C[x]),\ y.P,\ y.Q) \\ \qquad \text{if } C[x] \notin \mathrm{NF}_\beta \\ \mathrm{case}(C[x],\ y.F_l(P),\ y.Q) \\ \qquad \text{if } C[x] \in \mathrm{NF}_\beta \text{ and } P \notin \mathrm{NF}_\beta \\ \mathrm{case}(C[x],\ y.P,\ y.F_l(Q)) \\ \qquad \text{otherwise} \end{cases} \\
F_l(C[x]\ t) & = & F_l(C[x])\ t \\
F_l(\mathrm{let}\ \llbracket y, \alpha \rrbracket = C[x]\ \mathrm{in}\ P) & = & \begin{cases} \mathrm{let}\ \llbracket y, \alpha \rrbracket = F_l(C[x])\ \mathrm{in}\ P & \text{if } C[x] \notin \mathrm{NF}_\beta \\ \mathrm{let}\ \llbracket y, \alpha \rrbracket = C[x]\ \mathrm{in}\ F_l(P) & \text{if } C[x] \in \mathrm{NF}_\beta \end{cases} \\
F_l(\lambda x.P) & = & \lambda x.F_l(P) \\
F_l(\langle P, Q \rangle) & = & \begin{cases} \langle F_l(P), Q \rangle & \text{if } P \notin \mathrm{NF}_\beta \\ \langle P, F_l(Q) \rangle & \text{otherwise} \end{cases} \\
F_l(\mathrm{in}_i(P)) & = & \mathrm{in}_i(F_l(P)) \\
F_l(\Lambda\alpha.P) & = & \Lambda\alpha.F_l(P) \\
F_l(\llbracket P, t \rrbracket) & = & \llbracket F_l(P), t \rrbracket \\
F_l(C[(\lambda x.P)\ Q]) & = & C[P\{x := Q\}] \\
F_l(C[\pi_i(\langle P_1, P_2 \rangle)]) & = & C[P_i] \\
F_l(C[\mathrm{case}(\mathrm{in}_i(P),\ x.Q_1,\ x.Q_2)]) & = & C[Q_i\{x := P\}] \\
F_l(C[(\Lambda\alpha.P)\ t]) & = & C[P\{\alpha := t\}] \\
F_l(C[\mathrm{let}\ \llbracket x, \alpha \rrbracket = \llbracket P, t \rrbracket\ \mathrm{in}\ Q]) & = & C[Q\{\alpha := t, x := P\}]
\end{array}
$$

where $C \in \mathcal{C}_{\mathrm{lm}}$ is a leftmost evaluation context. ⧫

As the definition follows the decomposition in Proposition 5.3 all the entries are non-overlapping. The following proves that it is correct to call F_l a reduction strategy.

Lemma 5.5 *The mapping F_l is a reduction strategy.*

Proof. We should prove that

$$
\begin{array}{c}
M \in \mathrm{NF}_\beta \Longrightarrow F_l(M) = M \\
M \notin \mathrm{NF}_\beta \Longrightarrow M \to_\beta F_l(M)
\end{array}
\qquad (*)
$$

We use induction on the structure of M splitting according to its decomposition.

i. $M \equiv C[x]$. We split into cases according to the structure of C:

   (a) $C \equiv []$. Then $M \equiv x \in \mathrm{NF}_\beta$ and $F_l(M) \equiv x$ so $(*)$ holds.

   (b) $C \equiv C'\ P$, or $C \equiv \pi_i(C')$, or $C \equiv C'\ t$, or $C \equiv \mathrm{let}\ \llbracket x, \alpha \rrbracket = C'\ \mathrm{in}\ P$, or $C \equiv \mathrm{case}(C',\ x.P,\ x.Q)$. The cases are very similar and we use $C \equiv C'\ P$ as an example. We have three cases:

      i. $C'[x] \in \mathrm{NF}_\beta$ and $P \in \mathrm{NF}_\beta$. Then $F_l(M) \equiv C'[x]\ F_l(P)$. We have also $M \in \mathrm{NF}_\beta$ and, by the induction hypothesis, $F_l(P) \equiv P$. Therefore

$$F_l(M) \equiv C'[x]\ F_l(P) \equiv C'[x]\ P \equiv M$$

      as required by $(*)$.

ii. $C'[x] \in \text{NF}_\beta$ and $P \notin \text{NF}_\beta$. Then $F_l(M) \equiv C'[x]\ F_l(P)$ and by the induction hypothesis $P \to_\beta F_l(P)$. It follows from the compatibility of $\to_\beta$ that $M \to_\beta F_l(M)$ and therefore $(*)$ is satisfied.

iii. $C'[x] \notin \text{NF}_\beta$. In this case we have $F_l(M) \equiv F_l(C'[x])\ P$. From the induction hypothesis it follows that $C'[x] \to_\beta F_l(C'[x])$ and therefore by the compatibility of $\to_\beta$ that $M \to_\beta F_l(M)$.

The other forms of C are similar, only varying in the number of subcases to consider.

ii. $M \equiv \lambda x.P$, or $M \equiv \text{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. In all these cases F_l is applied under the constructor and $M \in \text{NF}_\beta$ if, and only if, $P \in \text{NF}_\beta$. It follows from the compatibility of $\to_\beta$ and the induction hypothesis that $(*)$ is satisfied.

iii. $M \equiv \langle P, Q\rangle$. The reduction strategy is applied on one of the subterms and the case is similar to Case i(b) above.

iv. $M \equiv C[(\lambda x.P)\ Q]$, or $M \equiv C[\pi_i(\langle P_1, P_2\rangle)]$, or $M \equiv C[\text{case}(\text{in}_i(P),\ x.Q_1,\ x.Q_2)]$, or $M \equiv C[(\Lambda\alpha.P)\ t]$, or $M \equiv C[\text{let } [\![x, \alpha]\!] = [\![P, t]\!] \text{ in } Q]$. In all these cases the redex is replaced by its contractum. It follows from compatibility that $M \to_\beta F_l(M)$.

This exhausts the possibilities, and we conclude that F_l is a reduction strategy. □

It follows from subject reduction that $F_l(M)$ is typable.

5.2 The Reduction Strategy F_l is Normalising

We now prove that the reduction strategy F_l is normalising. We follow [Tak95] closely, though it is more complicated due to our calculus having five constructors. In fact, compared to her proof, we have only used a different reasoning at one point in the proof. The overall idea is that the reductions in a term can be divided into two types: loosely speaking, the *head reductions* reduce the term until its outermost constructors are found, while the *internal reductions* reduce under the outermost constructors. The key insight is that the outermost constructors can never be reduced and therefore the two kinds of reductions are commutative. Thus, given any reduction path we can find a reduction path where the head reductions are done first, followed by the internal reductions. When we have established this, it suffices to see that this reduction path is closely related to the reduction path defined by F_l. We will restrict ourselves to typable terms when convenient to ease the proof.

In Λ^K the only constructor is abstraction. Therefore, any normal form (or head normal form) has a non-negative number of outermost abstractions. With five constructors, a normal form can have a mix of these five constructors at the outermost level. We therefore introduce the notion of *head context*.

Definition 5.6 The set of *head contexts* $\mathcal{C}_\text{h}$ on Λ_M is defined by the following grammar:

$$\mathcal{C}_\text{h} \ni C_h ::= []_j \mid \lambda x.C_h \mid \langle C_h, C_h\rangle \mid \text{in}_i(C_h) \mid \Lambda\alpha.C_h \mid [\![C_h, t]\!] \ .$$

All the indices $j \in \mathbb{N}$ are distinct and read $1, \ldots, n$ from left to right. The *arity* of the head context is n.

Let $M_1, \ldots, M_n$ be Λ_M-terms. The *filling* of a head context C_h of arity n with these terms is written $C_h[M_1, \ldots, M_n]$ and is defined by

$$C_h[M_1, \ldots, M_n] \equiv C_h\{[]_1 := M_1, \ldots, []_n := M_n\} \ .$$

♦

We will omit the indices in the following, as they are implicit when we write down a term.

Using the decomposition of Proposition 5.3 we have the following:

Proposition 5.7 *Let $M \in \Lambda_{\mathrm{M}}$ be a typable term. There is a* decomposition *of the term into a head context C_h of arity n, some leftmost evaluation contexts $C_1, \ldots, C_n \in \mathcal{C}_{\mathrm{lm}}$, and some terms $R_1, \ldots, R_n \in \Lambda_{\mathrm{M}}$. The decomposition satisfies*

$$M \equiv C_h[C_1[R_1], \ldots, C_n[M_n]] \tag{12}$$

where each of R_k fulfils one of the following for:

i. $R_k \equiv x$,

ii. $R_k \equiv (\lambda x.P)\, Q$,

iii. $R_k \equiv \pi_i(\langle P, Q\rangle)$,

iv. $R_k \equiv \mathrm{case}(\mathrm{in}_i(O),\, x.P,\, x.Q)$,

v. $R_k \equiv (\Lambda\alpha.P)\, t$, *or*

vi. $R_k \equiv \mathrm{let}\, [\![x, \alpha]\!] = [\![P, t]\!]\, \mathrm{in}\, Q$.

We will use $C_h[\vec{C}[\vec{R}]]$ as a shorthand for $C_h[C_1[R_1], \ldots, C_n[R_n]]$. This shorthand is used, *mutatis mutandis*, for other notions by applying the notion on each context and term in the vector.

Proof (Proposition 5.7). The proof is intuitively clear: In each of the Cases ii–vi of Proposition 5.3 we can take the constructor to be part of the head context. In the other cases we simply choose $[\,]$ as our head context. Formally, the proof is by induction on the decomposition of M. □

With this notation established, we can define the division of β-reduction into head reduction and internal reduction.

Definition 5.8 (Head and Internal Reduction on Λ_{M})

i. The *head reduction* $\xrightarrow{\mathtt{h}}_\beta$ of typable Λ_{M}-terms is defined by

$$C_h[C_1[R_1], \ldots, C_n[R_n]] \xrightarrow{\mathtt{h}}_\beta C_h[C_1[R'_1], \ldots, C_n[R'_n]]$$

where C_h is an arbitrary head context of arity n, and $R_k \;\beta\; R'_k$ for some k with $1 \leq k \leq n$, and $R'_l \equiv R_l$ for all other $l \neq k$.

We write $\xrightarrow{\mathtt{h}}\!\!\!\!\!\rightarrow_\beta$ for the reflexive, transitive closure of $\xrightarrow{\mathtt{h}}_\beta$.

ii. All other $\to_\beta$-reductions on typable terms are *internal reductions*, written $\xrightarrow{\mathtt{i}}_\beta$. We write $\xrightarrow{\mathtt{i}}\!\!\!\!\!\rightarrow_\beta$ for the reflexive, transitive closure of $\xrightarrow{\mathtt{i}}_\beta$.

It should be stressed that the use of β in Case i of the definition means that the only compatibility of $\xrightarrow{\mathtt{h}}_\beta$ is the one defined by C_h. Secondly, we see that in a $\xrightarrow{\mathtt{h}}\!\!\!\!\!\rightarrow_\beta$-reduction we do not necessarily use the same head context in all the head reduction steps. Consider for instance the term $\pi_1(\langle\Lambda\alpha.\pi_2(\langle P, Q\rangle), R\rangle)$. It has the head reduction path

$$\pi_1(\langle\Lambda\alpha.\pi_2(\langle P, Q\rangle), R\rangle) \xrightarrow{\mathtt{h}}_\beta \Lambda\alpha.\pi_2(\langle P, Q\rangle) \xrightarrow{\mathtt{h}}_\beta \Lambda\alpha.Q\ .$$

In the first reduction step we use the head context $[\,]$, while we in the second use $\Lambda\alpha.[\,]$. However, the head context will be increasing in size, and at any point in the reduction

path the head context, can be derived from each of the previous contexts by substituting head contexts for the holes. More formally: Consider the head reduction path

$$C_h^1[M_1^1, \ldots M_{n_1}^1] \mathrel{-\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\rightarrow_\beta} \cdots \mathrel{-\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\rightarrow_\beta} C_h^m[M_1^m, \ldots M_{n_m}^m] \ .$$

For any indices i and j where $i < j$, we have $C_h^j \equiv C_h^i[D_h^1, \ldots, D_h^{m_i}]$ where $D_h^1, \ldots, D_h^{m_i}$ are head contexts.

We then define the *parallel β-reduction* that simultaneously reduces a number of β-redexes in a term. The redexes might be overlapping.

Definition 5.9 The parallel β-reduction $\Rightarrow_\beta$ on Λ_{M} is defined as follows:

i. $x \Rightarrow_\beta x$,

ii. $M\ N \Rightarrow_\beta M'\ N'$, if $M \Rightarrow_\beta M'$ and $N \Rightarrow_\beta N'$,

iii. $(\lambda x.M)\ N \Rightarrow_\beta M'\{x := N'\}$, if $M \Rightarrow_\beta M'$ and $N \Rightarrow_\beta N'$,

iv. $\pi_i(M) \Rightarrow_\beta \pi_i(M')$, if $M \Rightarrow_\beta M'$,

v. $\pi_i(\langle M_1, M_2\rangle) \Rightarrow_\beta M_i'$, if $M_i \Rightarrow_\beta M_i'$,

vi. $\mathrm{case}(M,\, x.N,\, x.O) \Rightarrow_\beta \mathrm{case}(M',\, x.N',\, x.O')$, if $M \Rightarrow_\beta M'$, and $N \Rightarrow_\beta N'$, and $O \Rightarrow_\beta O'$,

vii. $\mathrm{case}(\mathrm{in}_i(M),\, x.N_1,\, x.N_2) \Rightarrow_\beta N_i'\{x := M'\}$, if $M \Rightarrow_\beta M'$ and $N_i \Rightarrow_\beta N_i'$,

viii. $M\ t \Rightarrow_\beta M'\ t$, if $M \Rightarrow_\beta M'$,

ix. $(\Lambda\alpha.M)\ t \Rightarrow_\beta M\{\alpha := t\}$, if $M \Rightarrow_\beta M'$,

x. $\mathrm{let}\ [\![x, \alpha]\!] = M\ \mathrm{in}\ N \Rightarrow_\beta \mathrm{let}\ [\![x, \alpha]\!] = M'\ \mathrm{in}\ N'$, if $M \Rightarrow_\beta M'$ and $N \Rightarrow_\beta N'$,

xi. $\mathrm{let}\ [\![x, \alpha]\!] = [\![M, t]\!]\ \mathrm{in}\ N \Rightarrow_\beta N'\{\alpha := t, x := M'\}$, if $M \Rightarrow_\beta M'$ and $N \Rightarrow_\beta N'$, or

xii. $C_h[M_1, \ldots, M_n] \Rightarrow_\beta C_h[M_1', \ldots, M_n']$, if $M_k \Rightarrow_\beta M_k'$ for all k where $1 \leq k \leq n$.

The terms M, N, and O are arbitrary Λ_{M}-terms and C_h is a head context.

The *parallel internal β-reduction* $\mathrel{=\!\!\!\!\!\!\mathbf{i}\!\!\!\!\!\!\Rightarrow_\beta}$ is defined by $M \mathrel{=\!\!\!\!\!\!\mathbf{i}\!\!\!\!\!\!\Rightarrow_\beta} N$, if, and only if, $M \mathrel{-\!\!\!\!\!\!\mathbf{i}\!\!\!\!\!\!\twoheadrightarrow_\beta} N$ and $M \Rightarrow_\beta N$. ♦

Though parallel β-reduction can contract all β-redexes in a term, this does not necessarily result in a β-normal form as β-reduction can create new redexes. This is most easily demonstrated by the infinite term Ω that has the infinite parallel β-reduction path

$$\Omega \Rightarrow_\beta \Omega \Rightarrow_\beta \cdots \ .$$

More formally, we can prove that $\twoheadrightarrow_\beta$ is the reflexive, transitive closure of $\Rightarrow_\beta$. To this end we need the following lemma:

Lemma 5.10 *Let M and N be Λ_{M}-terms such that $M \Rightarrow_\beta N$. Then $M \twoheadrightarrow_\beta N$.*

Proof. We use induction on the derivation $M \Rightarrow_\beta N$:

i. $M \equiv x \Rightarrow x \equiv N$. Then $x \twoheadrightarrow_\beta^0 x$.

ii. $M \equiv P\,Q \Rightarrow_\beta P'\,Q' \equiv N$, or
$M \equiv \pi_i(P) \Rightarrow_\beta \pi_i(P') \equiv N$, or
$M \equiv \mathrm{case}(P,\, x.Q,\, x.R) \Rightarrow_\beta \mathrm{case}(P',\, x.Q',\, x.R') \equiv N$, or
$M \equiv P\,t \Rightarrow_\beta P'\,t \equiv N$, or
$M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q \Rightarrow_\beta \mathrm{let}\,[\![x, \alpha]\!] = P'\,\mathrm{in}\,Q' \equiv N$, or
$M \equiv C_h[P_1, \ldots, P_n] \Rightarrow_\beta C_h[P'_1, \ldots, P'_n] \equiv N$ where $P \Rightarrow_\beta P'$, and $Q \Rightarrow_\beta Q'$, and $R \Rightarrow_\beta R'$. All these cases are similar and we illustrate their proof by $M \equiv P\,Q$. By the induction hypothesis, we have $P \twoheadrightarrow_\beta P'$ and $Q \twoheadrightarrow_\beta Q'$. Using the compatibility of $\rightarrow_\beta$, we conclude that

$$M \equiv P\,Q \twoheadrightarrow_\beta P'\,Q \twoheadrightarrow_\beta P'\,Q' \equiv N\ .$$

iii. $M \equiv (\lambda x.P)\,Q \Rightarrow_\beta P'\{x := Q'\} \equiv N$, or
$M \equiv \mathrm{case}(\mathrm{in}_i(P),\, x.Q_1,\, x.Q_2) \Rightarrow_\beta Q'_i\{x := P'\}$, or
$M \equiv (\Lambda\alpha.P)\,t \Rightarrow_\beta P'\{\alpha := t\}$, or
$M \equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![P, t]\!]\,\mathrm{in}\,Q \Rightarrow_\beta Q'\{\alpha := t, x := P'\}$. We illustrate the case by $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![P, t]\!]\,\mathrm{in}\,Q$. By the induction hypothesis, we have $P \twoheadrightarrow_\beta P'$ and $Q \twoheadrightarrow_\beta Q'$. Using generalisations of Lemmata 1.8 and 1.10 we conclude that

$$\begin{aligned} M &\equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![P, t]\!]\,\mathrm{in}\,Q \\ &\rightarrow_\beta Q\{\alpha := t, x := P\} \\ &\twoheadrightarrow_\beta Q\{\alpha := t, x := P'\} \\ &\twoheadrightarrow_\beta Q'\{\alpha := t, x := P'\} \\ &\equiv N\ . \end{aligned}$$

iv. $M \equiv \pi_i(\langle P_1, P_2\rangle) \Rightarrow_\beta P'_i \equiv N$ where $P_i \Rightarrow_\beta P'_i$. By the induction hypothesis we have $P_i \twoheadrightarrow_\beta P'_i$. Using the compatibility of $\rightarrow_\beta$, we conclude

$$M \equiv \pi_i(\langle P_1, P_2\rangle) \rightarrow_\beta P_i \twoheadrightarrow_\beta P'_i\ .$$

This exhausts the possibilities and concludes the proof that $M \Rightarrow_\beta N$ implies $M \twoheadrightarrow_\beta N$. □

Corollary 5.11 *The relation $\twoheadrightarrow_\beta$ is the reflexive, transitive closure of $\Rightarrow_\beta$.*

Proof. We consider $M_1 \Rightarrow_\beta M_2 \Rightarrow_\beta \cdots \Rightarrow_\beta M_n$ for any $n \geq 0$. By the previous lemma $M_1 \twoheadrightarrow_\beta M_2 \twoheadrightarrow_\beta \cdots \twoheadrightarrow_\beta M_n$, *ie*, $M \twoheadrightarrow_\beta N$. To see that $M \twoheadrightarrow_\beta N$ implies $M \Rightarrow_\beta^n N$ we simply note that $M \rightarrow_\beta N$ implies $M \Rightarrow_\beta N$. Then the result follows by induction on the number of $\rightarrow_\beta$-steps. □

Corollary 5.12 *The relation $\Rightarrow_\beta$ has the subject reduction property.*

At last, we define the notation of a $\rhd_\beta$-reduction path.

Definition 5.13 The relation $\rhd_\beta$ on typable Λ_{M}-terms is defined as $M \rhd_\beta N$, if, and only if, there is a reduction path

$$M \equiv M_0 \mathrel{-\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\rightarrow_\beta} M_1 \mathrel{-\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\rightarrow_\beta} \cdots \mathrel{-\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\rightarrow_\beta} M_n \mathrel{=\!\!\!\!\!\!\mathbf{1}\!\!\!\!\!\!\Rightarrow_\beta} N \tag{13}$$

where $M_k \Rightarrow_\beta N$ for all k such that $0 \leq k \leq n$. ♦

We call the reduction path in (13) a $\rhd_\beta$-reduction path. In [Tak95] this is written $M * N$.

We now prove three lemmata stating some fundamental properties about $\rhd_\beta$-reduction paths. First two compatibility lemmata

Lemma 5.14 *Let C_h be a head context of arity n. Let $M_1, \ldots, M_n$ and $N_1, \ldots, N_n$ be terms that for all, k where $1 \leq k \leq n$, satisfy $M_k \rhd_\beta N_k$, we have*

$$C_h[M_1, \ldots, M_n] \rhd_\beta C_h[N_1, \ldots, N_n] \ .$$

Proof. The proof is simple: One takes all the head reductions first and then all the internal reductions. The conditions on $\rhd_\beta$-reduction paths are fulfilled as the reductions are done in different subterms. Formally the proof is carried out using induction on the structure of the head context. □

Lemma 5.15 *Let M and N be typable Λ_{M}-terms with $M \rhd_\beta N$. We then have*

i. $M\ P \rhd_\beta N\ Q$, *if M has type $\tau \to \sigma$,*

ii. $\pi_i(M) \rhd_\beta \pi_i(M)$, *if M has type $\tau \vee \sigma$,*

iii. $\mathrm{case}(M,\, x.P,\, x.P') \rhd_\beta \mathrm{case}(N,\, x.Q,\, x.Q')$, *if M has type $\tau \wedge \sigma$,*

iv. $M\ t \rhd_\beta N\ t$, *if M has type $\forall\alpha.\tau$, and*

v. $\mathrm{let}\ [\![x, \alpha]\!] = M \,\mathrm{in}\, P \rhd_\beta \mathrm{let}\ [\![x, \alpha]\!] = N \,\mathrm{in}\, Q$, *if M has type $\exists\alpha.\tau$,*

for all terms $P, P', Q, Q' \in \Lambda_{\mathrm{M}}$ of appropriate type where $P \Rightarrow_\beta Q$ and $P' \Rightarrow Q'$.

Corollary 5.16 *Let $M, N \in \Lambda_{\mathrm{M}}$ be Λ_{M}-terms where $M \rhd_\beta N$. Then for any leftmost evaluation context C we have $C[M] \rhd_\beta C[N]$.*

Proof (Lemma 5.15). The five cases are done along the same line of reasoning. We only consider the third case. We assume $M \rhd_\beta N$, *ie*,

$$M \equiv M_0 \ \text{-}\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\to_\beta\ M_1 \ \text{-}\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\to_\beta \cdots \ \text{-}\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\to_\beta\ M_n \ \text{=}\!\!\!\!\!\!\mathbf{i}\!\!\!\!\!\!\Rightarrow_\beta\ N$$

where $M_k \Rightarrow_\beta N$ for all k such that $0 \leq k \leq n$.

We will prove that $\mathrm{case}(M,\, x.P,\, x.Q) \rhd_\beta \mathrm{case}(N,\, x.P',\, x.Q')$ when $P \Rightarrow_\beta P'$ and $Q \Rightarrow_\beta Q'$. We do this by establishing the reduction path

$$\begin{aligned} \mathrm{case}(M,\, x.P,\, x.Q) &\equiv \mathrm{case}(C_0[R_0],\, x.P,\, x.Q) \\ &\ \text{-}\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\to_\beta\ \mathrm{case}(C_1[R_1],\, x.P,\, x.Q) \\ &\ \text{-}\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\to_\beta \cdots \\ &\ \text{-}\!\!\!\!\!\!\mathbf{h}\!\!\!\!\!\!\to_\beta\ \mathrm{case}(C_n[R_n],\, x.P,\, x.Q) \\ &\equiv \mathrm{case}(M_n,\, x.P,\, x.Q) \\ &\ \text{=}\!\!\!\!\!\!\mathbf{i}\!\!\!\!\!\!\Rightarrow_\beta\ \mathrm{case}(N,\, x.P,\, x.Q) \end{aligned} \qquad (*)$$

where $C_k[R_k] \Rightarrow_\beta N$ for all k where $0 \leq k \leq n$.

From this we can continue as follows: We see that $C_n[R_n] \twoheadrightarrow_\beta N$ from Corollary 5.11. Furthermore, $P \twoheadrightarrow_\beta Q$ and $P' \twoheadrightarrow_\beta Q'$ using the same corollary on the assumptions $P \Rightarrow_\beta Q$ and $P' \Rightarrow_\beta Q'$. Hence, we have the following reduction path

$$\begin{aligned} \mathrm{case}(C_n[R_n],\, x.P,\, x.Q) &\twoheadrightarrow_\beta \mathrm{case}(N,\, x.P,\, x.Q) \\ &\twoheadrightarrow_\beta \mathrm{case}(N,\, x.P',\, x.Q) \\ &\twoheadrightarrow_\beta \mathrm{case}(N,\, x.P',\, x.Q') \ . \end{aligned}$$

We have $\mathrm{case}(C_n[R_n],\, x.P,\, x.Q) \ \text{-}\!\!\!\!\!\!\mathbf{i}\!\!\!\!\!\!\twoheadrightarrow_\beta \mathrm{case}(N,\, x.P',\, x.Q')$ as all these reductions are internal. From Definition 5.9 we see that $\mathrm{case}(C_n[R_n],\, x.P,\, x.Q) \Rightarrow_\beta \mathrm{case}(N,\, x.P',\, x.Q')$. We conclude that $\mathrm{case}(C_n[R_n],\, x.P,\, x.Q) \ \text{=}\!\!\!\!\!\!\mathbf{i}\!\!\!\!\!\!\Rightarrow_\beta\ \mathrm{case}(N,\, x.P',\, x.Q')$ by recalling that

$T \mathrel{=\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!\rightarrow}_{\beta} U$, if, and only if, $T \Rightarrow_{\beta} U$ and $T \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!\rightarrow}_{\beta} U$. Furthermore, it follows directly from the assumptions that $\mathrm{case}(C_k[R_k],\, x.P,\, x.Q) \Rightarrow_{\beta} \mathrm{case}(N,\, x.P',\, x.Q')$ for all k where $0 \leq k \leq n$.

Hence, to conclude $\mathrm{case}(M,\, x.P,\, x.Q) \rhd_{\beta} \mathrm{case}(N,\, x.P,\, x.Q)$, it suffices to establish $(*)$. We consider two cases.

i. $M_k \not\equiv \mathrm{in}_i(S)$ for all k, *ie*, M_k is never an injection. In all the reductions we have $C_h = []$ as the other forms would either contradict that M_k has type $\tau \wedge \sigma$ (by subject reduction) or $M_k \not\equiv \mathrm{in}_i(S)$. We therefore have the reduction path

$$M \equiv C_0[R_0] \mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} C_1[R_1] \mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \cdots \mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} C_n[R_n] \equiv M_n \ .$$

where C_k is a leftmost evaluation context. We see that $\mathrm{case}(C_k,\, x.P,\, x.Q)$ is a leftmost evaluation context for all k where $1 \leq k \leq n$. We therefore have the reduction path

$$\begin{aligned} \mathrm{case}(M,\, x.P,\, x.Q) &\equiv \mathrm{case}(C_0[R_0],\, x.P,\, x.Q) \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(C_1[R_1],\, x.P,\, x.Q) \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \cdots \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(C_n[R_n],\, x.P,\, x.Q) \\ &\mathrel{=\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(N,\, x.P,\, x.Q) \end{aligned}$$

that meets the conditions in $(*)$.

ii. $M_k \equiv \mathrm{in}_i(S)$ for some k where $0 \leq k \leq n$. We consider the lowest such k. For $j < k$, the head context C_h^j is empty as the other forms would either contradict that M_j has type $\tau \wedge \sigma$ (by subject reduction), or that k is the smallest index such that $M_k \equiv \mathrm{in}_i(S)$. We therefore have

$$M \equiv C_0[R_0] \mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} C_1[R_1] \mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \cdots \mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} C_k[R_k] \equiv M_n \ .$$

From this we find the reduction path

$$\begin{aligned} \mathrm{case}(M,\, x.P,\, x.Q) &\equiv \mathrm{case}(C_0[R_0],\, x.P,\, x.Q) \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(C_1[R_1],\, x.P,\, x.Q) \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \cdots \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(C_k[R_k],\, x.P,\, x.Q) \\ &\equiv \mathrm{case}(M_k,\, x.P,\, x.Q) \ . \end{aligned} \qquad (\star)$$

When we continue with the reduction $M_k \rightarrow_{\beta} M_{k+1} \rightarrow_{\beta} \cdots \rightarrow_{\beta} M_n$, these reductions are now internal. By Definition 5.9, we have $M_n \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!\rightarrow}_{\beta} N$ as $M_n \mathrel{=\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!\rightarrow}_{\beta} N$. We can therefore construct the following continuation of the reduction path in $(\star)$:

$$\begin{aligned} \mathrm{case}(M_k,\, x.P,\, x.Q) &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(M_{k+1},\, x.P,\, x.Q) \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \cdots \\ &\mathrel{-\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(M_n,\, x.P,\, x.Q) \\ &\mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(N,\, x.P,\, x.Q) \ . \end{aligned}$$

From Definition 5.13 of $\rhd_{\beta}$-reduction paths we know that $M_l \Rightarrow_{\beta} N$ for all l. We conclude that $\mathrm{case}(M_k,\, x.P,\, x.Q) \mathrel{=\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!\rightarrow}_{\beta} \mathrm{case}(N,\, x.P,\, x.Q)$ as required in $(*)$.

This case split establishes that $(*)$ holds and therefore concludes the proof. □

We also state two substitution lemmata for $\rhd_{\beta}$-reductions.

Lemma 5.17 *Let $M, N \in \Lambda_{\mathrm{M}}$ be Λ_{M}-terms where $M \mathrel{=\!\!\!\twoheadrightarrow_\beta} N$. For all terms $P, Q \in \Lambda_{\mathrm{M}}$ where $P \rhd_\beta Q$ we have $M\{x := P\} \rhd_\beta N\{x := Q\}$ for any variable x.*

Proof. We assume that $M \mathrel{=\!\!\!\twoheadrightarrow_\beta} N$. We have

$$M \equiv C_h[\vec{C}[\vec{R}]] \mathrel{=\!\!\!\twoheadrightarrow_\beta} C_h[\vec{C'}[\vec{R'}]] \equiv N$$

where $\vec{C} \mathrel{=\!\!\!\twoheadrightarrow_\beta} \vec{C'}$ and one of the following holds for each k:

i. $R_k \equiv y \equiv R'_k$,

ii. $R_k \equiv (\lambda y.S)\, T \Rightarrow_\beta (\lambda y.S')\, T' \equiv R'_k$ where $S \Rightarrow_\beta S'$ and $T \Rightarrow_\beta T'$,

iii. $R_k \equiv \pi_i(\langle S, T\rangle) \Rightarrow_\beta \pi_i(\langle S', T'\rangle) \equiv R'_k$ where $S \Rightarrow_\beta S'$ and $T \Rightarrow_\beta T'$,

iv. $R_k \equiv \mathrm{case}(\mathrm{in}_i(S),\, y.T,\, y.U) \Rightarrow_\beta \mathrm{case}(\mathrm{in}_i(S'),\, y.T',\, y.U') \equiv R'_k$ where $S \Rightarrow_\beta S'$ and $T \Rightarrow_\beta T'$ and $U \Rightarrow_\beta U'$,

v. $R_k \equiv (\Lambda\alpha.S)\, t \Rightarrow_\beta S\{\alpha := t\} \equiv R'_k$ where $S \Rightarrow_\beta S'$,

vi. $R_k \equiv \mathrm{let}\, [\![y, \alpha]\!] = [\![S, t]\!]\, \mathrm{in}\, T \Rightarrow_\beta \mathrm{let}\, [\![y, \alpha]\!] = [\![S', t]\!]\, \mathrm{in}\, T' \equiv R'_k$ where $S \Rightarrow_\beta S'$ and $T \Rightarrow_\beta T'$.

We should prove that

$$\begin{aligned} M\{x := P\} &\equiv C_h[\,(\vec{C}\{x := P\})[(\vec{R}\{x := P\})]\,] \\ &\mathrel{=\!\!\!\twoheadrightarrow_\beta} C_h[\,(\vec{C'}\{x := Q\})[(\vec{R'}\{x := Q\})]\,] \\ &\equiv N\{x := Q\} \end{aligned}$$

We have $\vec{C}\{x := P\} \mathrel{=\!\!\!\twoheadrightarrow_\beta} \vec{C'}\{x := Q\}$ and therefore $\vec{C}\{x := P\} \mathrel{=\!\!\!\twoheadrightarrow_\beta} \vec{C'}\{x := Q\}$ by generalisations of the substitution lemmata 1.8 and 1.10 to $\Rightarrow_\beta$. It therefore suffices to prove that $R_k\{x := P\} \rhd_\beta R'_k\{x := Q\}$ for each k, *ie*, that

$$R_k\{x := P\} \mathrel{-\!\!\!\twoheadrightarrow_\beta} R''_k \mathrel{=\!\!\!\twoheadrightarrow_\beta} R'_k\{x := Q\}\ . \tag{$*$}$$

If this holds we have

$$\begin{aligned} M\{x := P\} &\equiv C_h[\,(\vec{C}\{x := P\})[(\vec{R}\{x := P\})]\,] \\ &\rhd_\beta C_h[\,(\vec{C'}\{x := Q\})[(\vec{R'}\{x := Q\})]\,] \\ &\equiv N\{x := Q\}\ . \end{aligned}$$

using Lemma 5.14 and Corollary 5.16.

We prove $(*)$ by splitting into cases:

i. $R_k \equiv y \equiv R'_k$. We consider two subcases:

   (a) $x = y$. Then $R_k\{x := P\} \equiv P \rhd_\beta Q \equiv R'_k\{x := Q\}$ by assumption.

   (b) $x \neq y$. We have $R_k\{x := P\} \equiv y \equiv R'_k\{x := Q\}$ and from this we see that $R_k\{x := P\} \mathrel{=\!\!\!\twoheadrightarrow_\beta} R'_k\{x := Q\}$ and therefore $R_k\{x := P\} \rhd_\beta R'_k\{x := Q\}$.

ii. $R_k \equiv (\lambda y.S)\, T \Rightarrow_\beta (\lambda y.S')\, T' \equiv R'_k$, or
$R_k \equiv (\Lambda\alpha.S)\, t \Rightarrow_\beta (\Lambda\alpha.S')\, t \equiv R'_k$, or
$R_k \equiv \mathrm{case}(\mathrm{in}_i(S),\, y.T,\, y.U) \Rightarrow_\beta \mathrm{case}(\mathrm{in}_i(S'),\, y.T',\, y.U') \equiv R'_k$, or
$R_k \equiv \pi_i(\langle S, T\rangle) \Rightarrow_\beta \pi_i(\langle S', T'\rangle) \equiv R'_k$, or
$R_k \equiv \mathrm{let}\, [\![y, \alpha]\!] = [\![S, t]\!]\, \mathrm{in}\, T \Rightarrow_\beta \mathrm{let}\, [\![y, \alpha]\!] = [\![S', t]\!]\, \mathrm{in}\, T' \equiv R'_k$ where $S \Rightarrow_\beta S'$, and $T \Rightarrow_\beta T'$, and $U \Rightarrow_\beta U'$. The cases are similar as we apply the substitution on a subterm. We can illustrate this using $R_k \equiv (\lambda y.S)\, T$. As $P \rhd_\beta Q$ we have in particular $P \Rightarrow_\beta Q$ and therefore, by the generalisation of the substitution lemmata 1.8

and 1.10, that $S\{x := P\} \Rightarrow_\beta S'\{x := Q\}$ and $T\{x := P\} \Rightarrow_\beta T'\{x := Q\}$. We have

$$\begin{aligned}((\lambda y.S)\,T)\{x := P\} &\equiv (\lambda y.(S\{x := P\}))\,(T\{x := P\})\\ &\overset{\mathrm{I}}{\Rightarrow}_\beta (\lambda y.(S'\{x := Q\}))\,(T'\{x := Q\})\end{aligned}$$

where $\overset{\mathrm{I}}{\Rightarrow}_\beta$ holds because the reductions are done in a head redex.

This exhausts the possibilities and concludes the proof of (∗). We therefore conclude that $M \overset{\mathrm{I}}{\Rightarrow}_\beta N$ and $P \rhd_\beta Q$ implies $M\{x := P\} \rhd_\beta N\{x := Q\}$. □

Lemma 5.18 *Let M and N be Λ_{M}-terms where $M \rhd_\beta N$. For any eigenterm t and any type variable α, we then have $M\{\alpha := t\} \rhd_\beta N\{\alpha := t\}$.*

Proof. The proof exploits the fact that there is no reductions done in a eigenterm. We therefore replace one eigenterm by another throughout the reduction path. Formally, the proof is by induction on the length on the $\xrightarrow{h}_\beta$-reduction path and on the relation $\overset{\mathrm{I}}{\Rightarrow}_\beta$ used in the $\rhd_\beta$-relation. □

With these basic properties at hand, we can prove that we can postpone internal reductions. We start out with the following main lemma.

Lemma 5.19 *Let M and N be typable Λ_{M}-terms where $M \Rightarrow_\beta N$. We have $M \rhd_\beta N$.*

Corollary 5.20 *Let M and N be typable Λ_{M}-terms where $M \Rightarrow_\beta N$. We then have $M \xrightarrow{h}_\beta P \overset{\mathrm{I}}{\Rightarrow}_\beta N$ for some term P.*

Proof (Lemma 5.19). We will prove that

$$M \Rightarrow_\beta N \Longrightarrow M \rhd_\beta N \tag{$*$}$$

by induction on the structure of $M \Rightarrow N$:

i. $M \equiv x \Rightarrow x \equiv N$. We have the reduction path $M \xrightarrow{h}\!\!\!\!\rightarrow_\beta{}^0 M \overset{\mathrm{I}}{\Rightarrow}_\beta N$, which is clearly a $\rhd_\beta$-reduction path.

ii. $M \equiv C_h[P_1, \dots, P_n] \Rightarrow_\beta C_h[Q_1, \dots, Q_n] \equiv N$ where n is the arity of C_h and $P_k \Rightarrow_\beta Q_k$ for all k. By the induction hypothesis $P_k \rhd_\beta Q_k$. Using Lemma 5.14 we conclude $C_h[P_1, \dots, P_n] \Rightarrow_\beta C_h[Q_1, \dots, Q_n]$.

iii. $M \equiv P\,t \Rightarrow_\beta P'\,t \equiv N$, or
$M \equiv \mathrm{case}(P,\, x.Q,\, x.R) \Rightarrow_\beta \mathrm{case}(P',\, x.Q',\, x.R') \equiv N$, or
$M \equiv \mathrm{let}\,⦃x, \alpha⦄ = P \,\mathrm{in}\, Q \Rightarrow_\beta \mathrm{let}\,⦃x, \alpha⦄ = P' \,\mathrm{in}\, Q' \equiv N$, or
$M \equiv \pi_i(P) \Rightarrow_\beta \pi_i(P') \equiv N$, or
$M \equiv P\,Q \Rightarrow_\beta P'\,Q' \equiv N$ where $P \Rightarrow_\beta P'$, and $Q \Rightarrow_\beta Q'$ and $R \Rightarrow_\beta R'$. The cases are similar as they can be regarded as $C[P] \Rightarrow_\beta C'[P']$, for some leftmost evaluation contexts C and C' where $C \Rightarrow_\beta C'$. By the induction hypothesis, we have $P \Rightarrow^* P'$ and using Corollary 5.16 we conclude that $C[P] \rhd_\beta C'[P']$.

iv. $M \equiv (\lambda x.P)\,Q \Rightarrow_\beta P'\{x := Q'\} \equiv N$, or
$M \equiv \mathrm{case}(\mathrm{in}_i(P),\, x.Q_1,\, x.Q_2) \Rightarrow_\beta Q_i'\{x := P'\}$, or
$M \equiv (\Lambda\alpha.P)\,t \Rightarrow_\beta P'\{\alpha := t\}$, or
$M \equiv \mathrm{let}\,⦃x, \alpha⦄ = ⦃P, t⦄ \,\mathrm{in}\, Q \Rightarrow_\beta Q'\{\alpha := t, x := P'\}$. The cases are similar and we illustrate their proofs by the considering the case $M \equiv \mathrm{let}\,⦃x, \alpha⦄ = ⦃P, t⦄ \,\mathrm{in}\, Q$. By

the induction hypothesis, $P \twoheadrightarrow_\beta P'$ and $Q \twoheadrightarrow_\beta Q'$. Using the generalisations of Lemmata 1.8 and 1.10 we conclude that

$$\begin{aligned} M \equiv{} & \text{let}\, \{\!|x,\alpha|\!\} = \{\!|P,t|\!\} \,\text{in}\, Q \\ & \to_\beta Q\{\alpha := t, x := P\} \\ & \twoheadrightarrow_\beta Q\{\alpha := t, x := P'\} \\ & \twoheadrightarrow_\beta Q'\{\alpha := t, x := P'\} \\ & \equiv N \ . \end{aligned}$$

As $M \Rightarrow_\beta N$, we see that this reduction path satisfies the conditions for $\rhd_\beta$-reduction paths.

v. $M \equiv \pi_i(\langle P_1, P_2\rangle) \Rightarrow_\beta P_i' \equiv N$, where $P_i \Rightarrow_\beta P_i'$. By the induction hypothesis, we have $P_i \rhd_\beta P_i'$. We have

$$M \equiv \pi_i(\langle P_1, P_2\rangle) \to_\beta P_i \rhd_\beta P_i' \ .$$

As $M \Rightarrow_\beta P_i'$ by assumption, all steps in this reduction path fulfils the conditions for $\rhd_\beta$-reduction paths.

This exhausts the possibilities and also concludes the proof that $M \Rightarrow_\beta N$ implies $M \rhd_\beta N$. □

Lemma 5.21 (Postponement of internal reductions) *Let M and N be typable Λ_{M}-terms satisfying $M \overset{\mid}{\Rightarrow}_\beta P \overset{\not h}{\to}_\beta N$ for some term P. Then there is a term Q such that $M \overset{\not h}{\twoheadrightarrow}_\beta Q \overset{\mid}{\Rightarrow}_\beta N$.*

Proof. We let the terms M and N be given and assume that $M \overset{\mid}{\Rightarrow}_\beta P \overset{\not h}{\to}_\beta N$. This can only be achieved by

$$M \equiv C_h[\, \vec{C}[\vec{R}]\,] \overset{\mid}{\Rightarrow}_\beta P \equiv C_h[\, \vec{C}'[\vec{R}']\,] \overset{\not h}{\to}_\beta C_h[\, \vec{C}'[\vec{R}'']\,] \equiv N$$

where $\vec{C} \Rightarrow_\beta \vec{C}'$, and $\vec{R} \overset{\mid}{\Rightarrow}_\beta \vec{R}'$, and $R_k' \overset{\not h}{\to}_\beta R_k''$ for one k and $R_l'' \equiv R_l'$ for all $l \neq k$. We case split on the reduction $R_k' \overset{\not h}{\to}_\beta R_k''$:

i. $R_k' \equiv (\lambda x.S')\,T' \overset{\not h}{\to}_\beta S'\{x := T'\} \equiv R_k''$, or
$R_k' \equiv \text{case}(\text{in}_i(S'),\, x.T_1',\, x.T_2') \overset{\not h}{\to}_\beta T_i'\{x := S'\} \equiv R_k''$, or
$R_k' \equiv (\Lambda\alpha.S')\,t \overset{\not h}{\to}_\beta S'\{\alpha := t\} \equiv R_k''$, or
$R_k' \equiv \text{let}\,\{\!|x,\alpha|\!\} = \{\!|S',t|\!\}\,\text{in}\, T' \overset{\not h}{\to}_\beta T'\{\alpha := t, x := S'\} \equiv R_k''$. These cases are similar and we illustrate their proof by considering the case

$$R_k' \equiv \text{let}\,\{\!|x,\alpha|\!\} = \{\!|S',t|\!\}\,\text{in}\, T' \overset{\not h}{\to}_\beta T'\{\alpha := S', x := t\} \equiv R_k'' \ .$$

In this case we have $R_k \equiv \text{let}\,\{\!|x,\alpha|\!\} = \{\!|S,t|\!\}\,\text{in}\, T$ where $S \Rightarrow_\beta S'$ and $T \Rightarrow_\beta T'$. We can do the following reductions:

$$\begin{aligned} M \equiv{} & C_h[\, C_1[R_1], \ldots, C_k[\text{let}\,\{\!|x,\alpha|\!\} = \{\!|t,S|\!\}\,\text{in}\, T], \ldots, C_n[R_n]\,] \\ & \overset{\not h}{\to}_\beta C_h[\, C_1[R_1], \ldots, C_k[T\{\alpha := S, x := t\}], \ldots, C_n[R_n]\,] \\ & \Rightarrow_\beta C_h[\, C_1'[R_1'], \ldots, C_k[T'\{\alpha := t, x := S'\}], \ldots, C_n'[R_n']\,] \\ & \equiv N \end{aligned}$$

where the last step follows from generalisations of the substitution lemmata 1.8 and 1.10. Using Corollary 5.20 we conclude that $M \overset{\not h}{\twoheadrightarrow}_\beta Q \overset{\mid}{\Rightarrow}_\beta N$ for some term Q.

ii. $R'_k \equiv \pi_i(\langle S'_1, S'_2\rangle) \xrightarrow{\mathtt{h}\to}_\beta S'_i \equiv R''_k$. In this case $R_k \equiv \pi_i(\langle S_1, S_2\rangle)$ where $S_1 \Rightarrow_\beta S'_1$ and $S_2 \Rightarrow_\beta S'_2$. We have the following reduction path:

$$\begin{aligned} M &\equiv C_h[\, C_1[R_1], \ldots, C_k[\pi_i(\langle S_1, S_2\rangle)], \ldots, C_n[R_n]\,] \\ &\xrightarrow{\mathtt{h}\to}_\beta C_h[\, C_1[R_1], \ldots, C_k[S_i], \ldots, C_n[R_n]\,] \\ &\Rightarrow_\beta C_h[\, C'_1[R'_1], \ldots, C_k[S'_i], \ldots, C'_n[R'_n]\,] \\ &\equiv N\ . \end{aligned}$$

Using Corollary 5.20 we conclude that $M \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta Q \xRightarrow{\mathtt{i}\Rightarrow}_\beta N$ for some term Q.

This exhausts the possible forms of head reduction and concludes the proof that internal parallel reductions can be postponed. □

Proposition 5.22 *Let $M, N \in \Lambda^K$ be typable terms where $M \twoheadrightarrow_\beta N$. Then there is a term $K \in \Lambda^K$ such that $M \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta K \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta N$.*

Proof. We recall from Corollary 5.11 that $\twoheadrightarrow_\beta$ is the reflexive, transitive closure of $\Rightarrow_\beta$. We therefore have the parallel reduction path

$$M \equiv P_0 \Rightarrow_\beta P_1 \Rightarrow_\beta \cdots \Rightarrow_\beta P_n \equiv N\ .$$

Using induction on n we will prove that

$$M \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta K \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta N\ . \tag{$*$}$$

i. $n = 0$. We have $M \equiv N$ and therefore trivially ($*$).

ii. $n \geq 1$. By the induction hypothesis, we have the following reduction path

$$P_0 \Rightarrow_\beta P_1 \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta K' \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta N\ .$$

We use Corollary 5.20 to find the reduction path

$$P_0 \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta Q \xRightarrow{\mathtt{i}\Rightarrow}_\beta P_1 \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta{}^m K' \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta N\ .$$

We now use induction on m to prove that we can do the reduction

$$Q \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta R \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta K'\ . \tag{14}$$

(a) $m = 0$. We have $Q \xRightarrow{\mathtt{i}\Rightarrow}_\beta P_1 \equiv K'$. As $Q \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta P_1$ by the definition of $\xRightarrow{\mathtt{i}\Rightarrow}_\beta$ we have $Q \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta Q \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta P_1$.

(b) $m \geq 1$. We have the reduction path

$$Q \xRightarrow{\mathtt{i}\Rightarrow}_\beta P_1 \xrightarrow{\mathtt{h}\to}_\beta P'_1 \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta{}^{m-1} K'\ .$$

Using Lemma 5.21 we have

$$Q \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta Q' \xRightarrow{\mathtt{i}\Rightarrow}_\beta P'_1 \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta{}^{m-1} K'\ ,$$

and using the induction hypothesis we see

$$Q \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta Q' \xrightarrow{\mathtt{h}\twoheadrightarrow}_\beta K \xrightarrow{\mathtt{i}\twoheadrightarrow}_\beta K'$$

that fulfils (14).

This exhausts the possibilities and concludes the induction step in the main induction of the proof.

We conclude that internal reductions can be postponed. □

Theorem 5.23 *The reduction strategy F_l is normalising on typable terms.*

Proof. We consider any typable term M and assume that $M \in \mathrm{WN}_\beta$, *ie*, $M \twoheadrightarrow_\beta N$ for some $N \in \mathrm{NF}_\beta$. By the previous proposition we have the reduction path

$$M \twoheadrightarrow_{\mathbf{h}\beta} K \twoheadrightarrow_{\mathbf{i}\beta} N \ .$$

We should prove that $F_l^n(M) \equiv N$ for some $n \geq 0$. We will use induction on the structure of K. By Proposition 5.7 we have

$$K \equiv C_h[\, C_1[x], \ldots, C_k[x]\,]$$

for some head context C_h. (The leftmost evaluation contexts are filled with variables as there would otherwise be redexes in N and this would contradict it being a normal form.)

i. $C_h \equiv []$. We have $M \twoheadrightarrow_{\mathbf{h}\beta} C[x] \twoheadrightarrow_{\mathbf{i}\beta} N$. Using Definition 5.4, we find that $F_l^m(M) \equiv C[x]$. We split into cases on C.

   (a) $C \equiv []$. In this case we have $K \equiv x \equiv N$ and therefore $F_l^m(M) \equiv N$.

   (b) $C \equiv C'P$, or $C \equiv \pi_i(C')$, or $C \equiv C't$, or $C \equiv \mathrm{let}\,\{\!|x, \alpha|\!\} = C'\,\mathrm{in}\,P$, or $C \equiv \mathrm{case}(C', x.P, x.Q)$. All these cases are similar and we illustrate their proof using $C \equiv \mathrm{case}(C', x.P, x.Q)$. By Lemma 5.2, the case-destructor is never reduced. We therefore have $N \equiv \mathrm{case}(S, x.T, x.U)$ where $C'[x] \twoheadrightarrow_\beta S$, and $P \twoheadrightarrow_\beta T$ and $Q \twoheadrightarrow_\beta U$. Using the induction hypothesis, we find that $F_l^c(C'[x]) \equiv S$, and $F_l^p(P) \equiv T$, and $F_l^q(Q) \equiv U$ for some c, p and q. We see from Definition 5.4 that

   $$F_l^{m+c+p+q}(M) \equiv N$$

   as expected.

ii. $C_h \equiv \lambda x.C'_h$, or $C_h \equiv \langle C'_h, C''_h\rangle$, or $C_h \equiv \mathrm{in}_i(C'_h)$, or $C_h \equiv \Lambda\alpha.C'_h$, or $C_h \equiv \{\!|C'_h, t|\!\}$. These cases are similar and we illustrate them by $C_h \equiv \langle C'_h, C''_h\rangle$. We have

   $$M \twoheadrightarrow_{\mathbf{h}\beta} \langle P, Q\rangle \equiv \langle C'_h[R'], C''_h[R'']\rangle \twoheadrightarrow_{\mathbf{h}\beta} K \twoheadrightarrow_{\mathbf{i}\beta} N$$

   and therefore

   $$M \twoheadrightarrow_{\mathbf{h}\beta} \langle P, Q\rangle \equiv \langle C'_h[R'], C''_h[R'']\rangle \twoheadrightarrow_\beta N \equiv \langle S, T\rangle$$

   as head contexts only increase in size by head reduction. It is seen from Definition 5.4 that $F_l^m(M) \equiv \langle P, Q\rangle$ for some m. Furthermore by the induction hypothesis, $F_l^p(P) \equiv S$ and $F_l^q(Q) \equiv T$ for some p and q. Hence, we see from Definition 5.4 that

   $$F_l^{m+p+q}(M) \equiv \langle S, T\rangle \equiv N$$

   as required.

This exhausts the possibilities and concludes the proof that leftmost reduction is normalising. □

It is not only for simplicity that we do not try to relate the head normal form Q to a result of leftmost reduction: the two reduction forms are not confluent, and this would therefore not be possible. When reducing a pair, head reduction will immediately start reducing the second subterm if it is not in head normal form. The leftmost reduction will only start reducing the second subterm when the first subterm is in normal form. The

difference is illustrated by the term $\langle \lambda y.z\ ((\Lambda\alpha.y)\ t), (\Lambda\beta.x)\ t' \rangle$. Using leftmost reduction we have

$$\langle \lambda y.z\ ((\Lambda\alpha.y)\ t), (\Lambda\beta.x)\ t' \rangle \to_\beta \langle \lambda y.z\ y, (\Lambda\beta.x)\ t' \rangle \to_\beta \langle \lambda y.z\ y, x \rangle \in \mathrm{NF}_\beta \ ,$$

while head reduction gives

$$\langle \lambda y.z\ (\Lambda\alpha.y)\ t, (\Lambda\beta.x)\ t' \rangle \to_\beta \langle \lambda y.z\ (\Lambda\alpha.y)\ t, x \rangle \ ,$$

which is a head normal form.

The two notions of reduction only agree until the outermost constructor is found, but, as seen, this is sufficient to make the induction work. This is where, as mentioned in the introduction to the section, our proof differs from [Tak95].

5.3 Conclusion

To conclude this chapter: We have defined the leftmost reduction strategy and proved that it is a normalising reduction strategy. In the next chapter we will see that this strategy fulfils our purposes and is fairly simple to use.

Chapter 6

A Uniformly Normalising Subset of Λ_{M}

Having a uniformly normalising subset is central in inferring strong normalisation from weak normalisation as a uniformly normalising subset has the desired property by definition. When the uniformly normalising subset is found, the proof proceeds by mapping every term into a term of the uniformly normalising subset in a way that preserves reduction. We will therefore in this chapter define a β-uniformly normalising subset of Λ_{M} by the same approach as in Chapter 3: We first define a subset by a syntactic and a semantic condition. The subset is defined to have the *conservation property* under β-reductions. We then prove that the subset is closed under β-reduction—or to be more precise, we prove that it is closed under a normalising strategy. From this, we can conclude that the defined subset is β-uniformly normalising.

The vexatious problems we have to deal with is the pairing found in the terms $\langle Q, R\rangle$ and the terms $\mathrm{case}(P,\, x.Q,\, x.R)$, introduce intrinsically erasing reductions. When defining a uniformly normalising subset, we should therefore ensure that either part of the pairing construction can be erased without the contractum being less infinite than the redex. This can be achieved by restricting ourselves to subsets where the two parts of a pairing are equally infinite (in the sense that they are infinite under the same substitutions). However, the two parts being equally infinite might be lost in the process of normalising a term.

The problems are illustrated by the term[1]

$$\mathrm{case}(P,\, x.(\lambda z.x)\ (x\ x),\, x.x\ x)\ . \tag{15}$$

If P reduces to $\mathrm{in}_i(\omega)$, we get either

$$(\lambda z.\omega)\ (\omega\ \omega)$$

or

$$\omega\ \omega\ .$$

Thus both contractums are infinite terms. However, if we in (15) reduce the β_λ-redex in the first branch, we get

$$\mathrm{case}(P,\, x.x,\, x.x\ x)\ .$$

[1]Though we are now working in a typed calculus, the terms in this example are not typable. We conclude this thesis by proving that all typable terms of Λ_{M} are strongly normalising. Hence we can not find any typable term capable of producing infiniteness.

In this term, it is only the right term that becomes infinite, if P reduces to $\mathrm{in}_i(\omega)$.

The day is saved using the leftmost reduction strategy to reduce the term. This way, we only start reducing the minor subterms of a case-expression when the major subterm has been reduced to a normal form. When reducing one of the minor subterms of a case-expression, we therefore know that the case-expression will never be reduced. It is therefore no longer necessary to ensure that the two parts of the case-expression are equally infinite. The same situation applies to pairs: if they appear outermost, they will never be reduced and hence we have no reason to ensure that that the two parts are equally infinite.

In the first section, we introduce the subset defined by the semantic condition of $\mathbb{Y}$-goodness and prove that it its closed under F_l. In the second section section we prove that—with the additional restriction to $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$—this subset has the Conservation Property under F_l. Summing up, we conclude that the subset of $\mathbb{Y}$-good $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms is β-uniformly normalising.

6.1 $\mathbb{Y}$-goodness and Its Conservation Under F_l

In this section we define the subset of $\mathbb{Y}$-good terms. The subset is defined using a semantic condition, which is strong enough to prove the Conservation Property in the following section. The overall idea is that the two parts of a pairing that might be reduced should be equally infinite. In this way there is no harm done in erasing either of the parts when considering conservation of infinite terms under reduction.

Clearly, a case-expression with a normal form as major subterm will never be reduced. We can say even more: if the major subterm is $C[x]$ for some leftmost evaluation context C, then the case-expression is never reduced (this follows from Lemma 5.2 below). Similarly, a pair that appears outermost will never be reduced. In fact, the pair need not be outermost: as long as the pair is not a subterm of any redex, it will never be reduced. There is thus no reason that the two parts of such pairing constructs should be equally infinite. This is captured in the following definition of pairs and case-expressions being *relevant* (when considering uniform normalisation). This is a conservative approximation of the pairs and case-expressions that are reduced when reducing the term to normal form—we naturally have to include too many. The notion of relevant pairs was first introduced in [MN99].

Definition 6.1 (Relevant pairs and case-expressions)

i. For all typable Λ_{M}-terms M the set of *relevant pairs* $\mathrm{rel}_{\langle\rangle}(M)$ is defined as:

$$
\begin{array}{ll}
\mathrm{rel}_{\langle\rangle}(x) & = \emptyset \\
\mathrm{rel}_{\langle\rangle}(C[x]\ P) & = \mathrm{rel}_{\langle\rangle}(C[x]) \cup \mathrm{rel}_{\langle\rangle}(P) \\
\mathrm{rel}_{\langle\rangle}(\mathrm{let}\ \langle\!\langle y, \alpha \rangle\!\rangle = C[x]\ \mathrm{in}\ P) & = \mathrm{rel}_{\langle\rangle}(C[x]) \cup \mathrm{rel}_{\langle\rangle}(P) \\
\mathrm{rel}_{\langle\rangle}(\pi_i(C[x])) & = \mathrm{rel}_{\langle\rangle}(C[x]) \\
\mathrm{rel}_{\langle\rangle}(C[x]\ t) & = \mathrm{rel}_{\langle\rangle}(C[x]) \\
\mathrm{rel}_{\langle\rangle}(\mathrm{case}(C[x],\ y.P,\ y.Q)) & = \mathrm{rel}_{\langle\rangle}(C[x]) \cup \mathrm{rel}_{\langle\rangle}(P) \cup \mathrm{rel}_{\langle\rangle}(Q) \\
\mathrm{rel}_{\langle\rangle}(\lambda x.P) & = \mathrm{rel}_{\langle\rangle}(P) \\
\mathrm{rel}_{\langle\rangle}(\mathrm{in}_i(P)) & = \mathrm{rel}_{\langle\rangle}(P) \\
\mathrm{rel}_{\langle\rangle}(\Lambda\alpha.P) & = \mathrm{rel}_{\langle\rangle}(P) \\
\mathrm{rel}_{\langle\rangle}(\langle\!\langle P, t \rangle\!\rangle) & = \mathrm{rel}_{\langle\rangle}(P) \\
\mathrm{rel}_{\langle\rangle}(\langle P, Q \rangle) & = \mathrm{rel}_{\langle\rangle}(P) \cup \mathrm{rel}_{\langle\rangle}(Q)
\end{array}
$$

$$
\begin{aligned}
&\mathrm{rel}_{\langle\rangle}(C[(\lambda x.P)\,Q]) &&= \{\langle S,T\rangle \mid \langle S,T\rangle \subseteq C[(\lambda x.P)\,Q]\}\\
&\mathrm{rel}_{\langle\rangle}(C[\pi_i(\langle P,Q\rangle)]) &&= \{\langle S,T\rangle \mid \langle S,T\rangle \subseteq C[\pi_i(\langle P,Q\rangle)]\}\\
&\mathrm{rel}_{\langle\rangle}(C[\mathrm{case}(\mathrm{in}_i(P),\, x.Q,\, x.R)]) &&= \{\langle S,T\rangle \mid \langle S,T\rangle \subseteq C[\mathrm{case}(\mathrm{in}_i(P),\, x.Q,\, x.R)]\}\\
&\mathrm{rel}_{\langle\rangle}(C[(\Lambda\alpha.P)\,t]) &&= \{\langle S,T\rangle \mid \langle S,T\rangle \subseteq C[(\Lambda\alpha.P)\,t]\}\\
&\mathrm{rel}_{\langle\rangle}(C[\mathrm{let}\,[\![x,\alpha]\!] = [\![P,t]\!]\,\mathrm{in}\,Q]) &&= \{\langle S,T\rangle \mid \langle S,T\rangle \subseteq C[\mathrm{let}\,[\![x,\alpha]\!] = [\![P,t]\!]\,\mathrm{in}\,Q]\}
\end{aligned}
$$

where C is a leftmost evaluation context.

ii. For all typable Λ_{M}-terms M the set of *relevant* case-*expressions* $\mathrm{rel}_{\mathrm{c}}(M)$ is defined as:

$$
\begin{aligned}
&\mathrm{rel}_{\mathrm{c}}(x) &&= \emptyset\\
&\mathrm{rel}_{\mathrm{c}}(C[x]\;P) &&= \mathrm{rel}_{\mathrm{c}}(C[x]) \cup \mathrm{rel}_{\mathrm{c}}(P)\\
&\mathrm{rel}_{\mathrm{c}}(\mathrm{let}\,[\![y,\alpha]\!] = C[x]\,\mathrm{in}\,P) &&= \mathrm{rel}_{\mathrm{c}}(C[x]) \cup \mathrm{rel}_{\mathrm{c}}(P)\\
&\mathrm{rel}_{\mathrm{c}}(\pi_i(C[x])) &&= \mathrm{rel}_{\mathrm{c}}(C[x])\\
&\mathrm{rel}_{\mathrm{c}}(C[x]\;t) &&= \mathrm{rel}_{\mathrm{c}}(C[x])\\
&\mathrm{rel}_{\mathrm{c}}(\mathrm{case}(C[x],\, y.P,\, y.Q)) &&= \mathrm{rel}_{\mathrm{c}}(C[x]) \cup \mathrm{rel}_{\mathrm{c}}(P) \cup \mathrm{rel}_{\mathrm{c}}(Q)\\
&\mathrm{rel}_{\mathrm{c}}(\lambda x.P) &&= \mathrm{rel}_{\mathrm{c}}(P)\\
&\mathrm{rel}_{\mathrm{c}}(\mathrm{in}_i(P)) &&= \mathrm{rel}_{\mathrm{c}}(P)\\
&\mathrm{rel}_{\mathrm{c}}(\Lambda\alpha.P) &&= \mathrm{rel}_{\mathrm{c}}(P)\\
&\mathrm{rel}_{\mathrm{c}}([\![P,t]\!]) &&= \mathrm{rel}_{\mathrm{c}}(P)\\
&\mathrm{rel}_{\mathrm{c}}(\langle P,Q\rangle) &&= \mathrm{rel}_{\mathrm{c}}(P) \cup \mathrm{rel}_{\mathrm{c}}(Q)\\
&\mathrm{rel}_{\mathrm{c}}(C[(\lambda x.P)\,Q]) &&= \{\mathrm{case}(S,\, z.T,\, z.U) \mid\subseteq C[(\lambda x.P)\,Q]\}\\
&\mathrm{rel}_{\mathrm{c}}(C[\pi_i(\langle P,Q\rangle)]) &&= \{\mathrm{case}(S,\, z.T,\, z.U) \mid\subseteq C[\pi_i(\langle P,Q\rangle)]\}\\
&\mathrm{rel}_{\mathrm{c}}(C[\mathrm{case}(\mathrm{in}_i(P),\, x.Q,\, x.R)]) &&=\\
&\qquad\qquad \{\mathrm{case}(S,\, z.T,\, z.U) \mid\subseteq C[\mathrm{case}(\mathrm{in}_i(P),\, x.Q,\, x.R)]\}\\
&\mathrm{rel}_{\mathrm{c}}(C[(\Lambda\alpha.P)\,t]) &&=\\
&\{\mathrm{case}(S,\, z.T,\, z.U) \mid\subseteq C[(\Lambda\alpha.P)\,t]\}\\
&\mathrm{rel}_{\mathrm{c}}(C[\mathrm{let}\,[\![x,\alpha]\!] = [\![P,t]\!]\,\mathrm{in}\,Q]) &&=\\
&\qquad\qquad \{\mathrm{case}(S,\, z.T,\, z.U) \mid\subseteq C[\mathrm{let}\,[\![x,\alpha]\!] = [\![P,t]\!]\,\mathrm{in}\,Q]\}
\end{aligned}
$$

where C is a leftmost evaluation context. ⧫

The definition is unambiguous as the split on terms follows the decomposition in Proposition 5.3; the first four cases are the case $R \equiv x$ split on the six different kinds of leftmost evaluation contexts.

Next follows the definition of $\mathbb{Y}$-goodness. Generally, goodness ensures that F_l has the Conservation Property. Again, we have to take into consideration that we, in the following chapter, will need to introduce some free variables y that appear free in the terms under consideration. The reasoning done in the following would break down, if we did not leave out these variables from our considerations.

Definition 6.2 For a given set of variables $\mathbb{Y}$ a substitution θ is $\mathbb{Y}$-*neutral*, if, and only if, $y\theta = y$ for all $y \in \mathbb{Y}$.

The following lemma is obvious:

Lemma 6.3 *If θ and ϕ are $\mathbb{Y}$-neutral substitutions, then the composition $\theta\phi$ is a $\mathbb{Y}$-neutral substitution.*

Proof. As θ and ϕ are $\mathbb{Y}$-neutral, we have for all $y \in \mathbb{Y}$ that

$$y(\theta\phi) = (y\theta)\phi = y\phi = y$$

□

In the proof of uniform normalisation, $\mathbb{Y}$-goodness is used to ensure that the subset has the Conservation Property. As non-erasing reductions trivially have the Conservation Property, it is only necessary to consider the erasing reductions. As we will be working in $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, only $\beta_{\langle\rangle}$ and β_{in}-reductions are erasing. When reducing a $\beta_{\langle\rangle}$-redex $\pi_i(\langle Q, R\rangle)$ or a β_{in}-redex $\mathrm{case}(\mathrm{in}_i(P),\, x.Q,\, x.R)$ we should ensure that none of the subterms Q and R of the redex can be erased with the contractum being less infinite than the redex. This is encapsulated by the following definitions.

Definition 6.4 Let a set of variables $\mathbb{Y}$ be given.

i. A pair $\langle P, Q\rangle \in \Lambda_{\mathrm{M}}$ is $\mathbb{Y}$-good, if, and only if, for all $\mathbb{Y}$-neutral substitutions θ

$$P\theta \in \infty_\beta \iff Q\theta \in \infty_\beta \ .$$

ii. A case-expression $\mathrm{case}(P,\, x.Q,\, x.R) \in \Lambda_{\mathrm{M}}$ is $\mathbb{Y}$-good, if, and only if, for all $\mathbb{Y}$-neutral substitutions θ

$$Q\theta \in \infty_\beta \iff R\theta \in \infty_\beta \ .$$

⧫

Definition 6.5 ($\mathbb{Y}$-goodness)

i. A term $M \in \Lambda_{\mathrm{M}}$ is $\mathbb{Y}$-*good*, if, and only if,

   (a) all relevant pairs are $\mathbb{Y}$-good, and

   (b) all relevant case-expressions are $\mathbb{Y}$-good.

ii. A leftmost evaluation context C is $\mathbb{Y}$-good, if, and only if,

   (a) all pairs $\langle P, Q\rangle \subseteq C$ are $\mathbb{Y}$-good,

   (b) all case-expressions $\mathrm{case}(P,\, x.Q,\, x.R) \subseteq C$ are $\mathbb{Y}$-good, and

   (c) if $C \equiv \mathrm{case}(C',\, x.P,\, x.Q)$, then for all $\mathbb{Y}$-neutral substitutions θ

$$P\theta \in \infty_\beta \iff Q\theta \in \infty_\beta \ .$$

⧫

As seen from Definition 6.1, all pairs and case-expressions are relevant in a context filled with a redex. As we do not know what might be filled into a context, we consider all pairs and case-expressions when defining $\mathbb{Y}$-goodness of a context. The third case for $\mathbb{Y}$-goodness of contexts arises because the context $\mathrm{case}(C',\, x.P,\, x.Q)$ filled with a redex has an outermost case-expression that is not a subterm of the context.

It is an important difference to the definition in Chapter 3 that the relevant constructors are used directly in the definition of goodness. The discrepancy arises due to a slightly different focus between Chapter 3 and the present chapter. We recall from the conclusion of Chapter 3 that we used the notion of relevance to find the abstractions that could eventually become an operator of a $\mathbf{K}$-redex. The set of relevant abstractions is thus a conservative approximation of the abstractions in the original term that can be

reduced in a reduction path from the term. By requiring that these abstractions where $\mathbf{K}$-expanded, we ensured that there would not be created new $\mathbf{K}$-redexes in a reduction path. It was enough to give us the uniform normalisation.

In the present chapter, we use the notion of relevance to make a conservative approximation of the pairing constructions that will be reduced in a reduction path. We should ensure that the two parts of these pairing constructions are equally infinite. As seen from the discussion above, we cannot require that all pairing constructions are $\mathbb{Y}$-good as this property is not preserved under reduction. We therefore only require $\mathbb{Y}$-goodness of the relevant pairing constructions in the definition of $\mathbb{Y}$-goodness. As we shall see below, this gives us a property that is preserved under leftmost reduction. This is sufficient to establish uniform normalisation.

Lemma 6.6 *Let $M, N \in \Lambda_{\mathrm{M}}$ be typable terms where $N \subseteq M$. If M is $\mathbb{Y}$-good, then N is $\mathbb{Y}$-good.*

Proof. The proof consists of two parts. We should prove that all relevant pairs of N are $\mathbb{Y}$-good, and we should prove that all relevant case-expressions of N are $\mathbb{Y}$-good. The two parts are very similar.

The first part of the proof is to prove that all relevant pairs of N are $\mathbb{Y}$-good. As M is $\mathbb{Y}$-good, all relevant pairs of M are $\mathbb{Y}$-good by definition 6.5. Hence, it suffices to prove that $\mathrm{rel}_{\langle\rangle}(N) \subseteq \mathrm{rel}_{\langle\rangle}(M)$. The case $N \equiv M$ is trivially true, as $\mathrm{rel}_{\langle\rangle}(M) \subseteq \mathrm{rel}_{\langle\rangle}(M)$, and we concentrate on the case $N \subset M$. This case is handled by proving

$$N \subset M \Rightarrow \mathrm{rel}_{\langle\rangle}(N) \subseteq \mathrm{rel}_{\langle\rangle}(M) \qquad (*)$$

by induction on the structure of M. We split according to the decomposition of M:

i. $M \equiv x$: There is no N so that $N \subset x$ and $(*)$ is trivially true.

ii. $M \equiv \lambda x.P$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$, or $M \equiv C[x]\,P$, or $M \equiv \pi_i(C[x])$, or $M \equiv \mathrm{case}(C[x],\, y.P,\, y.Q)$, or $M \equiv C[x]\,t$, or $M \equiv \mathrm{let}\,[\![y, \alpha]\!] = C[x]\,\mathrm{in}\,P$. These cases are all similar: either $N \subseteq C[x]$, or $N \subseteq P$, or $N \subseteq Q$. We take the most complicated case, $M \equiv \mathrm{case}(C[x],\, y.P,\, y.Q)$, as an example.

   We have $N \subseteq C[x]$, or $N \subseteq P$, or $N \subseteq Q$. By the induction hypothesis we have $\mathrm{rel}_{\langle\rangle}(N) \subseteq \mathrm{rel}_{\langle\rangle}(C[x])$, or $N \subseteq \mathrm{rel}_{\langle\rangle}(P)$, or $N \subseteq \mathrm{rel}_{\langle\rangle}(Q)$, resp. We hence have

$$\mathrm{rel}_{\langle\rangle}(N) \subseteq \mathrm{rel}_{\langle\rangle}(C[x]) \cup \mathrm{rel}_{\langle\rangle}(P) \cup \mathrm{rel}_{\langle\rangle}(Q)\ .$$

iii. $M \equiv C[(\lambda x.P)\,Q]$, or $M \equiv C[\pi_i(\langle P,]\rangle)$, or $M \equiv C[\mathrm{case}(\mathrm{in}_i(P),\, x.Q,\, x.R)]$, or $M \equiv C[(\Lambda\alpha.P)\,t]$, or $M \equiv C[\mathrm{let}\,[\![x, \alpha]\!] = [\![P, t]\!]\,\mathrm{in}\,Q]$. These cases are alike as $\mathrm{rel}_{\langle\rangle}(M) = \{\langle S, T\rangle \mid \langle S, T\rangle \subseteq M\}$. As we have $N \subseteq M$, it follows that for any pair $\langle S, T\rangle \subseteq N$ we have $\langle S, T\rangle \subseteq M$. The pair is thus a relevant pair of M. Since any pair in N is in $\mathrm{rel}_{\langle\rangle}(M)$, in particular all relevant pairs of N are in $\mathrm{rel}_{\langle\rangle}(M)$.

This concludes the proof that $N \subset M \Rightarrow \mathrm{rel}_{\langle\rangle}(N) \subseteq \mathrm{rel}_{\langle\rangle}(M)$. We can do a similar proof for $N \subset M$ implies $\mathrm{rel}_{\mathrm{c}}(N) \subseteq \mathrm{rel}_{\mathrm{c}}(M)$: we should exchange $\mathrm{rel}_{\langle\rangle}$ by $\mathrm{rel}_{\mathrm{c}}$ and in the last case $\langle S, T\rangle$ by $\mathrm{case}(S,\, z.T,\, z.U)$. This concludes the proof that if M is $\mathbb{Y}$-good and $N \subseteq M$, then N is $\mathbb{Y}$-good. □

We then prove two lemmata considering conservation of $\mathbb{Y}$-goodness when filling the context and considering conservation under reduction.

Lemma 6.7 *Let C be a $\mathbb{Y}$-good leftmost evaluation context and M be a Λ_{M}-term. If*

i. all pairs $\langle S, T\rangle \subseteq M$ are $\mathbb{Y}$-good, and

ii. all case*-expressions $\mathrm{case}(S,\, x.T,\, x.U) \subseteq M$ are $\mathbb{Y}$-good,*

then $C[M]$ is $\mathbb{Y}$-good.

Proof. It is immediate from the Definition 6.5 that all pairs and case-expressions in $C[M]$ are $\mathbb{Y}$-good. Then in particular all relevant pairs and all relevant case-expressions are $\mathbb{Y}$-good. Hence $C[M]$ is $\mathbb{Y}$-good. □

Lemma 6.8 *Let t be an eigenterm and let M and N be Λ_{M}-terms where all pairs and* case*-expressions occuring as subterms in M and N are $\mathbb{Y}$-good. Then*

i. all pairs and case*-expressions in $M\{x := N\}$ are $\mathbb{Y}$-good, and*

ii. all pairs and case*-expressions in $M\{\alpha := t\}$ are $\mathbb{Y}$-good.*

Here x is any term variable not in $\mathbb{Y}$ and α is any type variable.

Proof. We consider Case i first. If $x \notin \mathrm{FV}(M)$ then the lemma is trivially true as $P\{x := N\} \equiv P$ where all pairs and case-expressions are $\mathbb{Y}$-good by assumption. We hence assume that $x \in \mathrm{FV}(M)$. We have two situations to consider

i. A pair $\langle S, T\rangle \subseteq P\{x := N\}$. We split further into subcases:

   (a) $\langle S, T\rangle \subseteq N$. In this case $\langle S, T\rangle$ is $\mathbb{Y}$-good by assumption.

   (b) $\langle S, T\rangle \not\subseteq N$. Using the substitution generation lemma we can find a pair $\langle P', Q'\rangle \subseteq M$ such that $\langle P', Q'\rangle\{x := N\} \equiv \langle P, Q\rangle$. We now consider any $\mathbb{Y}$-neutral substitution θ. As $x \notin \mathbb{Y}$ the composite substitution $\{x := N\}\theta$ is $\mathbb{Y}$-neutral by Lemma 6.3. As $\langle P', Q'\rangle$ is $\mathbb{Y}$-good by assumption, we conclude that

$$\infty_\beta \ni P\theta \equiv P'\{x := N\}\theta \Longleftrightarrow Q'\{x := N\}\theta \equiv Q \in \infty_\beta \ .$$

ii. A case-expression $\mathrm{case}(S,\, z.T,\, z.U) \subseteq P\{x := N\}$. We split into subcases as follows:

   (a) $\mathrm{case}(S,\, z.T,\, z.U) \subseteq N$. The case-expression is $\mathbb{Y}$-good by assumption.

   (b) $\mathrm{case}(S,\, z.T,\, z.U) \subseteq M\{x := N\}$. By a reasoning similar to Case i we conclude that $\mathrm{case}(S,\, z.T,\, z.U)$ is $\mathbb{Y}$-good.

This concludes the proof that all pairs and case-expressions in $M\{x := N\}$ are $\mathbb{Y}$-good. As the substitution generation lemma also holds for type substitution, the proof for type substitution (Case ii) is identical with α substituted for x and t for N. □

We prove that $\mathbb{Y}$-goodness is preserved under the reduction strategy F_l.

Proposition 6.9 *For all typable terms $M \in \Lambda_{\mathrm{M}}$ where M is $\mathbb{Y}$-good, the term $F_l(M)$ is $\mathbb{Y}$-good.*

Proof. We should prove that all relevant pairs and all relevant case-expressions in $F_l(M)$ are $\mathbb{Y}$-good. Again the two parts are similar and we only give the proof for relevant pairs. We use induction over the decomposition of M.

i. $M \equiv C[x]$: We divide into subcases:

(a) $C \equiv []$. Then $\mathrm{rel}_{\langle\rangle}(F_l(x)) \overset{5.4}{=} \mathrm{rel}_{\langle\rangle}(x) \overset{6.1}{=} \emptyset$.

(b) $C \equiv C'\,P$, or $C \equiv \pi_i(C')$, or $C \equiv \mathrm{case}(C', x.P, x.Q)$, or $C \equiv C'\,t$, or $C \equiv \mathrm{let}\,[\![x, \alpha]\!] = C'\,\mathrm{in}\,P$. These cases are all similar, only varying by the number of subcases. We take $C \equiv C'\,P$ as an illustration.

By Definition 6.1 and 6.5, all pairs in $\mathrm{rel}_{\langle\rangle}(M) = \mathrm{rel}_{\langle\rangle}(C'[x]) \cup \mathrm{rel}_{\langle\rangle}(P)$ are $\mathbb{Y}$-good. We consider two cases:

i. $C'[x] \notin \mathrm{NF}_\beta$. In this case we have $F_l(M) = F_l(C'[x])\,P$, and furthermore $\mathrm{rel}_{\langle\rangle}(F_l(M)) = \mathrm{rel}_{\langle\rangle}(F_l(C'[x])) \cup \mathrm{rel}_{\langle\rangle}(P)$. All pairs in $\mathrm{rel}_{\langle\rangle}(P)$ are $\mathbb{Y}$-good, as stated above, and it follows from the induction hypothesis that all pairs in $\mathrm{rel}_{\langle\rangle}(F_l(C'[x]))$ are $\mathbb{Y}$-good. Hence all relevant pairs in $F_l(M)$ are $\mathbb{Y}$-good.

ii. $C'[x] \in \mathrm{NF}_\beta$. Then $F_l(M) = C'[x]\,F_l(P)$ and the proof is the same as the previous case, except for an interchange of $C'[x]$ and P.

ii. $M \equiv \lambda x.P$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. These cases are all similar to Case i(b) above.

iii. $M \equiv C[(\lambda x.P)\,Q]$. Then $F_l(M) = C[P\{x := Q\}]$. As M is $\mathbb{Y}$-good, all relevant pairs are $\mathbb{Y}$-good. By Definition 6.1 all pairs of $C[(\lambda x.P)\,Q]$ are relevant pairs, so all pairs of M, and in particular of P and Q, are $\mathbb{Y}$-good. We note from Definition 6.5 that C is $\mathbb{Y}$-good. It follows from Lemmata 6.7 and 6.8 that all pairs in $F_l(M)$ are $\mathbb{Y}$-good. Then, in particular, all relevant pairs are $\mathbb{Y}$-good.

iv. $M \equiv C[\pi_i(\langle P_1, P_2\rangle)]$. In this case we have $F_l(M) = C[P_i]$. We note from Definition 6.1 that all pairs are relevant and therefore, as M is $\mathbb{Y}$-good, that all pairs are $\mathbb{Y}$-good. The context C is $\mathbb{Y}$-good and, using Lemma 6.7, we conclude that all pairs in $F_l(M)$ are $\mathbb{Y}$-good. Then, in particular, all relevant pairs are $\mathbb{Y}$-good.

v. $M \equiv C[\mathrm{case}(\mathrm{in}_i(P), x.Q_1, x.Q_2)]$. We note that $F_l(M) \equiv C[Q_i\{x := P\}]$. By Definition 6.1 all pairs are relevant, and it follows from the $\mathbb{Y}$-goodness of M that all pairs are $\mathbb{Y}$-good. Thus all pairs in P and Q_i are $\mathbb{Y}$-good and the context C is $\mathbb{Y}$-good. We conclude, using Lemmata 6.7 and 6.8, that all pairs in $F_l(M)$ are $\mathbb{Y}$-good.

vi. $M \equiv C[(\Lambda\alpha.P)\,t]$. We have $F_l(M) \equiv P\{\alpha := t\}$. We note from Definition 6.1 that all pairs are relevant and therefore, by Definition 6.5, that all pairs in M, and in particular P, are $\mathbb{Y}$-good. Furthermore the context C is $\mathbb{Y}$-good. We conclude from Lemmata 6.7 and 6.8 that all pairs in $F_l(M)$ are $\mathbb{Y}$-good. Then in particular all relevant pairs are $\mathbb{Y}$-good.

vii. $M \equiv C[\mathrm{let}\,[\![\alpha, x]\!] = [\![P, t]\!]\,\mathrm{in}\,Q]$. We have that $F_l(M) \equiv C[Q\{x := P, \alpha := t\}]$. It follows from Definitions 6.1 and 6.5 that all pairs are $\mathbb{Y}$-good. We also note that the context C is $\mathbb{Y}$-good. Using Lemmata 6.7 and 6.8, we conclude that all pairs in $F_l(M)$ are $\mathbb{Y}$-good and hence that all relevant pairs are $\mathbb{Y}$-good.

This exhausts the possibilities and concludes the proof that $F_l(M)$ is $\mathbb{Y}$-good. □

When reading the proof for the first time, it might not be clear that it would break down without the restriction to consider only the relevant pairs and case-expressions. However, as illustrated in the introduction, we cannot be sure that $\langle F_l(P), Q\rangle$ is $\mathbb{Y}$-good if $\langle P, Q\rangle$ is $\mathbb{Y}$-good. Thus, without the restriction, Case ii would no longer hold for $M \equiv \langle P, Q\rangle$ and Case v would not hold for $M \equiv \mathrm{case}(P, x.Q, x.R)$.

It is also worth noting that $\mathbb{Y}$-goodness is not preserved under arbitrary reductions. We can recall our example term from the introduction,

$$\mathrm{case}(P, x.(\lambda z.x)\,(x\,x), x.x\,x)\ , \tag{16}$$

and use some $P \not\equiv C[x]$ where all pairs and case-expressions in P are $\mathbb{Y}$-good. We see that the case-expression in (16) should be considered relevant and that the term is $\mathbb{Y}$-goodm as $(\lambda z.x)\,(x\,x)$ and $x\,x$ are equally infinite. However, if we reduce the redex $(\lambda z.x)\,(x\,x)$ we get the term

$$\mathrm{case}(P,\, x.x,\, x.x\,x)\ ,$$

which is not $\mathbb{Y}$-good. By using leftmost reduction, we ensure that P is reduced to a term on the form $C[x]$ before we start reducing any of the redexes in the minor subterms of the case-expression. But when $P \equiv C[x]$, then the case-expression is no longer relevant and the two minor subterms are not required to be equally infinite.

6.2 Infinity is Preserved Under F_l

As the last step in finding a uniformly normalising subset, we prove that $\mathbb{Y}$-good terms have the conservation property. In itself, $\mathbb{Y}$-goodness is not enough to ensure the conservation property, as β-reductions can still erase infinite subterms. We therefore further restrict ourselves to $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms.

Proposition 6.10 *Let $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ be a $\mathbb{Y}$-good typable $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term. If $M \in \infty_\beta$, then $F_l(M) \in \infty_\beta$.*

Proof. We shall prove that

$$M \in \infty_\beta \Longrightarrow F_l(M) \in \infty_\beta \qquad (*)$$

for any $\mathbb{Y}$-good term $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$. We consider any such M and proceed by induction on the structure of M, splitting according to its decomposition:

i. $M \equiv C[x]$. We have six subcases:

   (a) $C \equiv []$. As $M = x \in \mathrm{NF}_\beta$ there are no possibilities of $x \in \infty_\beta$. Hence, the $(*)$ is trivially true.

   (b) $C \equiv C'\,P$, or $C \equiv \pi_i(C')$, or $C \equiv C'\,t$, or $C \equiv \mathrm{let}\,\langle\!\langle x, \alpha\rangle\!\rangle = C'\,\mathrm{in}\,P$, or $C \equiv \mathrm{case}(C',\, x.P,\, x.Q)$. By Lemma 5.2 we have no possible reduction path such that $C'[x] \twoheadrightarrow_\beta \lambda x.S$. Hence, M can only be infinite, if one of subterms is infinite. We take $C \equiv C'\,P$ as illustration of the proof:

      i. $C'[x] \in \infty_\beta$. Then $F_l(M) = F_l(C'[x])\,P$. By the induction hypothesis $F_l(C'[x]) \in \infty_\beta$ and thus also $F_l(M) \in \infty_\beta$.

      ii. $P \in \infty_\beta$. If $C'[x] \notin \mathrm{NF}_\beta$, then $F_l(M) = F_l(C'[x])\,P$ and therefore clearly $F_l(M) \in \infty_\beta$.
      If $C'[x] \in \mathrm{NF}_\beta$, then $F_l(M) = C'[x]\,F_l(P)$. By the induction hypothesis $F_l(P) \in \infty_\beta$, and then also $F_l(M) \in \infty_\beta$.

   This proves that $F_l(C'[x]\,P)$ is infinite. The other forms of contexts are treated similarly.

ii. $M \equiv \lambda x.P$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv \langle\!\langle P, t\rangle\!\rangle$. In all these cases M can only be infinite by one of its subterms being infinite. Therefore, $(*)$ follows from the induction hypothesis by a reasoning similar to Case i(b) above.

iii. $M \equiv C[(\lambda x.P)\,Q]$. Then $F_l(M) = C[P\{x := Q\}]$. We can have $C[(\lambda x.P)\,Q] \in \infty_\beta$ in the following ways:

(a) $C \in \infty_\beta$. We have a $M \subseteq C$ such that $M \in \infty_\beta$ by the definition of $C \in \infty_\beta$. Then also $M \subseteq C[P\{x := Q\}]$ and therefore $C[P\{x := Q\}] \in \infty_\beta$ by the compatibility of $\to_\beta$.

(b) $Q \in \infty_\beta$. As $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, we have $x \in \mathrm{FV}(P)$ and therefore $Q \subseteq C[P\{x := Q\}]$.

(c) $\lambda x.P \in \infty_\beta$. Then $P \in \infty_\beta$ and also $P\{x := Q\} \in \infty_\beta$ by Corollary 1.9.

(d) $C[(\lambda x.P)\, Q] \twoheadrightarrow_\beta C'[(\lambda x.P')\, Q'] \to_\beta C'[P'\{x := Q'\}] \in \infty_\beta$ where $C \twoheadrightarrow_\beta C'$, and $P \twoheadrightarrow_\beta P'$ and $Q \twoheadrightarrow_\beta Q'$. By Lemmata 1.8 and 1.10, the following reduction path exists:

$$\begin{aligned} C[P\{x := Q\}] &\twoheadrightarrow_\beta C'[P\{x := Q\}] \\ &\overset{1.8}{\twoheadrightarrow}_\beta C'[P'\{x := Q\}] \\ &\overset{1.10}{\twoheadrightarrow}_\beta C'[P'\{x := Q'\}] \in \infty_\beta \ . \end{aligned}$$

iv. $M \equiv C[\pi_i(\langle P_1, P_2\rangle)]$. We have $F_l(M) = C[P_i]$. We have four ways to achieve $M \in \infty_\beta$:

(a) $C \in \infty_\beta$. Then $C[P_i] \in \infty_\beta$ by the compatibility of $\to_\beta$ as we have a $S \subseteq C$ such that $S \in \infty_\beta$.

(b) $P_i \in \infty_\beta$. We have $C[P_i] \in \infty_\beta$ by the compatibility of $\to_\beta$.

(c) $P_j \in \infty_\beta$, where $j \neq i$. The pair $\langle P_1, P_2\rangle$ is a relevant pair of M by Definition 6.1. The empty substitution $\emptyset$ is $\mathbb{Y}$-neutral and we therefore have

$$P_j\emptyset \in \infty_\beta \overset{6.4.\mathrm{i}}{\iff} P_i\emptyset \in \infty_\beta \ .$$

(d) $C[\pi_i(\langle P_1, P_2\rangle)] \twoheadrightarrow_\beta C'[\pi_i(\langle P_1', P_2'\rangle)] \to_\beta C'[P_i'] \in \infty_\beta$ where $C \twoheadrightarrow_\beta C'$, and $P_1 \twoheadrightarrow_\beta P_1'$ and $P_2 \twoheadrightarrow_\beta P_2'$. Using the compatibility of $\to_\beta$ we can construct the following infinite reduction path

$$\begin{aligned} C[P_i] &\twoheadrightarrow_\beta C'[P_i] \\ &\twoheadrightarrow_\beta C'[P_i'] \in \infty_\beta \ . \end{aligned}$$

v. $M \equiv C[\mathrm{case}(\mathrm{in}_i(P),\, x.Q_1,\, x.Q_2)]$. In this case $F_l(M) \equiv C[Q_i\{x := P\}]$. We have five ways to have $M \in \infty_\beta$:

(a) $C \in \infty_\beta$. Then $C[Q_i\{x := P\}] \in \infty_\beta$ by the compatibility of $\to_\beta$ as we have a $S \subseteq C$ such that $S \in \infty_\beta$.

(b) $P \in \infty_\beta$. We then have $C[Q_i\{x := P\}] \in \infty_\beta$ by the compatibility of $\to_\beta$ as $P \subseteq Q_i\{x := P\}$.

(c) $Q_i \in \infty_\beta$. Then $C[Q_i\{x := P\}] \in \infty_\beta$ follows from Corollary 1.9 by the compatibility of $\to_\beta$.

(d) $Q_j \in \infty_\beta$. The reduced case-expression is $\mathbb{Y}$-good as all case-expressions are $\mathbb{Y}$-good by Definition 6.1. As the empty substitution $\emptyset$ is $\mathbb{Y}$-neutral, we note that

$$Q_j\emptyset \in \infty_\beta \overset{6.4.\mathrm{ii}}{\iff} Q_i\emptyset \in \infty_\beta \ .$$

Then it follows from Corollary 1.9 that $C[Q_i\{x := P\}] \in \infty_\beta$ by the compatibility of $\to_\beta$.

(e) The reduction path

$$\begin{aligned} C[\mathrm{case}(\mathrm{in}_i(P),\, x.Q_1,\, x.Q_2)] &\to_\beta C'[\mathrm{case}(\mathrm{in}_i(P'),\, x.Q_1',\, x.Q_2')] \\ &\to_{\beta_{\mathrm{in}}} C'[Q_i'\{x := P'\}] \in \infty_\beta \end{aligned}$$

where $C \twoheadrightarrow_\beta C'$ and $P \twoheadrightarrow_\beta P'$ and $Q_1 \twoheadrightarrow_\beta Q_1'$ and $Q_2 \twoheadrightarrow_\beta Q_2'$. Using Lemmata 1.8 and 1.10, the following reduction path can be found:

$$\begin{aligned} C[Q_i\{x := P\}] &\twoheadrightarrow_\beta C'[Q_i\{x := P\}] \\ &\overset{1.8}{\twoheadrightarrow}_\beta C'[Q_i'\{x := P\}] \\ &\overset{1.10}{\twoheadrightarrow}_\beta C'[Q_i'\{x := P'\}] \in \infty_\beta \ . \end{aligned}$$

vi. $M \equiv C[(\Lambda\alpha.P)\, t]$. In this case $F_l(M) \equiv C[P\{\alpha := t\}]$. We case split on the three ways to have $M \in \infty_\beta$:

   (a) $C \in \infty_\beta$. Then $C[P\{\alpha := t\}] \in \infty_\beta$ by the compatibility of $\rightarrow_\beta$ as we have a $S \subseteq C$ such that $S \in \infty_\beta$.

   (b) $P \in \infty_\beta$. By a generalisation of Corollary 1.9 we have $P\{\alpha := t\} \in \infty_\beta$ and then $C[P\{\alpha := t\}] \in \infty_\beta$ by the compatibility of $\rightarrow_\beta$.

   (c) $C[(\Lambda\alpha.P)\, t] \twoheadrightarrow_\beta C'[(\Lambda\alpha.P')\, t] \twoheadrightarrow_\beta C'[P'\{\alpha := t\}] \in \infty_\beta$. Exploiting the compatibility of $\rightarrow_\beta$ and generalisations of Lemmata 1.8 and 1.10 to type substitutions, we can do the following reduction:

$$\begin{aligned} C[P\{\alpha := t\}] &\twoheadrightarrow C'[P\{\alpha := t\}] \\ &\twoheadrightarrow C'[P'\{\alpha := t\}] \in \infty_\beta \ . \end{aligned}$$

vii. $M \equiv C[\mathrm{let}\, \llbracket x, \alpha \rrbracket = \llbracket P, t \rrbracket \,\mathrm{in}\, Q]$. We have $F_l(M) = C[Q\{\alpha := t, x := P\}]$. We have the following ways to have $C[\mathrm{let}\, \llbracket x, \alpha \rrbracket = \llbracket P, t \rrbracket \,\mathrm{in}\, Q] \in \infty_\beta$:

   (a) $P \in \infty_\beta$. We have $x \in \mathrm{FV}(Q)$ as $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$, and from this follows that $P \subseteq C[Q\{\alpha := t, x := P\}]$. Therefore, we have $C[Q\{\alpha := t, x := P\}] \in \infty_\beta$ by the compatibility of $\rightarrow_\beta$.

   (b) $Q \in \infty_\beta$. We have $Q\{\alpha := t, x := P\} \in \infty_\beta$ by Corollary 1.9 and therefore $C[Q\{\alpha := t, x := P\}] \in \infty_\beta$ by the compatibility of $\rightarrow_\beta$.

   (c) $C \in \infty_\beta$. We have $C[Q\{\alpha := t, x := P\}] \in \infty_\beta$ by the compatibility of $\rightarrow_\beta$ and the fact that there is an infinite subterm S of C.

   (d) The reduction

$$\begin{aligned} C[\mathrm{let}\, \llbracket x, \alpha \rrbracket = \llbracket P, t \rrbracket \,\mathrm{in}\, Q] &\twoheadrightarrow_\beta C'[\mathrm{let}\, \llbracket x, \alpha \rrbracket = \llbracket P', t \rrbracket \,\mathrm{in}\, Q'] \\ &\rightarrow_\beta C'[Q'\{\alpha := t, x := P'\}] \in \infty_\beta \end{aligned}$$

   where $C \twoheadrightarrow_\beta C'$ and $P \twoheadrightarrow_\beta P'$ and $Q \twoheadrightarrow Q'$. Using Lemmata 1.8 and 1.10 for both term and type substitutions, we find the following reduction path

$$\begin{aligned} C[Q\{\alpha := t, x := P\}] &\twoheadrightarrow_\beta C'[Q\{\alpha := t, x := P\}] \\ &\overset{1.8}{\twoheadrightarrow}_\beta C'[Q'\{\alpha := t, x := P\}] \\ &\overset{1.10}{\twoheadrightarrow}_\beta C'[Q'\{\alpha := t, x := P'\}] \in \infty_\beta \ . \end{aligned}$$

This exhausts the possibilities and concludes the proof that infinity of $\mathbb{Y}$-good $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms is conserved under the reduction strategy F_l. □

It is important to notice how the restriction to $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ and the $\mathbb{Y}$-goodness are used in the previous proof. Restricting ourselves to $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ ensures that the substituted term is always a subterm of the contractum in the reductions using substitution. The $\mathbb{Y}$-goodness ensures that even if the erased term is infinite, we are sure that the contractum is infinite, as seen in Cases iv(c) and v(d).

We recall that uniform uniform normalisation means that weak normalisation implies strong normalisation. We conclude this section with the following theorem.

Theorem 6.11 *Let* $M \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ *be a typable* $\mathbb{Y}$*-good term. Then* $M \in \mathrm{UN}_\beta$.

Proof. We will prove that $M \in \mathrm{WN}_\beta \Rightarrow M \in \mathrm{SN}_\beta$. We suppose that $M \in \mathrm{WN}_\beta$. As F_l is normalising by Theorem 5.23, we have the following reduction path

$$M \to_\beta F_l(M) \to_\beta F_l^2(M) \to_\beta \cdots \to_\beta F_l^n(M) \in \mathrm{NF}_\beta \ . \qquad (*)$$

We now use *reductio ad absurdum* to prove that $M \in \mathrm{SN}_\beta$. We assume that M is infinite. By simple induction over n exploiting subject reduction, Proposition 4.14, Proposition 6.9, Proposition 6.10, and that F_l is a reduction strategy, all the terms $F_l^i(M)$ of $(*)$ are $\mathbb{Y}$-good, typable $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms with $F_l^i(M) \in \infty_\beta$. In particular, this implies $F_l^n(M) \in \infty_\beta$, We conclude that $M \notin \infty_\beta$ and hence $M \in \mathrm{UN}_\beta$. □

6.3 Conclusion

In conclusion, we have in this chapter defined a uniformly normalising subset of Λ_{M}. The subset will be a corner stone in the proof that we can infer strong normalisation from weak normalisation in the following chapter.

It is instructive to compare the proof above with the proof of Theorem 3.8 to see the similarity in the reasoning: both are instances of Lemma 1.7. We have thus found a uniformly normalising subset of Λ_{M}.

Chapter 7

Inferring Strong Normalisation from Weak Normalisation

In this chapter we will prove that the Barendregt-Geuvers-Klop conjecture holds for Λ_{M}. In itself, this result is not new; it is for instance proved by Statman [Sta78] and Stålmarck [Stå91]. They both consider classical first-order logic: Statman embeds classical first-order logic into second-order intuitionistic propositional logic; Stålmarck adapts Prawitz' notion of validity [Pra71]. The new achievement in the proof presented here is that the proof is based on reducing the problem of strong normalisation to weak normalisation of the *simpler* notion of reduction. Such techniques are important if one should keep hope to eventually prove the Barendregt-Geuvers-Klop conjecture; if one reduces the problem to a stronger system, one does as Münchhausen and tries to pull oneself up by ones hair.

As we know by now, the important point when inferring strong normalisation from weak normalisation is to prevent the erasure of infinite reduction paths under reduction. In his thesis [Klo80], Klop presents one landmark technique to achieve this. He introduces *systems with memory* derived from the original systems. In the systems with memory, copies of the subterms of the redex are saved for each reduction. For instance, when reducing the β_λ-redex $(\lambda x.P)\,Q$, the subterms P and Q are stored in the memory part of the contractum of $P\{x := Q\}$. In this way it is impossible to erase an infinite subterm, and though we are likely to double redexes, we can still have weak normalisation as we have removed one of the original redexes, namely the reduced redex $(\lambda x.P)\,Q$.

We can apply a simpler technique, also due to Klop [Klo80], where we translate terms of Λ_{M} to the calculus $\Lambda_{\mathrm{M}}^{[]}$ where subterms likely to be erased upon reduction are saved in the memory part. For the reductions using substitution it is sufficient to save the substitution variable, as the correct subterm will be stored in the memory part upon reduction. The idea is illustrated by the translation of the case-expression $\mathrm{case}(P,\, x.Q,\, x.R)$. It is transformed to

$$\mathrm{case}(P,\, x.[Q \mid x, R],\, x.[R \mid x, Q])$$

where the square brackets are used for denoting a term with the memory part after the bar. If we carry over the β-reductions trivially, a β_{in}-redex will now reduce as follows

$$\mathrm{case}(\mathrm{in}_1(P),\, x.[Q \mid x, R],\, x.[R \mid x, Q]) \to [Q \mid x, R]\{x := P\} \equiv [Q \mid P, R]\ .$$

As seen, no subterm is erased under reduction. Furthermore, the system includes some permutative reductions on the memory bracket—the reason will be discussed below. As no subterms are erased under reduction, we should be able to prove that all the translations of Λ_{M}-terms are uniformly normalising. However, we will not do so.

Instead, we will use a technique invented independently by Sørensen [Sør97] and Xi [Xi97]. The key insight behind their idea is that we can translate the terms with memory into our original typed λ-calculus. The memory term $[P \mid Q_1, \ldots, Q_n]$ is basically translated to $y\, P\, Q_1\, \ldots\, Q_n$ where y is a fresh variable. Some care has to be taken to ensure that the permutative reductions are conserved, but it is well-known [CD91, SF93] that this can be achieved using CPS-translations [Plo75, Rey93]. In fact, we are able to make the permutative reductions superfluous. We will therefore be aiming at a CPS-translation achieving this and at the same time simulating β-reductions, in a manner to be made explicit below. This makes the translation more complicated than sketched above, but it is worth the price as we reduce the question of strong normalisation to the question of weak normalisation for a simpler notion of reduction.

In outline, we will start out by presenting the term translations sketched above in the following section. We will formally define the system with memory brackets $\Lambda_{\mathrm{M}}^{[]}$ and the translation ι memorising subterms likely to be erased. We will also define the CPS-translation $\underline{\bullet}$. With this machinery established, we will use the following line of reasoning

$$\underline{\iota(M)} \in \mathrm{WN}_\beta \Longrightarrow \underline{\iota(M)} \in \mathrm{SN}_\beta \Longrightarrow \iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi} \Longrightarrow M \in \mathrm{SN}_{\beta\mathrm{p}} \tag{17}$$

where π denotes the permutative reductions associated with memory terms. The first implication of (17) is established in Section 7.2 by proving that the image of the combined mapping $\underline{\iota(\bullet)}$ ends in the uniformly normalising subset of the previous chapter. The second implication states that β-reduction in CPS-translated terms simulate the different notions of permutative reductions. The last implication is proved in Section 7.4 by establishing that we can regain the original term by removing the memory parts after doing all the reductions. In Section 7.5 we discuss why we have not included the full classical logic, as was the original objective of this thesis.

7.1 The Term Translations

In this section we introduce the two important notions for the line of reasoning sketched above. In the first subsection we introduce the system with memory $\Lambda_{\mathrm{M}}^{[]}$ and the translation ι from Λ_{M} to $\Lambda_{\mathrm{M}}^{[]}$. In the second subsection we define the CPS-translation from $\Lambda_{\mathrm{M}}^{[]}$ to Λ_{M}. In the last subsection we consider the combined mapping $\underline{\iota(\bullet)}$ from Λ_{M} to Λ_{M}.

7.1.1 Terms with Memory

To circumvent the erasing of infinite subterms Klop introduces terms with memory. For our purpose the following system $\Lambda_{\mathrm{M}}^{[]}$ inspired by Klop [Klo80, sec. 1.8] suffices:

Definition 7.1 (The terms of the $\Lambda_{\mathrm{M}}^{[]}$-calculus) Let $\Lambda_{\mathrm{M}}^{[]}$ be the set defined by extending Λ_{M} as follows.

$$\begin{array}{rcl} \Lambda_{\mathrm{M}}^{[]} \ni M & ::= & x \\ & \mid & \lambda x.M \mid M\ M \\ & \mid & \langle M, M\rangle \mid \pi_i(M) \\ & \mid & \mathrm{in}_i(M) \mid \mathrm{case}(M,\, x.M,\, x.M) \\ & \mid & \Lambda\alpha.M \mid Mt \\ & \mid & \{\!|M, t|\!\} \mid \mathrm{let}\ \{\!|x, \alpha|\!\} = M\ \mathrm{in}\ M \\ & \mid & [M \mid M, \ldots, M] \end{array}$$

where $i = 1, 2$. ♦

We will use $[P \mid \vec{Q}]$ as a shorthand for $[P \mid Q_1, \ldots, Q_n]$. Furthermore, we will consider Λ_{M} to be a proper subset of $\Lambda_{\mathrm{M}}^{[]}$. We straightforwardly extend the notions for Λ_{M} from Chapter 4 to $\Lambda_{\mathrm{M}}^{[]}$

We will map each term $M \in \Lambda_{\mathrm{M}}$ to a corresponding term $M' \in \Lambda_{\mathrm{M}}^{[]}$ so that no infinite subterms are erased when reducing M'. This is done by the following mapping ι, which saves a copy of the subterms that could be erased under β-reduction.

Definition 7.2 The translation $\iota : \Lambda_{\mathrm{M}} \to \Lambda_{\mathrm{M}}^{[]}$ is defined by

$$
\begin{array}{lcl}
\iota(x) & = & x \\
\iota(\lambda x.P) & = & \lambda x.[\iota(P) \mid x] \\
\iota(P\,Q) & = & \iota(P)\,\iota(Q) \\
\iota(\langle P, Q\rangle) & = & \langle [\iota(P) \mid \iota(Q)], [\iota(Q) \mid \iota(P)]\rangle \\
\iota(\pi_i(P)) & = & \pi_i(\iota(P)) \\
\iota(\mathrm{in}_i(P)) & = & \mathrm{in}_i(\iota(P)) \\
\iota(\mathrm{case}(P,\, x.Q,\, x.R)) & = & \mathrm{case}(\iota(P),\, x.[\iota(Q) \mid x, \lambda x.\iota(R)],\, x.[\iota(R) \mid x, \lambda x.\iota(Q)]) \\
\iota(\Lambda\alpha.P) & = & \Lambda\alpha.\iota(P) \\
\iota(P\,t) & = & \iota(P)\,t \\
\iota(\llbracket P, t\rrbracket) & = & \llbracket \iota(P), t\rrbracket \\
\iota(\mathrm{let}\,\llbracket x, \alpha\rrbracket = P\,\mathrm{in}\,Q) & = & \mathrm{let}\,\llbracket x, \alpha\rrbracket = \iota(P)\,\mathrm{in}\,[\iota(Q) \mid x]
\end{array}
$$

♦

The abstraction over the variable x in the translation of case is done for typing reasons: When typing $\mathrm{case}(P,\, x.Q,\, x.R)$, the variable x has, say, type σ when typing Q and type ϑ when typing R. The abstraction over x allows us to make this distinction and has no computational importance as the abstraction is never reduced. We see that $\iota(D_1[P]) = D_1'[\iota(P')]$ for some one-level destructor context D_1'.

By considering the different notions of reduction we can see that no subterms are erased under βp-reduction:

$$\iota((\lambda x.P)\,Q) = (\lambda x.[P' \mid x])\,Q' \longrightarrow_{\beta_\lambda} [P'\{x := Q'\} \mid Q']$$

$$\iota(\pi_1(\langle P, Q\rangle)) = \pi_1(\langle [P' \mid Q'], [Q' \mid P']\rangle) \longrightarrow_{\beta_\langle\rangle} [P' \mid Q']$$

$$
\begin{array}{rcl}
\iota(\mathrm{case}(\mathrm{in}_1(P),\, x.Q,\, x.R)) & = & \mathrm{case}(\mathrm{in}_1(P'),\, x.[Q' \mid x, \lambda x.R'],\, x.[R' \mid x, \lambda x.Q']) \\
 & & \longrightarrow_{\beta_{\mathrm{in}}} [Q'\{x := P'\} \mid P', \lambda x.R']
\end{array}
$$

$$\iota((\Lambda\alpha.P)\,t) = (\Lambda\alpha.P')\,t \longrightarrow_{\beta_\Lambda} P'\{\alpha := t\}$$

$$
\begin{array}{rcl}
\iota(\mathrm{let}\,\llbracket x, \alpha\rrbracket = \llbracket P, t\rrbracket\,\mathrm{in}\,Q) & = & \mathrm{let}\,\llbracket x, \alpha\rrbracket = \llbracket P', t\rrbracket\,\mathrm{in}\,[Q' \mid x] \\
 & & \longrightarrow_{\beta_{\llbracket\rrbracket}} [Q'\{\alpha := t, x := P'\} \mid P']
\end{array}
$$

$$
\begin{array}{rcl}
\iota(D_1[\mathrm{case}(P,\, x.Q,\, x.R)]) & = & D_1'[\mathrm{case}(P',\, x.[Q' \mid x, \lambda x.R'],\, x.[R' \mid x, \lambda x.Q'])] \\
 & & \longrightarrow_{\mathrm{p_{in}}} \mathrm{case}(P',\, x.D_1'[\,[Q' \mid x, \lambda x.R']\,],\, x.D_1'[\,[R' \mid x, \lambda x.Q']\,])
\end{array}
$$

$$
\begin{array}{rcl}
\iota(D_1[\mathrm{let}\,\llbracket x, \alpha\rrbracket = P\,\mathrm{in}\,Q]) & = & D_1'[\mathrm{let}\,\llbracket x, \alpha\rrbracket = P'\,\mathrm{in}\,\,[Q' \mid x]\,]) \\
 & & \longrightarrow_{\mathrm{p}_{\llbracket\rrbracket}} \mathrm{let}\,\llbracket x, \alpha\rrbracket = P'\,\mathrm{in}\,D_1'[\,[Q' \mid x]\,]
\end{array}
$$

However, redexes in the original Λ_{M}-term M can now be "blocked" by the memory brackets. For instance[1]

$$\pi_1((\lambda x.\langle x, x\,x\rangle)\,\lambda z.z\,z)\,y$$

[1]This example is taken from [MN99].

has the following reduction

$$\begin{aligned} & \pi_1((\lambda x.\langle x, x\ x\rangle)\ \lambda z.z\ z)\ y \\ \to_{\beta_\lambda} & \pi_1(\langle \lambda z.z\ z, (\lambda z.z\ z)\ \lambda z.z\ z\rangle)\ y \\ \to_{\beta_{\langle\rangle}} & (\lambda z.z\ z)\ y \\ \to_{\beta_\lambda} & y\ y \end{aligned}$$

However, from the ι-translated term we can only do the β_λ-reduction corresponding to the β_λ-reduction in the original path:

$$\begin{aligned} & \pi_1((\lambda x.[\langle [x \mid x\ x], [x\ x \mid x]\rangle \mid x])\ \lambda z.[z\ z \mid z])\ y \\ \to_{\beta_\lambda} & \pi_1([\langle [Z \mid Z\ Z], [Z\ Z \mid Z]\rangle \mid Z])\ y \end{aligned} \tag{18}$$

where $Z \equiv \lambda z.[z\ z \mid z]$. The $\beta_{\langle\rangle}$-reduction in the original reduction path is effectively "blocked" by the memory brackets.

The solution is to introduce the reduction π, which moves the argument of a redex in the original term past the brackets. We also introduce the reduction κ, which deletes the memory part of a term.

Definition 7.3 (The reductions of the $\Lambda^{[]}_{\mathrm{M}}$-calculus) In addition to the β-reductions of Λ_{M} there are the following notions of reductions on $\Lambda^{[]}_{\mathrm{M}}$:

$$\begin{array}{rcl} D_1[\,[P \mid \vec{Q}]\,] & \pi & [D_1[P] \mid \vec{Q}] \\ [P \mid \vec{Q}] & \kappa & P \end{array}$$

♦

With these reductions the reduction path of (18) can be continued :

$$\begin{aligned} \cdots \to_\pi & [\pi_1(\langle [Z \mid Z\ Z], [Z\ Z \mid Z]\rangle) \mid Z]\ y \\ \to_{\beta_{\langle\rangle}} & [[Z \mid Z\ Z] \mid Z]\ y \\ \twoheadrightarrow_\pi & [[Z\ y \mid Z\ Z] \mid Z] \\ \to_{\beta_\lambda} & [[[y\ y \mid y] \mid Z\ Z] \mid Z] \end{aligned} \tag{19}$$

Thus the combination of the reductions β and π corresponds to β-reductions in the original term. To get the original end contractum from (19) one must apply κ thrice, *ie*,

$$\cdots \to_\kappa [[y\ y \mid y] \mid Z\ Z] \to_\kappa [y\ y \mid y] \to_\kappa y\ y$$

7.1.2 CPS-translations

Continuations and *continuation passing style* [HD94, Rey93, Fis93, Maz71]—also called *CPS* for short—are useful tools used, for instance, in compilers [App92, KKR$^+$86] and partial evaluators [CD91, Jør98, JGS93]. The main assessments of CPS-translations are that all function arguments become weak head normal form (and are hence considered values in practical implementations), and strategically placed η-expansions delay computation. The overall effect is that the evaluation order is made explicit. This makes the evaluation of the program independent of the underlying evaluation scheme being call-by-value or call-by-name [Plo75]. By the same token one refers to a translation as, say, call-by-name or call-by-value, depending on the control flow it encodes.

The underlying idea is that under a given evaluation order, we can at any time divide the expression being evaluated into the current redex and the current context. The latter is called the current *continuation*. Take for instance the ML-expression $\mathrm{sqr}\ (2+4)$. As ML uses call-by-value $2+4$ should be evaluated to 6 before applying sqr. The current redex

is therefore $2 + 4$, while the current context is $\mathrm{sqr}\ []$. Using this insight, we can translate every expression into a function over the expression's context. When applied, the function evaluates the expression and fills the result into the context used as argument. Our ML-example translates to[2]

$$\mathrm{fn}\, k \Rightarrow (\mathrm{fn}\, l \Rightarrow k(\mathrm{sqr}\, l))(2 + 4)\ .$$

It is a basic insight in the theory of CPS-translations that, if the result of evaluating an expression is not a function value, the same result is obtained by applying the CPS-translation of the expression on the identity function. (This is well-defined as the CPS-translation is always a function.) Indeed, the above function evaluates to 36 when given the identity function $\mathrm{fn}\, x \Rightarrow x$ as argument.

A further convenience of CPS-translations is that complicated flow facilities in the source languages, for instance, exceptions and jumps can be simulated using application [FHK84]. In particular, we can simulate the π-reductions and the permutative reductions by β-reductions on CPS-translated terms. To illustrate this we can consider the CPS-translations, written $\underline{\bullet}$, of application and case-expression (these will be introduced formally below):

$$\begin{array}{lcl} \underline{P\,Q} & = & \lambda k.\underline{P}\,\lambda l.l\,\underline{Q}\,k \\ \underline{\mathrm{case}(P,\, x.Q,\, x.R)} & = & \lambda k.\underline{P}\,\lambda l.\mathrm{case}(l,\, x.\underline{Q}\,k,\, x.R\,k)\ . \end{array}$$

We can now consider the CPS-translation of the two sides of the p_{in_λ}-reduction on Page 39:

$$\mathrm{case}(P,\, x.Q_1,\, x.Q_2)R \rightarrow_{\mathrm{p_{in}}} \mathrm{case}(P,\, x.Q_1\ R,\, x.Q_2\ R)$$

of (8). The translation of the left-hand side is

$$\begin{array}{l} \lambda k.(\lambda l.\underline{P}\,\lambda m.\mathrm{case}(m,\, x.\underline{Q_1}\,l,\, x.\underline{Q_2}\,l))\,\lambda n.n\,\underline{R}\,k \\ \qquad\qquad \rightarrow_\beta \lambda k.\underline{P}\,\lambda m.\mathrm{case}(m,\, x.\underline{Q_1}\,\lambda n.n\,\underline{R}\,k,\, x.\underline{Q_2}\,\lambda n.n\,\underline{R}\,k)\ , \end{array} \quad (20)$$

and the translation of the right-hand side is

$$\begin{array}{l} \lambda k.\underline{P}\,\lambda m.\mathrm{case}(m,\, x.(\lambda l.\underline{Q_1}\,\lambda n.n\,\underline{R}\,l)\,k,\, x.(\lambda l.\underline{Q_2}\,\lambda n.n\,\underline{R}\,l)\,k) \\ \qquad\qquad \rightarrow_\beta \lambda k.\underline{P}\,\lambda m.\mathrm{case}(m,\, x.\underline{Q_1}\,\lambda n.n\,\underline{R}\,k,\, x.\underline{Q_2}\,\lambda n.n\,\underline{R}\,k)\ . \end{array} \quad (21)$$

As seen, the two translated terms are β-equivalent.

We now pursue these ideas and get rid of the memory brackets by embedding $\Lambda_{\mathrm{M}}^{[]}$ in Λ_{M}. In Section 7.3 we prove that using CPS-translations we can make the permutative reductions and the π-reductions superfluous. First, we define the CPS-translations:

Definition 7.4 Let $\mathbb{Y}$ be an infinite set of term variables not occuring in any other terms than terms produced by the following translation. The translation $\underline{\bullet} : \Lambda_{\mathrm{M}}^{[]} \rightarrow \Lambda_{\mathrm{M}}$ is

[2]The translation is based on the translation given for Core Scheme in [SF93]. As stated there, we should strictly speaking use CPS-versions sqr_k and $+_k$ of the primitive operations. The functions sqr_k and $+_k$ take the continuation as argument in addition to the ordinary arguments.

defined by

$$
\begin{array}{lcl}
\underline{x} & = & \lambda k.x\ k \\
\underline{\lambda x.P} & = & \lambda k.k\ \lambda x.\underline{P} \\
\underline{\langle P, Q\rangle} & = & \lambda k.k\ \langle \underline{P}, \underline{Q}\rangle \\
\underline{\mathrm{in}_i(P)} & = & \lambda k.k\ \mathrm{in}_i(\underline{P}) \\
\underline{\Lambda\alpha.P} & = & \lambda k.k\ \Lambda\alpha.\underline{P} \\
\underline{[\![P, t]\!]} & = & \lambda k.k\ [\![\underline{P}, t]\!] \\
\underline{P\ Q} & = & \lambda k.\underline{P}\ \lambda l.l\ \underline{Q}\ k \\
\underline{\pi_i(P)} & = & \lambda k.\underline{P}\ \lambda l.\pi_i(l)\ k \\
\underline{\mathrm{case}(P,\ x.Q,\ x.R)} & = & \lambda k.\underline{P}\ \lambda l.\mathrm{case}(l,\ x.\underline{Q}\ k,\ x.\underline{R}\ k) \\
\underline{P\ t} & = & \lambda k.\underline{P}\ \lambda l.l\ t\ k \\
\underline{\mathrm{let}\ [\![x, \alpha]\!] = P\ \mathrm{in}\ Q} & = & \lambda k.\underline{P}\ \lambda l.\mathrm{let}\ [\![x, \alpha]\!] = l\ \mathrm{in}\ \underline{Q}k \\
\underline{[P \mid Q_1, \ldots, Q_n]} & = & \lambda k.y\ (\underline{P}\ k)\ \underline{Q_1}\ \ldots\ \underline{Q_n}
\end{array}
$$

such that all occurrences of $y \in \mathbb{Y}$ in a term $\underline{M}$ are distinct variables and such that no free variables of P and Q are caught. We use k, l, m, n, and o to denote bound variables arising from the translations ⧫

This is a standard call-by-name CPS-translation [BHS97a]. The translations might look like black magic at first sight. They are understood best from the fact that the variables k and l are continuations, *ie*, they are functions representing a context, and upon application the value is filled into context and the evaluation is completed. In this setting all expressions, evaluated or unevaluated, are represented as abstractions over continuations.

A variable is translated to its η-expansion: When the variable is bound during the evaluation, it is bound to an expression being an abstraction over a continuation. By applying this expression to the continuation of the variable, the expression is evaluated and the computation continued by filling the result into the context. The next cases represent values under a weak-head normal form evaluation. Consequently, there is no more computation to be done, and the value is simply filled into the context by applying the continuation on the value.

The rest of the cases are the destructors, which evaluate to a redex if the major part evaluates to an operator. Taking application $P\,Q$ as an example, it is divided as $E[P]$ where $E = []\,Q$. When the expression P is evaluated to weak-head normal form, the result is an abstraction $\lambda x.P'$. As we use call-by-name, the argument should not be evaluated upon application, and the evaluation of $P\,Q$ is hence continued by applying $\lambda x.P'$ to Q. It is finished by filling the result of this application into the context of $P\,Q$. The continuation of P is thus $\lambda m.(m\,Q)\,k$. First m is applied on Q. The result is an expression that, when given the continuation k, finds the value of the expression and completes the computation by filling the result into the context represented by k. A similar description applies to the other destructors. The placement of the continuation k inside the case can be understood intuitively as a way of expressing that one of the branches should be chosen before continuing the evaluation. Similarly, the local bindings of let should be done before evaluating the body.

The last case, the translation of $[P \mid \vec{Q}]$, deserves a little more attention. As the memory parts are included to store subterms, the evaluation simply proceeds with the evaluation of P. We should keep all the subterms, but ensure that they cannot interfere with one another. We find inspiration in the standard way to encode a pair $\langle P, Q\rangle$ in the λ-calculus. This encoding is $\lambda k.k\,P\,Q$ where the terms P and Q can only interfere, if k is substituted by an abstraction. We use y instead of k and by choosing $y \in \mathbb{Y}$ we are guaranteed that y is free in the translated term. Therefore, $y\,(\underline{P}\,k)$ can never evaluate to an abstraction and the application $\big(y\,(\underline{P}\,k)\big)\,\underline{Q_1}$ is never reduced. The same argument applies to the following terms in $\vec{Q}$. In effect, we have encoded the $n+1$-tuple $\langle P, Q_1, \ldots, Q_n\rangle$.

This concludes the introduction of the translations. In Section 7.3.1 we will prove that the reductions of the translated terms simulate the reductions of the original terms. To do this we need to overcome a minor inconvenience of the translation in Definition 7.4, namely the introduction of internal redexes, which have no counterpart in the original term. In particular, as seen from (20) and (21), these redexes have the inconvenience that $M \to_\beta N$ does not imply $\underline{M} \to_\beta \underline{N}$, though $\underline{M} =_\beta \underline{N}$. We therefore introduce another invention by Plotkin [Plo75]:the *compacting CPS-translation* $\underline{\underline{\bullet}}$. It has the feature that $M \to_{\beta \mathrm{p} \pi} N$ implies $\underline{\underline{M}} \to_\beta \underline{\underline{N}}$. As an intermediate step to define $\underline{\underline{\bullet}}$ we need the map $\bullet : \bullet$,which removes some of the internal redexes.

Definition 7.5 (Compacting CPS) Let $\mathbb{Y}$ be the set of variables $\mathbb{Y}$ from Definition 7.4. The translations $\underline{\underline{\bullet}} : \Lambda_{\mathrm{M}}^{[]} \to \Lambda_{\mathrm{M}}$ and $\bullet : \bullet : \Lambda_{\mathrm{M}}^{[]} \times \Lambda_{\mathrm{M}} \longrightarrow \Lambda_{\mathrm{M}}$ are defined by

$$\underline{\underline{M}} = \lambda k.(M : k)$$

and

$$\begin{aligned}
x : K &= x\ K \\
(\lambda x.P) : K &= K\ (\lambda x.\underline{\underline{P}}) \\
\langle P, Q \rangle : K &= K\ \langle \underline{\underline{P}}, \underline{\underline{Q}} \rangle \\
\mathrm{in}_i(P) : K &= K\ \mathrm{in}_i(\underline{\underline{P}}) \\
\Lambda\alpha.P : K &= K\ \Lambda\alpha.\underline{\underline{P}} \\
[\![P, t]\!] : K &= K\ [\![\underline{\underline{P}}, t]\!] \\
(P\ Q) : K &= P : (\lambda k.k\ \underline{\underline{Q}}\ K) \\
\pi_i(P) : K &= P : (\lambda k.\pi_i(k)\ K) \\
\mathrm{case}(P,\ x.Q,\ x.R) : K &= P : \lambda k.\mathrm{case}(k,\ x.(Q : K),\ x.(R : K)) \\
(P\ t) : K &= P : (\lambda k.k\ t\ K) \\
(\mathrm{let}\ [\![x, \alpha]\!] = P\ \mathrm{in}\ Q) : K &= P : (\lambda k.\mathrm{let}\ [\![x, \alpha]\!] = k\ \mathrm{in}\ (Q : K)) \\
[P \mid Q_1, \ldots, Q_n] : K &= y\ (P : K)\ \underline{\underline{Q_1}} \cdots \underline{\underline{Q_n}}
\end{aligned}$$

where $y \in \mathbb{Y}$, and $k \notin K$, and such that all occurrences of y in a term $\underline{\underline{M}}$ are distinct variables and that no free variables of P and Q are caught. ♦

We let ':' be loosest bound; in particular we do not bind variables past a ':'. The entry for the translation of $\mathrm{let}\ [\![x, \alpha]\!] = P\ \mathrm{in}\ Q$ can thus be written

$$\mathrm{let}\ [\![x, \alpha]\!] = P\ \mathrm{in}\ Q : K = P : \lambda k.\mathrm{let}\ [\![x, \alpha]\!] = k\ \mathrm{in}\ (Q : K)\ .$$

Definition 7.5 is derived from Definition 7.4 by noting that the translation $\underline{\underline{\bullet}}$ introduces some redexes of the form $\underline{\underline{P}}\ Q \equiv (\lambda k.P')\ Q$. These can be reduced to $P'\{k := Q\}$. They are removed by introducing the second argument K. The second argument holds the value Q that would be substituted for k when reducing the CPS-translated term.

7.1.3 The Well-Typedness of the Translation

In the two previous subsections we have defined a translation from Λ_{M} to $\Lambda_{\mathrm{M}}^{[]}$ and a translation from $\Lambda_{\mathrm{M}}^{[]}$ to Λ_{M}, resp. It is natural to combine the translations into the

following translation of Λ_{M} into Λ_{M}:

$$\begin{array}{lcl}
\underline{\iota(x)} & = & \lambda k.x\ k \\
\underline{\iota(\lambda x.P)} & = & \lambda k.k\ \lambda x.\lambda l.y\ (\underline{\iota(P)}\ l)\lambda m.x\ m \\
\underline{\iota(\langle P,Q\rangle)} & = & \lambda k.k\ \langle \lambda l.y'\ (\underline{\iota(P)}\ l)\ \underline{\iota(Q)}, \lambda l.y''\ (\underline{\iota(Q)}\ l)\ \underline{\iota(P)}\rangle \\
\underline{\iota(\mathrm{in}_i(P))} & = & \lambda k.k\ \mathrm{in}_i(\underline{\iota(P)}) \\
\underline{\iota(\Lambda\alpha.P)} & = & \lambda k.k\ \Lambda\alpha.\underline{\iota(P)} \\
\underline{\iota(\llbracket P,t\rrbracket)} & = & \lambda k.k\ \llbracket\underline{\iota(P)},t\rrbracket \\
\underline{\iota(P\,Q)} & = & \lambda k.\underline{\iota(P)}\ \lambda l.l\ \underline{\iota(Q)}\ k \\
\underline{\iota(\pi_i(P))} & = & \lambda k.\underline{\iota(P)}\ \lambda l.\pi_i(l)\ k \\
\underline{\iota(\mathrm{case}(P,\ x.Q,\ x.R))} & = & \lambda k.\underline{\iota(P)}\ \lambda l.\mathrm{case}(l,\ x.\ k, \\
 & & \qquad\qquad x.(\lambda m.y''\ S'')\ k) \\
\underline{\iota(P\,t)} & = & \lambda k.\underline{\iota(P)}\ \lambda l.l\ t\ k \\
\underline{\iota(\mathrm{let}\ \llbracket x,\alpha\rrbracket = P\ \mathrm{in}\ Q)} & = & \lambda k.\underline{\iota(P)}\ \lambda l.\mathrm{let}\ \llbracket x,\alpha\rrbracket = l\ \mathrm{in}\ (\lambda m.y\ (\underline{\iota(Q)}\ m)\ \lambda n.x\ n)\ k
\end{array} \tag{22}$$

where

$$S' \equiv (\lambda m.y'\ S')\ (\underline{\iota(Q)}\ m)\ (\lambda n.x\ n)\ \lambda o.o\ \lambda x.\underline{\iota(R)}$$

and

$$S'' \equiv (\lambda m.y''\ S'')\ (\underline{\iota(R)}\ m)\ (\lambda n.x\ n)\ \lambda o.o\ \lambda x.\underline{\iota(Q)}\ .$$

This is the map we are looking for. In the following, we will prove that it actually has the property $\underline{\iota(M)} \in \mathrm{WN}_\beta \Rightarrow M \in \mathrm{SN}_{\beta\mathrm{p}}$.

The motivation behind inferring strong normalisation from weak normalisation, is that we often can prove weak normalisation of typed λ-calculi. We should therefore ensure ourself that the image of the combined mapping in (22) is well-typed. We do that in this subsection. In outline, we define a mapping $[\bullet]$ between the types of Λ_{M}. We prove that if term M is typable with type τ, then $\underline{\iota(M)}$ is typable with type $[\tau]$. We conclude that typable terms of Λ_{M} are mapped to typable terms of Λ_{M}.

Meyer and Wand [MW85] make the first account for the translation of types of call-by-value CPS-translations for the simply typed λ-calculus. Sørensen [Sør97] presents a translation for a part of the call-by-name CPS-translations we use here.

Definition 7.6 The maps

$$[\bullet], [\bullet]' : L_{\mathrm{M}} \to L_{\mathrm{M}}$$

are defined by

$$[\tau] = \neg\neg[\tau]'$$

and

$$\begin{array}{lll}
[\bot]' & = \bot & \\
[r^n(t_1,\ldots,t_n)]' & = r^n(t_1,\ldots,t_n) & \\
[\tau\to\sigma]' & = [\tau]\to[\sigma] & (=\neg\neg[\tau]'\to\neg\neg[\sigma]') \\
[\tau\wedge\sigma]' & = [\tau]\wedge[\sigma] & (=\neg\neg[\tau]'\wedge\neg\neg[\sigma]') \\
[\tau\vee\sigma]' & = [\tau]\vee[\sigma] & (=\neg\neg[\tau]'\vee\neg\neg[\sigma]') \\
[\exists\alpha.\tau]' & = \exists\alpha.[\tau] & (=\exists\alpha.\neg\neg[\tau]') \\
[\forall\alpha.\tau]' & = \forall\alpha.[\tau] & (=\forall\alpha.\neg\neg[\tau]')
\end{array}$$

For all $\Gamma \in \Gamma_{\mathrm{M}}$, we define $[\Gamma] = \{x : [\tau] \mid x : \tau \in \Gamma\}$. ♦

Lemma 7.7 *Let $\tau \in L_{\mathrm{M}}$ be a type, α any type variable, and t any eigenterm. Then it holds that $[\tau]\{\alpha := t\} = [\tau\{\alpha := t\}]$.*

Proof. It is intuitively easy to see why the lemma holds: Eigenterms are translated directly to themselves. Hence the translation changes neither the bindings of type variables, nor the substituted eigenterm. Formally, it is proven using induction on the structure of M. We skip the formal proof. □

Proposition 7.8 *Let $M \in \Lambda_{\mathrm{M}}$ be a term, $\psi \in L_{\mathrm{M}}$ a type, and $\Gamma \in \Gamma_{\mathrm{M}}$ a type context such that $\Gamma \vdash M : \psi$, then*

$$\Delta, [\Gamma] \vdash \underline{\iota(M)} : [\psi]$$

for some context $\Delta \in \Gamma_{\mathrm{M}}$ with $\operatorname{dom}\Delta = \mathrm{FV}([M]) \setminus \mathrm{FV}(M)$.

Proof. The only free variables in $\underline{\iota(M)}$ that are not free in M, are the variables y introduced by $\underline{\iota(\bullet)}$. They are the variables in $\operatorname{dom}\Delta$. As each of these variables is distinct and does not occur anywhere else, their types can be chosen as needed. It is therefore implicit in the following proof that Δ is chosen such that each y gets an appropriate type; therefore, we only state below how we choose the domain of Δ. Furthermore, we will write the type derivation for $y\ P_1\ \cdots\ P_n$ as

$$\frac{\Gamma \vdash P_1 : \tau_1 \quad \cdots \quad \Gamma \vdash P_1 : \tau_n}{\Gamma \vdash y\ P_1\ \cdots\ P_n : \sigma}$$

where we collapse the applications of $\to E$ in the type derivation.

First, we do some small type derivations that we will use several places below.

$$\dfrac{\dfrac{\Xi \vdash x : \neg\neg[\tau]' \quad \Xi \vdash k : \neg[\tau]'}{\Xi = x : [\tau], k : \neg[\tau] \vdash x\ k : \bot}}{x : [\tau] \vdash \lambda k.x\ k : [\tau] = \neg\neg[\tau]'} \quad (*)$$

This can also be stated as $x : [\tau] \vdash \underline{x} : [\tau]$. Using weakening, Lemma 4.6, we can apply this derivation in any context where x has type $[\tau]$. We also note that, if $\Theta \vdash \underline{\iota(S)} : [\tau]$ and $k \notin \operatorname{dom}\theta$, then

$$\frac{\Theta, k : \neg[\tau]' \vdash \underline{\iota(S)} : [\tau] = \neg\neg[\tau]' \quad \Theta, k : \neg[\tau]' \vdash k : \neg[\tau]'}{\Theta, k : \neg[\tau]' \vdash \underline{\iota(S)}\ k : \bot} \quad (\star)$$

We now turn to the proof for the proposition. We use induction on the derivation of $\Gamma \vdash M : \tau$ and make extensively use of weakening.

i. $\Gamma, x : \tau \vdash x : \tau$. We have $\underline{\iota(M)} \equiv \lambda k.x\ k$ and $[\psi] \equiv [\tau]$. We choose $\Delta = \emptyset$ and using weakening on $(*)$ we have $[\Gamma], x : [\tau] \vdash \underline{x} : [\tau]$.

ii. $\Gamma \vdash \lambda x.P : \tau \to \sigma$. We have $\underline{\iota(M)} \equiv \lambda k.k\ \lambda x.\lambda l.y\ (\underline{\iota(P)}\ l)\underline{x}$ and $[\psi] \equiv \neg\neg([\tau] \to [\sigma])$. By the induction hypothesis, $\Delta_P, [\Gamma], x : [\tau] \vdash \underline{\iota(P)} : [\sigma]$ where we recall that $[\sigma] \equiv \neg\neg[\sigma]'$. We choose the type context Δ such that $\operatorname{dom}\Delta = \{y\} \cup \operatorname{dom}\Delta_P$. We use this in the following type derivation:

$$\dfrac{\dfrac{\Xi \vdash k : \neg([\tau] \to [\sigma])}{\Xi = \Delta, [\Gamma], k : \neg([\tau] \to [\sigma]) \vdash k\ \lambda x.\lambda l.y\ (\underline{\iota(P)}\ l)\ \underline{x} : \bot}}{\Delta, [\Gamma] \vdash \lambda k.k\ \lambda x.\lambda l.y\ (\underline{\iota(P)}\ l)\ \underline{x} : \neg\neg([\tau] \to [\sigma])} \qquad \dfrac{\dfrac{(\star)\ \dfrac{}{\Xi' \vdash \underline{\iota(P)}\ l : \bot} \quad (*)\ \dfrac{}{\Xi' \vdash \lambda m.x\ m : [\tau]}}{\Xi' = \Xi, x : [\tau], l : \neg[\sigma]' \vdash y\ (\underline{\iota(P)}\ l)\ \underline{x} : \bot}}{\Xi \vdash \lambda x.\lambda l.y\ (\underline{\iota(P)}\ l)\ \underline{x} : [\tau] \to [\sigma]}$$

iii. $\Gamma \vdash \langle P, Q\rangle : \tau \wedge \sigma$. We have $\underline{\iota(M)} \equiv \lambda k.k\,\langle R', R''\rangle$ where $R' \equiv \lambda l'.y'\,(\underline{\iota(P)}\,l')\,\underline{\iota(Q)}$ and $R'' \equiv \lambda l''.y''\,(\underline{\iota(Q)}\,l'')\,\underline{\iota(P)}$. Furthermore, we have $[\psi] = \neg\neg([\tau] \wedge [\sigma])$. By the induction hypothesis, $\Delta_P, [\Gamma] \vdash \underline{\iota(P)} : [\tau]$ and $\Delta_Q, [\Gamma] \vdash \underline{\iota(Q)} : [\sigma]$. We recall that $[\tau] \equiv \neg\neg[\tau]'$ and $[\sigma] \equiv \neg\neg[\sigma]'$. We choose the type context Δ such that $\operatorname{dom}\Delta = \{y', y''\} \cup \operatorname{dom}\Delta_P \cup \operatorname{dom}\Delta_Q$. We can do the type derivation

$$\dfrac{\dfrac{\overset{(\star)}{\dfrac{}{\Xi \vdash \underline{\iota(P)}\,l' : \bot}} \qquad \overset{\text{IH}}{\dfrac{}{\Xi \vdash \underline{\iota(Q)} : [\sigma]}}}{\Xi = \Delta, [\Gamma], l' : \neg[\tau]' \vdash y'(\underline{\iota(P)}\,l')\,\underline{\iota(Q)} : \bot}}{\Delta, [\Gamma] \vdash \lambda l'.y'\,(\underline{\iota(P)}\,l')\,\underline{\iota(Q)} : [\tau]}$$

and similarly $\Delta, [\Gamma] \vdash \lambda l''.y''\,(\underline{\iota(Q)}\,l'')\,\underline{\iota(P)} : [\sigma]$. We conclude that

$$\dfrac{\dfrac{\dfrac{}{\Xi \vdash k : \neg([\tau] \wedge [\sigma])} \qquad \dfrac{\Xi \vdash R' : [\tau] \quad \Xi \vdash R'' : [\sigma]}{\Xi \vdash \langle R', R''\rangle : [\tau] \wedge [\sigma]}}{\Xi = \Delta, [\Gamma], k : \neg([\tau] \wedge [\sigma]) \vdash k\,\langle R', R''\rangle : \bot}}{\Delta, [\Gamma] \vdash \lambda k.k\,\langle R', R''\rangle : \neg\neg([\tau] \wedge [\sigma])}$$

iv. $\Gamma \vdash \mathrm{in}_i(P) : \tau_1 \vee \tau_2$, or $\Gamma \vdash \Lambda\alpha.P : \forall\alpha.\tau$, or $\Gamma \vdash \{\!|P, t|\!\} : \exists\alpha.\tau$. These three cases are similar only differing in the connective or quantifier in the type derivation. We illustrate the derivation by $\Gamma \vdash \mathrm{in}_i(P) : \tau_1 \vee \tau_2$. In this case, we have $\underline{\iota(M)} \equiv \lambda k.k\,\mathrm{in}_i(\underline{\iota(P)})$ and $[\psi] \equiv [\tau_1 \vee \tau_2] \equiv \neg\neg([\tau_1] \vee [\tau_2])$. The induction hypothesis gives us $\Delta, [\Gamma] \vdash \underline{\iota(P)} : [\tau_i]$. We do the following type derivation

$$\dfrac{\dfrac{\dfrac{}{\Xi \vdash k : \neg([\tau_1] \vee [\tau_2])} \qquad \dfrac{\overset{\text{IH}}{\dfrac{}{\Xi \vdash \underline{\iota(P)} : [\tau_i]}}}{\Xi \vdash \mathrm{in}_i(\underline{\iota(P)}) : [\tau_1] \vee [\tau_2]}}{\Xi = \Delta, [\Gamma], k : \neg([\tau_1] \vee [\tau_2]) \vdash k\,\mathrm{in}_i(\underline{\iota(P)}) : \bot}}{\Delta, [\Gamma] \vdash \lambda k.k\,\mathrm{in}_i(\underline{\iota(P)}) : \neg\neg([\tau_1] \vee [\tau_2])}$$

In the case $M \equiv \Lambda\alpha.P$ we replace the type $[\tau_1] \vee [\tau_2]$ by $\forall\alpha.[\tau]$ and the type $[\tau_i]$ by $[\tau]$; in the case $M \equiv \{\!|P, t|\!\}$ we replace the type $[\tau_1] \vee [\tau_2]$ by $\exists\alpha.[\tau]$ and the type $[\tau_i]$ by $[\tau]\{\alpha := t\} \equiv [\tau\{\alpha := t\}]$.

v. $\Gamma \vdash P\,Q : \sigma$ or $\Gamma \vdash P\,t : \sigma$. These two cases are similar; the only difference is that the latter has t at the place of $\underline{\iota(Q)}$ so a second application of the induction hypothesis is unnecessary. We illustrate their proof by $\Gamma \vdash P\,Q : \sigma$, the most complicated of the two cases. We have $\underline{\iota(M)} \equiv \lambda k.\underline{\iota(P)}\,\lambda l.l\,\underline{\iota(Q)}\,k$ and $[\psi] \equiv \neg\neg[\sigma]'$. By the induction hypothesis, we have $\Delta_P, [\Gamma] \vdash \underline{\iota(P)} : [\tau \to \sigma]$ and $\Delta_Q, [\Gamma] \vdash \underline{\iota(Q)} : [\tau]$. We recall that $[\tau \to \sigma] \equiv \neg\neg[\tau \to \sigma]'$ and $[\tau] \equiv \neg\neg[\tau]'$. We choose Δ such that $\operatorname{dom}\Delta = \operatorname{dom}\Delta_P \cup \operatorname{dom}\Delta_Q$. First, we note that for any Θ where $\Theta \supseteq \Delta \cup [\Gamma]$ and $k \notin \operatorname{dom}\Theta$ we can do the type derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Xi' \vdash l : [\tau \to \sigma]'} \qquad \overset{\text{IH}}{\dfrac{}{\Xi' \vdash \underline{\iota(Q)} : \neg\neg[\tau]'}}}{\Xi' \vdash l\,\underline{\iota(Q)} : \neg\neg[\sigma]'} \qquad \dfrac{}{\Xi' \vdash k : \neg[\sigma]'}}{\Xi' = \Xi, l : [\tau \to \sigma]' \vdash l\,\underline{\iota(Q)}\,k : \bot}}{\Xi = \Theta, k : \neg[\sigma]' \vdash \lambda l.l\,\underline{\iota(Q)}\,k : \neg[\tau \to \sigma]'}$$

From this we conclude

$$\dfrac{\dfrac{\overset{\text{IH}}{\dfrac{}{\Xi \vdash \underline{\iota(P)} : \neg\neg[\tau \to \sigma]'}} \qquad \Xi \vdash \lambda l.l\,\underline{\iota(Q)}\,k : \neg[\tau \to \sigma]'}{\Xi = \Delta, [\Gamma], k : \neg[\sigma]' \vdash \underline{\iota(P)}\,\lambda l.l\,\underline{\iota(Q)}\,k : \bot}}{\Delta, [\Gamma] \vdash \lambda k.\underline{\iota(P)}\,\lambda l.l\,\underline{\iota(Q)}\,k : \neg\neg[\sigma]'}$$

In the other case $[\tau \to \sigma]'$ is replaced by $[\forall\alpha.\tau]' \equiv \forall\alpha.\neg\neg[\tau]'$ and we derive $\Xi' \vdash l\,t : (\neg\neg[\tau]')\{\alpha := t\}$ instead of $\Xi' \vdash l\,\underline{\iota(Q)} : \neg\neg[\sigma]'$. Using Lemma 7.7, we find the equality $(\neg\neg[\tau]')\{\alpha := t\} \equiv \neg\neg[\tau\{\alpha := t\}]'$, which concludes the case where $M \equiv P\,t$.

vi. $\Gamma \vdash \pi_i(P) : \tau_i$. We have $\underline{\iota(M)} \equiv \lambda k.\underline{\iota(P)}\ \lambda l.\pi_i(l)\ k$ and $[\psi]$ where we recall that $[\psi] \equiv \neg\neg[\tau_i]'$. Using the induction hypothesis we get $\Delta, [\Gamma] \vdash \underline{\iota(P)} : [\tau_1 \wedge \tau_2]$ where we note that $[\tau_1 \wedge \tau_2] \equiv \neg\neg([\tau_1] \wedge [\tau_2])$. We note that for any Θ where $\Theta \supseteq \Delta \cup [\Gamma]$ and $k \notin \operatorname{dom}\Theta$ we can do the type derivation

$$\dfrac{\dfrac{\dfrac{}{\Xi \vdash l : \neg\neg[\tau_1]' \wedge \neg\neg[\tau_2]'}}{\Xi \vdash \pi_i(l) : \neg\neg[\tau_i]'} \qquad \dfrac{}{\Xi \vdash k : \neg[\tau_i]'}}{\dfrac{\Xi = \Theta, k : \neg[\tau_i]', l : [\tau_1] \wedge [\tau_2] \vdash \pi_i(l)\ k : \bot}{\Theta, k : \neg[\tau_i]' \vdash \lambda l.\pi_i(l)\ k : \neg([\tau_1] \wedge [\tau_2])}}$$

which we use to conclude

$$\dfrac{\dfrac{\text{IH}\ \dfrac{}{\Xi \vdash \underline{\iota(P)} : \neg\neg([\tau_1] \wedge [\tau_2])} \qquad \Xi \vdash \lambda l.\pi_i(l)\ k : \neg([\tau_1] \wedge [\tau_2])}{\Xi = \Delta, [\Gamma], k : \neg[\tau_i]' \vdash \underline{\iota(P)}\ \lambda l.\pi_i(l)\ k : \bot}}{\Delta, [\Gamma] \vdash \lambda k.\underline{\iota(P)}\ \lambda l.\pi_i(l)\ k : \neg\neg[\tau_i]'}$$

vii. $\Gamma \vdash \operatorname{case}(P,\ x.Q,\ x.R) : \tau$. We have $\underline{\iota(M)} \equiv \lambda k.\underline{\iota(P)}\ \lambda l.\operatorname{case}(l,\ x.S,\ x.T)$ where

$$S \equiv \left(\lambda m.y'\ (\underline{\iota(Q)}\ m)\ \underline{x}\ (\lambda o.o\ \lambda x.\underline{\iota(R)})\right)\ k$$

and

$$T \equiv \left(\lambda m.y''\ (\underline{\iota(R)}\ m)\ \underline{x}\ (\lambda o.o\ \lambda x.\underline{\iota(Q)})\right)\ k\ .$$

Furthermore, $[\psi] \equiv \neg\neg[\tau]'$. Using the induction hypothesis we have type derivations for $\Delta_P, [\Gamma] \vdash \underline{\iota(P)} : [\sigma \vee \vartheta]$, and $\Delta_Q, [\Gamma], x : [\sigma] \vdash \underline{\iota(Q)} : [\tau]$, and also $\Delta_R, [\Gamma], x : [\vartheta] \vdash \underline{\iota(R)} : [\tau]$. We choose the type context Δ such that we have $\operatorname{dom}\Delta = \{y', y''\} \cup \operatorname{dom}\Delta_P \cup \operatorname{dom}\Delta_Q \cup \operatorname{dom}\Delta_R$. We first note that for any type context $\Theta \supseteq \Delta \cup [\Gamma]$ we have

$$\dfrac{\dfrac{}{\Xi \vdash o : \neg([\vartheta] \to [\tau])} \qquad \dfrac{\text{IH}\ \dfrac{}{\Xi, x : [\vartheta] \vdash \underline{\iota(R)} : [\tau]}}{\Xi \vdash \lambda x.\underline{\iota(R)} : [\vartheta] \to [\tau]}}{\dfrac{\Xi = \Theta, o : \neg([\vartheta] \to [\tau]) \vdash o\ \lambda x.\underline{\iota(R)} : \bot}{\Theta \vdash \lambda o.o\ \lambda x.\underline{\iota(R)} : [\vartheta \to \tau]}}$$

where we have exploited that $[\vartheta \to \tau] \equiv \neg\neg([\vartheta] \to [\tau])$. We can do a similar derivation of $\Theta \vdash \lambda o.o\ \lambda x.\underline{\iota(Q)} : [\sigma \to \tau]$. Furthermore, we note that for any $\Theta \supseteq \Delta \cup [\Gamma]$ and $k, x \notin \operatorname{dom}\Theta$.

$$\dfrac{\dfrac{\dfrac{(\star)\ \dfrac{}{\Xi \vdash \underline{\iota(Q)}, m : \bot} \qquad (*)\ \dfrac{}{\Xi \vdash \underline{x} : \bot} \qquad \dfrac{}{\Xi \vdash \underline{\lambda x.\iota(R)} : [\vartheta \to \tau]}}{\Xi' = \Xi, m : \neg[\tau]' \vdash y'\ (\underline{\iota(Q)}\ m)\ \underline{x}\ \underline{\lambda x.\iota(R)} : \bot}}{\Xi \vdash \lambda m.y'\ (\underline{\iota(Q)}\ m)\ \underline{x}\ \underline{\lambda x.\iota(R)} : \neg\neg[\tau]'} \qquad \dfrac{}{\Xi \vdash k : \neg[\tau]'}}{\Xi = \Theta, k : \neg[\tau]', x : [\sigma] \vdash \ \lambda m.y'\ (\underline{\iota(Q)}\ m)\ \underline{x}\ \underline{\lambda x.\iota(R)}\ \ k : \bot}$$

Similarly, $\Theta, k : \neg[\tau], x : [\vartheta] \vdash \left(\lambda m.y''\ (\underline{\iota(R)}\ m)\ \underline{x}\ \underline{\lambda x.\iota(Q)}\right)\ k : \bot$. Fitting these pieces together we get the following type derivation for any $\Theta \supseteq \Delta \cup [\Gamma]$ where $k \notin \operatorname{dom}\Theta$:

$$\dfrac{\dfrac{\dfrac{}{\Xi' \vdash l : [\sigma] \vee [\vartheta]} \qquad \dfrac{}{\Xi', x : [\sigma] \vdash S : \bot} \qquad \dfrac{}{\Xi', x : [\vartheta] \vdash T : \bot}}{\Xi' = \Xi, l : [\sigma] \vee [\vartheta] \vdash \operatorname{case}(l,\ x.S,\ x.T) : \bot}}{\Xi = \Theta, k : \neg[\tau]' \vdash \lambda l.\operatorname{case}(l,\ x.S,\ x.T) : \neg([\sigma] \vee [\vartheta])}$$

Using this derivation we conclude

$$\dfrac{\dfrac{\text{IH}\ \dfrac{}{\Xi \vdash \underline{\iota(P)} : \neg\neg([\sigma] \vee [\vartheta])} \qquad \Xi \vdash \lambda l.\operatorname{case}(l,\ x.S,\ x.T) : \neg([\sigma] \vee [\vartheta])}{\Xi = \Delta, [\Gamma], k : \neg[\tau]' \vdash \underline{\iota(P)}\ \lambda l.\operatorname{case}(l,\ x.S,\ x.T) : \bot}}{\Delta, [\Gamma] \vdash \lambda k.\underline{\iota(P)}\ \lambda l.\operatorname{case}(l,\ x.S,\ x.T) : \neg\neg[\tau]'}$$

viii. $\Gamma \vdash \mathrm{let}\,\llbracket x, \alpha \rrbracket = P \,\mathrm{in}\, Q : \tau$. We have $\underline{\iota(M)} \equiv \lambda k.\underline{\iota(P)}\ \lambda l.\mathrm{let}\,\llbracket x, \alpha \rrbracket = l \,\mathrm{in}\, Q$ where $Q \equiv (\lambda m.y\ (\underline{\iota(Q)}\ m)\ \underline{x})\ k$ and $[\psi] \equiv \neg\neg[\tau]'$. By the induction hypothesis, we have $\Delta_P, [\Gamma] \vdash \underline{\iota(P)} : [\exists\alpha.\sigma]$ where we recall that $[\exists\alpha.\sigma] \equiv \neg\neg\exists\alpha.[\sigma]$. We also see that $\Delta_Q, [\Gamma], x : [\sigma] \vdash \underline{\iota(Q)} : [\tau]$. We choose the type context Δ such that we have $\mathrm{dom}\,\Delta = \{y\} \cup \mathrm{dom}\,\Delta_P \cup \mathrm{dom}\,\Delta_Q$. We first note that we have

$$\dfrac{\dfrac{\dfrac{\overset{(*)}{\overline{\Xi \vdash \underline{\iota(Q)}\ m : \bot}} \quad \overset{(*)}{\overline{\Xi \vdash \underline{x} : [\sigma]}}}{\Xi' = \Xi, m : \neg[\tau]' \vdash y'\ (\underline{\iota(Q)}\ m)\ \underline{x} : \bot}}{\Xi \vdash \lambda m.y'\ (\underline{\iota(Q)}\ m)\ \underline{x} : \neg\neg[\tau]'} \quad \dfrac{}{\Xi \vdash k : \neg[\tau]'}}{\Xi = \Theta, k : \neg[\tau]', x : [\sigma] \vdash \big(\lambda m.y'\ (\underline{\iota(Q)}\ m)\ \underline{x}\big)\ k : \bot}$$

for any $\Theta \supseteq \Delta \cup \Gamma$ where $x, k \notin \mathrm{dom}\,\Theta$. We note that for any $\Theta \supseteq \Delta \cup [\Gamma]$ where $k \notin \mathrm{dom}\,\Theta$ we can do the type derivation

$$\dfrac{\dfrac{\overline{\Xi' \vdash l : \exists\alpha.[\sigma]} \quad \overline{\Xi', x : [\sigma] \vdash Q' : \bot}}{\Xi' = \Xi, l : \exists\alpha.[\sigma] \vdash \mathrm{let}\,\llbracket x, \alpha \rrbracket = l \,\mathrm{in}\, Q' : \bot}}{\Xi = \Theta, k : \neg[\tau]' \vdash \lambda l.\mathrm{let}\,\llbracket x, \alpha \rrbracket = l \,\mathrm{in}\, Q' : \neg\exists\alpha.[\sigma]}$$

and from this we conclude

$$\dfrac{\dfrac{\overset{\mathrm{IH}}{\overline{\Xi \vdash \underline{\iota(P)} : \neg\neg\exists\alpha.[\sigma]}} \quad \Theta, k : \neg[\tau]' \vdash \lambda l.\mathrm{let}\,\llbracket x, \alpha \rrbracket = l \,\mathrm{in}\, Q' : \neg\exists\alpha.[\sigma]}{\Xi = \Delta, [\Gamma], k : \neg[\tau]' \vdash \underline{\iota(P)}\ \lambda l.\mathrm{let}\,\llbracket x, \alpha \rrbracket = l \,\mathrm{in}\, Q' : \bot}}{\Delta, [\Gamma] \vdash \lambda k.\underline{\iota(P)}\ \lambda l.\mathrm{let}\,\llbracket x, \alpha \rrbracket = l \,\mathrm{in}\, Q' : \neg\neg[\tau]'}$$

This exhausts the possibilities and concludes the proof that typability is conserved under the translation $\underline{\iota(\bullet)}$. □

We conclude this section with the following corollary, which was what we were looking for.

Corollary 7.9 *Let $M \in \Lambda_{\mathrm{M}}$ be a typable term. Then $\underline{\iota(M)} \in \Lambda_{\mathrm{M}}$ is a typable term.*

7.2 The Image of $\underline{\iota(\bullet)}$ is Uniformly Normalising

At this point we have a translation of Λ_{M}-terms to other Λ_{M}-terms. We will now start establishing that the image of this translation has the properties we are looking for. The first step is to prove that the translated terms are uniformly normalising. In the previous chapter we found a uniformly normalising subset: the $\mathbb{Y}$-good typable $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms. As the typability of terms in the image of $\underline{\iota(\bullet)}$ was established in Corollary 7.9, it is sufficient to establish that the image of $\underline{\iota(\bullet)}$ is in the set of $\mathbb{Y}$-good $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-terms. This is the achievement of this section.

Lemma 7.10 *Let $M \in \Lambda_{\mathrm{M}}^{[]}$ be a term. All pairs $\langle S_1, S_2\rangle \subseteq \underline{\iota(M)}$ are $\mathbb{Y}$-good wrt. to the set of variables $\mathbb{Y}$ of the $\underline{\bullet}$-translation.*

Proof. We use induction over the structure of M:

i. $M \equiv x$. We have $\underline{\iota(M)} \overset{22}{\equiv} \lambda k.x\ k$. There is no $\langle S_1, S_2\rangle \subseteq \underline{\iota(M)}$.

ii. $M \equiv \lambda x.P$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$, or $M \equiv P\,Q$, or $M \equiv \pi_i(P)$, or $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$, $M \equiv P\,t$ or $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$. The cases are similar as we for any pair $\langle S, T\rangle \subseteq \underline{\iota(M)}$ have $\langle S, T\rangle \subseteq \underline{\iota(P)}$, or $\langle S, T\rangle \subseteq \underline{\iota(Q)}$, or $\langle S, T\rangle \subseteq \underline{\iota(R)}$. The $\mathbb{Y}$-goodness of the pair follows from the induction hypothesis.

iii. $M \equiv \langle P, Q\rangle$. We have $\underline{\iota(M)} \stackrel{22}{=} \lambda k.k\,\langle R_1, R_2\rangle$ where $R_1 \equiv \lambda l'.y\,(\underline{\iota(P)}\,l')\,\underline{\iota(Q)}$ and $R_2 \equiv \lambda l''.y\,(\underline{\iota(Q)}\,l'')\,\underline{\iota(P)}$

We consider any pair $\langle S, T\rangle \subseteq \underline{\iota(M)}$. We have the the following three possibilities:

(a) $\langle S, T\rangle \subseteq \underline{\iota(P)}$ or $\langle S, T\rangle \subseteq \underline{\iota(Q)}$. Then $\langle S, T\rangle$ is $\mathbb{Y}$-good by the induction hypothesis.

(b) $\langle S, T\rangle \equiv \langle R_1, R_2\rangle$. We should prove that $R_1\theta \in \infty_\beta \iff R_2\theta \in \infty_\beta$ for all $\mathbb{Y}$-neutral substitutions θ. We consider any $\mathbb{Y}$-good substitution θ.

We assume that $R_1\theta \in \infty_\beta$. We have the following possibilities

i. $\underline{\iota(P)}\theta \in \infty_\beta$ or $\underline{\iota(Q)}\theta \in \infty_\beta$. Clearly, we also have $R_2\theta \in \infty_\beta$.

ii. $\underline{\iota(P)}\theta\,l' \in \infty_\beta$. We see from Definition 7.4 that $\underline{\iota(P)} \equiv \lambda k'.P'$ for some P'. We therefore have

$$(\underline{\iota(P)}\theta)\,l' \equiv (\lambda k'.P')\,l' \twoheadrightarrow_\beta (\lambda k'.P'')\,l' \to_\beta P''\{k' := l'\} \in \infty_\beta\ .$$

We see that we only do a variable renaming in P'. Therefore, we have $P' \in \infty_\beta$ and also $\underline{\iota(P)}\theta \in \infty_\beta$. We conclude $R_2\theta \in \infty_\beta$.

Thus $R_1\theta \in \infty_\beta$ implies $R_2\theta \in \infty_\beta$. The proof that $R_2\theta \in \infty_\beta$ implies $R_1\theta \in \infty_\beta$ is the same with R_1 and R_2 interchanged.

This exhausts the possibilities and concludes the proof that all pairs of $\underline{\iota(M)}$ are $\mathbb{Y}$-good □

Lemma 7.11 *Let $M \in \Lambda^{[]}_{\mathrm{M}}$ be a term. All* case*-expressions* $\mathrm{case}(S,\, z.T,\, z.U) \subseteq \underline{\iota(M)}$ *are $\mathbb{Y}$-good wrt. to the set of variables $\mathbb{Y}$ of the $\underline{\bullet}$-translation.*

Proof. We will prove that for any case-expression $\mathrm{case}(T,\, z.U,\, z.V) \subseteq \underline{\iota(M)}$ we have

$$U\theta \in \infty_\beta \iff V\theta \in \infty_\beta$$

for any $\mathbb{Y}$-neutral substitution θ. We use induction on the structure of M:

i. $M \equiv x$. We have $\underline{\iota(M)} \equiv \lambda k.x\,k$ and there is no $\mathrm{case}(T,\, z.U,\, z.V) \subseteq \underline{\iota(M)}$. Thus the proposition is trivially true.

ii. $M \equiv \lambda x.P$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$, or $M \equiv P\,Q$, or $M \equiv \pi_i(P)$, or $M \equiv \langle P, Q\rangle$, $M \equiv P\,t$ or $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$. The cases are similar; for any case-expression $\mathrm{case}(S,\, z.T,\, z.U) \subseteq \underline{\iota(M)}$ we have either $\mathrm{case}(S,\, z.T,\, z.U) \subseteq \underline{\iota(P)}$, or $\mathrm{case}(T,\, z.U,\, z.V) \subseteq \underline{\iota(Q)}$, or $\mathrm{case}(S,\, z.T,\, z.U) \subseteq \underline{\iota(R)}$. The case-expression is $\mathbb{Y}$-good by the induction hypothesis.

iii. $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$. We have $\underline{\iota(M)} \equiv \lambda k.\underline{\iota(P)}\,\lambda l.\mathrm{case}(l,\, x.S_1,\, x.S_2)$ where

$$S_1 \equiv \big(\lambda m.y'\,(\underline{\iota(Q)}\,m)\,\underline{x}\,\underline{\lambda x.\iota(R)}\big)\,k$$

and

$$S_2 \equiv \big(\lambda m.y'\,(\underline{\iota(R)}\,m)\,\underline{x}\,\underline{\lambda x.\iota(Q)}\big)\,k\ .$$

We consider any case-expression $\mathrm{case}(T,\, z.U,\, z.V) \subseteq \underline{\iota(M)}$ and split into two subcases:

(a) $\mathrm{case}(T,\, z.U,\, z.V) \subseteq \underline{\iota(P)}$, or $\mathrm{case}(T,\, z.U,\, z.V) \subseteq \underline{\iota(Q)}$, or $\mathrm{case}(T,\, z.U,\, z.V) \subseteq \underline{\iota(R)}$. The $\mathbb{Y}$-goodness of the case-expression follows from the induction hypothesis.

(b) $\mathrm{case}(T,\, z.U,\, z.V) \equiv \mathrm{case}(l,\, x.S_1,\, x.S_2)$. We will prove that for any $\mathbb{Y}$-neutral substitution θ we have

$$U\theta \in \infty_\beta \iff V\theta \in \infty_\beta \ .$$

We only consider the "$\Rightarrow$"-direction as the two directions are symmetric. We can have $U\theta \in \infty_\beta$ in the following ways:

i. $\underline{\iota(Q)}\theta \in \infty_\beta$ or $\underline{\iota(R)}\theta \in \infty_\beta$. We have $V\theta \in \infty_\beta$ as $\underline{\iota(Q)} \subseteq V$ and $\underline{\iota(R)} \subseteq V$, resp.

ii. $\underline{\iota(Q)}\, m \in \infty_\beta$. As $\underline{\iota(Q)} \equiv \lambda p.Q'$ for some term Q', we have the following reduction path:

$$\underline{\iota(Q)}\, m \equiv (\lambda p.Q')\, m \twoheadrightarrow_\beta (\lambda p.Q'')\, m \to_\beta Q''\{p := m\} \in \infty_\beta \ .$$

Thus, we only do a variable renaming when reducing the abstraction $\lambda p.Q''$. Therefore $Q'' \in \infty_\beta$ and hence also $\underline{\iota(Q)} \in \infty_\beta$. As $\underline{\iota(Q)} \subseteq V$, we conclude that $V\theta \in \infty_\beta$.

iii. $S_1\theta \in \infty_\beta$. We have one of the following two cases:

A. We have the reduction path

$$\begin{aligned}\big(\big(\lambda m.y'\ (\underline{\iota(Q)}\, m)\ \underline{x}\ (\lambda o.o\ \lambda x.R)\big)\ k\big)\theta \\ &\to_\beta \big(\lambda m.y'\ (Q'\ m)\ \underline{x}\ (\lambda o.o\ \lambda x.R')\big)\ k \\ &\to_\beta y'\ (Q'\ k)\ \underline{x}\ (\lambda o.o\ \lambda x.R') \\ &\in \infty_\beta\end{aligned}$$

We have $Q'\ k \in \infty_\beta$ or $R' \in \infty_\beta$; both cases are covered above.

B. We have the reduction path

$$\begin{aligned}\big(\big(\lambda m.y'\ (\underline{\iota(Q)}\, m)\ \underline{x}\ (\lambda o.o\ \lambda x.R)\big)\ k\big)\theta \\ &\twoheadrightarrow_\beta \big(\lambda m.y'\ ((\lambda p.Q')\ m)\ \underline{x}\ (\lambda o.o\ \lambda x.R')\big)\ k \\ &\to_\beta y'\ ((\lambda p.Q')\ k)\ \underline{x}\ (\lambda o.o\ \lambda x.R') \\ &\twoheadrightarrow_\beta y'\ ((\lambda p.Q'')\ k)\ \underline{x}\ (\lambda o.o\ \lambda x.R'') \\ &\to_\beta y'\ (Q''\{p := k\})\ \underline{x}\ (\lambda o.o\ \lambda x.R'') \\ &\in \infty_\beta\end{aligned}$$

Then either $Q''\{p := k\} \in \infty_\beta$ or $R'' \in \infty_\beta$; both cases are covered above.

This exhausts the possibilities and concludes the proof that all case-expressions of $\underline{\iota(M)}$ are $\mathbb{Y}$-good. □

Lemma 7.12 *Let $M \in \Lambda_{\mathrm{M}}^{[]}$ be a term. Then $\underline{\iota(M)}$ is $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term wrt. to the set of variables $\mathbb{Y}$ of the $\underline{\bullet}$-translation.*

Proof. For this proof we recall the notion of I-friendliness used in the proof of Proposition 4.14: A term is I-friendly if it is either

i. One of the four forms mentioned in Definition 4.11 of $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$ and fulfils the conditions for the form, or

ii. Any other term.

We will prove that any subterm is I-friendly; we use induction on the structure of M:

i. $M \equiv x$. We have $\underline{\iota(M)} \equiv \lambda k.x\,k$. We see that $k \in \mathrm{FV}(k)$ and therefore that $\underline{\iota(\in)}\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$.

ii. $M \equiv \lambda x.P$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P,t]\!]$, or $M \equiv P\,Q$, or $M \equiv \pi_i(P)$, or $M \equiv P\,t$ or $M \equiv \mathrm{let}\,[\![x,\alpha]\!] = P\,\mathrm{in}\,Q$. These cases are similar. All subterms are I-friendly by the induction hypothesis. Furthermore, we note by simple inspection of (22) that all the abstraction variables appear free in the body.

iii. $M \equiv \langle P,Q\rangle$. We have $\underline{\iota(M)} \equiv \lambda k.k\,\langle\lambda l.y'\,(\underline{\iota(P)}\,l)\,\underline{\iota(Q)}, \lambda l.y''\,(\underline{\iota(Q)}\,l)\,\underline{\iota(P)}\rangle$. We see that the abstraction variables k and l appear free in the body. Furthermore, all subterms of $\underline{\iota(P)}$ or $\underline{\iota(Q)}$ are I-friendly. We should therefore only prove that the pair in this translation is I-friendly. We have

$$\begin{aligned}\mathrm{FV}(\lambda l.y'\,(\underline{\iota(P)}\,l)\,\underline{\iota(Q)}) \setminus \mathbb{Y} &= \big(\{y'\} \cup \mathrm{FV}(\underline{\iota(P)}) \cup \mathrm{FV}(\underline{\iota(Q)})\big) \setminus \mathbb{Y} \\ &= \mathrm{FV}(\underline{\iota(P)}) \cup \mathrm{FV}(\underline{\iota(Q)}) \\ &= \big(\{y''\} \cup \mathrm{FV}(\underline{\iota(P)}) \cup \mathrm{FV}(\underline{\iota(Q)})\big) \setminus \mathbb{Y} \\ &= \mathrm{FV}(\lambda l.y''\,(\underline{\iota(P)}\,l)\,\underline{\iota(Q)}) \setminus \mathbb{Y}\ .\end{aligned}$$

We conclude that all subterms are I-friendly, and hence that $\underline{\iota(\langle P,Q\rangle)}$ is I-friendly.

iv. $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$. We have $\underline{\iota(M)} \equiv \lambda k.\underline{\iota(P)}\,\lambda l.\mathrm{case}(l,\, x.S_1,\, x.S_2)$ where

$$S_1 \equiv (\lambda m.y'\,(\underline{\iota(Q)}\,m)\,(\lambda n.x\,n)\,\lambda o.o\,\lambda x.\underline{\iota(R)})\,k$$

and

$$S_2 \equiv (\lambda m.y''\,(\underline{\iota(R)}\,m)\,(\lambda n.x\,n)\,\lambda o.o\,\lambda x.\underline{\iota(Q)})\,k\ .$$

All subterms are I-friendly by the induction hypothesis. Furthermore, we see that the abstraction variables k, l, m, n, o, and x all appear in the body. The case-expression fulfils the conditions as

$$\begin{aligned}\mathrm{FV}(S_1) \setminus \mathbb{Y} &= \big(\{y',x,k\} \cup \mathrm{FV}(\underline{\iota(Q)}) \cup \mathrm{FV}(\underline{\iota(R)})\big) \setminus \mathbb{Y} \\ &= \{x,k\} \cup \mathrm{FV}(\underline{\iota(Q)}) \cup \mathrm{FV}(\underline{\iota(R)}) \\ &= \big(\{y'',x,k\} \cup \mathrm{FV}(\underline{\iota(Q)}) \cup \{x\} \cup \mathrm{FV}(\underline{\iota(R)})\big) \setminus \mathbb{Y} \\ &= \mathrm{FV}(S_2) \setminus \mathbb{Y}\ .\end{aligned}$$

We conclude that all subterms are I-friendly, and hence that $\underline{\iota(\mathrm{case}(P,\, x.Q,\, x.R))}$ is I-friendly.

This exhausts the possibilities and concludes the proof that $\underline{\iota(M)} \in \Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$. □

Proposition 7.13 *Let $M \in \Lambda_{\mathrm{M}}$ be a term. Then $\underline{\iota(M)} \in \mathrm{UN}_\beta$.*

Proof. By the three previous lemmata, $\underline{\iota(M)}$ is $\mathbb{Y}$-good $\Lambda_{\mathrm{M}}^{I\setminus\mathbb{Y}}$-term wrt. the variable $\mathbb{Y}$ of the $\underline{\bullet}$-translation. It is typable by Corollary 7.9. We conclude from Theorem 6.11 that $\underline{\iota(M)} \in \mathrm{UN}_\beta$. □

7.3 Simulation of Permutative Reductions

We have now proved that our combined mapping $\underline{\iota(\bullet)}$ results in uniformly normalising terms. We therefore know that $\underline{\iota(\bullet)}$-translated term is strongly β-normalising if it weakly β-normalising. In this section we further this by proving that the ι-translation

of the original term is strongly $\beta\mathrm{p}\pi$-normalising, if the $\underline{\iota(\bullet)}$-translated is strongly β-normalising.

This is done by proving that reductions in the memory term correspond to reductions in the $\underline{\bullet}$-translated term, *ie*, that we, in a certain sense, can simulate the $\beta\mathrm{p}\pi$-reductions on memory terms by β-reductions on the $\underline{\bullet}$-translated term. This reduces the system $\Lambda_{\mathrm{M}}^{[]}$ to Λ_{M}. More formally, we will prove

$$\underline{M} \in \mathrm{SN}_{\beta} \Longrightarrow M \in \mathrm{SN}_{\beta\mathrm{p}\pi} \ . \tag{23}$$

From this follows in particular that

$$\underline{\iota(M)} \in \mathrm{SN}_{\beta} \Longrightarrow \iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi} \ . \tag{24}$$

We do this by showing that β-reductions on the CPS-translated term $\underline{M}$ *simulate* $\beta\mathrm{p}\pi$-reductions in M, *ie*, that each $\beta\mathrm{p}\pi$-reduction can be replaced by some β-reductions in the translated term $\underline{M}$. The proof falls in two steps. The first subsection introduces some general notions about simulation and establishes a reasonable notion of simulation of $\beta\mathrm{p}\pi$-reductions. The second subsection proves that our specific CPS-translation fulfils the simulation conditions.

7.3.1 Simulation of $\beta\mathrm{p}\pi$-reductions

Our intention with the CPS-translations is to simulate the $\beta\mathrm{p}\pi$-reductions on $\Lambda_{\mathrm{M}}^{[]}$ by β-reductions in Λ_{M}. In this section we formalise what we require of such a simulation. We will define the concept of *simulation* and establish that, using a simulating translation, strong $\beta\mathrm{p}\pi$-normalisation of $\Lambda_{\mathrm{M}}^{[]}$ is reduced to strong β-normalisation of Λ_{M}.

Formally, we define the notion of simulation as follows:

Definition 7.14 A translation $\chi : \Lambda_{\mathrm{M}}^{[]} \to \Lambda_{\mathrm{M}}$ simulates $\beta\mathrm{p}\pi$ if

$$\begin{aligned} L \to_{\beta} K &\Longrightarrow \chi(L) \to_{\beta}^{+} \chi(K) \\ L \to_{\mathrm{p}} K &\Longrightarrow \chi(L) \twoheadrightarrow_{\beta} \chi(K) \\ L \to_{\pi} K &\Longrightarrow \chi(L) \twoheadrightarrow_{\beta} \chi(K) \end{aligned}$$

This is a generalisation of the notion found in [Sør97]. It is important to note that only β-reductions should be simulated through real computational steps. In the other cases the translation of the term and and its contractum might be identical.

The central point of this section is the following simulation proposition:

Proposition 7.15 (Simulation) *Let $\psi, \chi : \Lambda_{\mathrm{M}}^{[]} \to \Lambda_{\mathrm{M}}$ be functions such that χ simulates $\beta\mathrm{p}\pi$ and $\psi(M) \twoheadrightarrow_{\beta} \chi(M)$ for all $M \in \Lambda_{\mathrm{M}}^{[]}$. Then*

$$\psi(M) \in \mathrm{SN}_{\beta} \Longrightarrow M \in \mathrm{SN}_{\beta\mathrm{p}\pi} \ .$$

At first glance it might seem superfluous to have two functions in the proposition. The reason for doing so is that the translation $\underline{\bullet}$ is fairly simple to understand and, as we saw in Subsection 7.1.3, has an simple proof for the typability of the translated term. However, as mentioned on Page. 83, it has the unfortunate property that $M \to_{\beta\mathrm{p}\pi} N$ only implies $\underline{M} =_{\beta} \underline{N}$, not $\underline{M} \twoheadrightarrow_{\beta} \underline{N}$. The latter is repaired using a compacting CPS-translation $\underline{\underline{\bullet}}$, which is less intuitive, but has the property that $\underline{\underline{M}} \twoheadrightarrow_{\beta} \underline{\underline{N}}$ when $M \to_{\beta\mathrm{p}\pi} N$. We

relate the two translations by the property $\underline{M} \twoheadrightarrow_\beta \underline{\underline{M}}$, which in particular gives us the typability of $\underline{\underline{M}}$ by subject reduction.

We need that $\mathrm{p}\pi$-reductions are strongly normalising to prove the simulation proposition.

Lemma 7.16 *All terms $M \in \Lambda^{[]}_{\mathrm{M}}$ are strongly normalising under $\mathrm{p}\pi$- reductions.*

To understand the proof it is instructive to look at the diagrams of the permutative reductions of case-expressions in Figure 4.6. It is seen that in a permutative reduction the case-expression is moved closer to the root and some subterms are moved from the leftmost branch to the other branches. The idea is to find a weight function that decreases under $\mathrm{p}\pi$-reduction. It is a bit tricky, but the function below using exponentials suits the purpose.

Proof (Strong normalisation of $\mathrm{p}\pi$, Lemma 7.16). We define the following weight measure $w(\bullet)$ on $\Lambda^{[]}_{\mathrm{M}}$-terms. For the ease of reading we use the convention that the weight of a term is referred to by the small roman letter corresponding to the term, *ie*, the weight of P is p.

$$\begin{array}{lcl}
w(x) & = & 3 \\
w(\lambda x.P) = w(\mathrm{in}_i(P)) = w(\Lambda\alpha.P) = w(\llbracket P, t \rrbracket) & = & p \\
w(\langle P, Q\rangle) & = & pq \\
w(P\,Q) = w(\mathrm{let}\,\llbracket x, \alpha\rrbracket = P\,\mathrm{in}\,Q) & = & q^p \\
w(\pi_i(P)) = w(P\,t) & = & 2^p \\
w(\mathrm{case}(P,\, x.Q,\, x.R)) & = & (qr)^p \\
w([P \mid Q_1, \ldots, Q_n]) & = & pq_1 \cdots q_n
\end{array}$$

We clearly have $w(M) \geq 3$ for all terms M. By a combination of basic calculations and induction on $\mathbb{N}$, one can prove the inequalities

$$(pq)^r > (p+q)^r \qquad (*)$$

and

$$p^q > pq \qquad (*)$$

for all $p, q, r \geq 3$.

We will prove that for all M and N

$$M \to_{\mathrm{p}\pi} N \Longrightarrow w(M) > w(N)\ . \qquad (\spadesuit)$$

The proof is by induction on the structure of M using an extension of the splitting (7) on page 39 with the case $[P \mid \vec{Q}]$.

i. $M \equiv x$. There is no N such that $M \to_{\mathrm{p}\pi} N$. Hence, ($\spadesuit$) holds trivially.

ii. $M \equiv \lambda x.P$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv \llbracket P, t\rrbracket$. In all these cases the reduction is done in the subterm P. Taking $M \equiv \lambda x.P$ as an illustration, we have $M \equiv \lambda x.P \to_{\mathrm{p}\pi} \lambda x.Q \equiv N$ with $P \to_{\mathrm{p}\pi} Q$. Using the induction hypothesis, we see that $m = p > q = n$.

iii. $M \equiv \langle P, Q\rangle$. The reduction is done in either of the subterms. We consider the case where it is done in P. We have $N \equiv \langle R, Q\rangle$ where $P \to_{\mathrm{p}\pi} R$. Using the induction hypothesis, we get

$$m = pq \overset{\text{IH}}{>} rq = n\ .$$

The case where the reduction is done in Q is similar.

iv. $M \equiv D_1[S]$. In this case M might be the $\mathrm{p}\pi$-redex. By a inspection of the five forms of one-level destructor contexts we see that

$$w(D_1[S]) = g^s \tag{$\heartsuit$}$$

for some $g \geq 3$ depending on D_1 only. We split into five cases depending on the position of the $\mathrm{p}\pi$-redex:

(a) $M \equiv D_1[\mathrm{case}(P,\, x.Q,\, x.R)]$ and $N \equiv \mathrm{case}(P,\, x.D_1[Q],\, x.D_1[R])$, *ie*, M is the reduced p_{in}-redex. We have

$$m \overset{(\heartsuit)}{=} g^{(qr)^p} \overset{(*)}{>} g^{(q+r)p} = (g^q g^r)^p \overset{(\heartsuit)}{=} n \ .$$

(b) $M \equiv D_1[\mathrm{let}\, \llparenthesis x, \alpha \rrparenthesis = P \,\mathrm{in}\, Q]$ and $N \equiv \mathrm{let}\, \llparenthesis x, \alpha \rrparenthesis = P \,\mathrm{in}\, D_1[Q]$, *ie*, M is the reduced $\mathrm{p}_{\llparenthesis\rrparenthesis}$-redex. We have

$$m \overset{(\heartsuit)}{=} g^{q^p} \overset{(\star)}{>} g^{qp} = (g^q)^p \overset{(\heartsuit)}{=} n \ .$$

(c) $M \equiv D_1[\,[P \mid Q_1, \ldots, Q_n]\,]$ and $N \equiv [D_1[P] \mid Q_1, \ldots, Q_n]$, *ie*, M is the reduced π-redex. We have

$$m \overset{(\heartsuit)}{=} g^{pq_1 \cdots q_n} \overset{(\star)}{>} g^p q_1 \cdots q_n \overset{(\heartsuit)}{=} n \ .$$

(d) $M \equiv D_1[S]$ and $N \equiv D_1[T]$ where $S \to_{\mathrm{p}\pi} T$, *ie*, the reduction is done in S. Using the induction hypothesis we have

$$m \overset{(\heartsuit)}{=} g^s \overset{\mathrm{IH}}{>} g^t \overset{(\heartsuit)}{=} n \ .$$

(e) $M \equiv D_1[S]$ and $N \equiv D_1'[S]$ where $D_1 \to_{\mathrm{p}\pi} D_1'$, *ie*, the reduction is done in a subterm of the one-level destructor context D_1. By ($\heartsuit$) we have g and h such that $w(D_1[S]) = g^s$ and $w(D_1'[S]) = h^s$. Inspection of the five forms of one-level destructor contexts shows that $g > h$ by the induction hypothesis. This leads to the conclusion that

$$m = g^s \overset{\mathrm{IH}}{>} h^s = n \ .$$

This exhausts the possibilities, and we conclude that the measure $w(\bullet)$ is decreasing under $\mathrm{p}\pi$-reduction. As $\mathbb{N}$ is well-founded, we cannot have a infinite $\mathrm{p}\pi$-reduction path. We therefore conclude that all Λ_{M}-terms are $\mathrm{SN}_{\mathrm{p}\pi}$. □

We then prove the simulation proposition:

Proof (Simulation Proposition, Proposition 7.15). We use *reductio ad absurdum* and therefore assume that $\psi(M) \in \mathrm{SN}_\beta$. Suppose M has an infinite $\beta\mathrm{p}\pi$ reduction path. By Lemma 7.16 there can only be finitely many $\mathrm{p}\pi$-reductions between each β-reduction. Thus, we have infinitely many β-reductions. As χ simulates $\beta\mathrm{p}\pi$-reduction, each of these corresponds to at least one β-reduction in a reduction path of $\chi(M)$. Thus, $\chi(M) \in \infty_\beta$ and then $\psi(M) \twoheadrightarrow_\beta \chi(M) \in \infty_\beta$ contradicting $\psi(M) \in \mathrm{SN}_\beta$.

This proves that $\psi(M) \in \mathrm{SN}_\beta$ implies $M \in \mathrm{SN}_{\beta\mathrm{p}\pi}$. □

7.3.2 CPS-translations internalise $\mathrm{p}\pi$-reductions

Using the general simulation framework, we will prove that the specific CPS-transformations of Definition 7.4 and 7.5 fulfil the conditions of the Simulation Proposition with $\psi = \underline{\bullet}$ and $\chi = \underline{\underline{\bullet}}$. We start out with some lemmata. The first one states that if the substitution variable is not free in the first argument of $\bullet : \bullet$, then the substitution can simply be done in the second argument.

Lemma 7.17 *Let $M \in \Lambda_{\mathrm{M}}^{[]}$ and $K, L \in \Lambda_{\mathrm{M}}$ be terms and let k a variable such that $k \notin \mathrm{FV}(M)$. Then*

$$(M : K)\{k := L\} \equiv M : K\{k := L\} \ .$$

Proof. The proof is done by induction on the structure of M. We note that in each of the induction steps we have

$$\underline{\underline{M}}\{k := L\} \overset{7.5}{=} \lambda l.(M : l)\{k := L\} \overset{\text{IH}}{=} (\lambda l.M : l) \overset{7.5}{=} \underline{\underline{M}} \ . \qquad (*)$$

We consider the cases:

i. $M \equiv x$. We have $k \neq x$ and hence

$$(x : K)\{k := L\} \overset{7.5}{=} (x\ K)\{k := L\} = x\ (K\{k := L\}) \overset{7.5}{=} x : K\{k := L\} \ .$$

ii. $M \equiv \lambda x.P$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. The cases are similar and we take $M \equiv \langle P, Q\rangle$ as illustration. We have $k \notin \mathrm{FV}(P) \cup \mathrm{FV}(Q)$ and therefore

$$\begin{aligned}(\langle P, Q\rangle : K)\{x := L\} &\overset{7.5}{=} (K\ \langle \underline{\underline{P}}, \underline{\underline{Q}}\rangle)\{x := L\} \\ &\overset{(*)}{=} (K\{x := L\})\ \langle \underline{\underline{P}}, \underline{\underline{Q}}\rangle \\ &\overset{7.5}{=} \langle P, Q\rangle : K\{x := L\} \ .\end{aligned}$$

iii. $M \equiv P\,Q$, or $M \equiv \pi_i(P)$, or $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$, or $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$, or $M \equiv P\,t$. The cases are similar and we use $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$ as an example. We have

$$\begin{aligned}&(\mathrm{case}(P,\, x.Q,\, x.R) : K)\{x := L\} \\ &\quad\overset{7.5}{=} (P : \lambda k.\mathrm{case}(k,\, x.(Q : K),\, x.(R : K)))\{x := L\} \\ &\quad\overset{\text{IH}}{=} (P : \lambda k.\mathrm{case}(k,\, x.(Q : K)\{x := L\},\, x.(R : K)\{x := L\})) \\ &\quad\overset{\text{IH}}{=} P : \lambda k.\mathrm{case}(k,\, x.(Q : K\{x := L\}),\, x.(R : K\{x := L\})) \ .\end{aligned}$$

iv. $M \equiv [P \mid Q_1, \ldots, Q_n]$. We have $k \notin \mathrm{FV}(P), \mathrm{FV}(Q)$ and $y \neq k$ by Definition 7.5. Therefore

$$\begin{aligned}([P \mid Q_1, \ldots, Q_n] : K)\{k := L\} &\overset{7.5}{=} (y\ (P : K)\ \underline{\underline{Q_1}}\ \cdots\ \underline{\underline{Q_n}})\{k := L\} \\ &\overset{(*)\ \&\ \text{IH}}{\equiv} y\ (P : K\{k := L\})\ \underline{\underline{Q_1}}\ \cdots\ \underline{\underline{Q_n}} \\ &\overset{7.5}{\equiv} [P \mid Q_1, \ldots, Q_n] : K\{k := L\} \ .\end{aligned}$$

This exhausts the possibilities. It also concludes the proof that substitutions on a variable not free in the first argument of $\bullet : \bullet$ should only be applied to the second argument. □

Lemma 7.18 *Let $M, N \in \Lambda^{[]}_{\mathrm{M}}$ be a $\Lambda^{[]}_{\mathrm{M}}$-term and t be eigenterm. Then*

$$(M:K)\{\alpha := t\} \equiv M\{\alpha := t\}:K\{\alpha := t\}$$

for any type variable α.

Proof. It is intuitively easy to see why the lemma holds: In the compacting CPS-translations in Definition 7.5 eigenterms are not translated; they are copied literally into the translated term. As type variables only appear in eigenterms, a free type variable is simply copied with the eigenterm it appears in. Formally, it is proven using induction on the structure of M. We skip this proof to prevent ourself from falling asleep. □

The next lemma states that reductions done in the second argument of $\bullet : \bullet$ are conserved when applying $\bullet:\bullet$

Lemma 7.19 *Let $M \in \Lambda^{[]}_{\mathrm{M}}$ be a $\Lambda^{[]}_{\mathrm{M}}$-term and $K, L \in \Lambda_{\mathrm{M}}$ be Λ_{M}-terms. If $K \twoheadrightarrow^{+}_{\beta} L$, then*

$$M:K \twoheadrightarrow^{+}_{\beta} M:L \ .$$

Corollary 7.20 *For all $M, N \in \Lambda^{[]}_{\mathrm{M}}$ and $K, L \in \Lambda_{\mathrm{M}}$ we have*

$$K \twoheadrightarrow_{\beta} L \Longrightarrow M:K \twoheadrightarrow_{\beta} M:L \ .$$

Proof (Lemma 7.19). We proceed by induction on the structure of M.

i. $M \equiv x$, or $M \equiv \lambda x.P$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. The cases are similar and we use $M \equiv x$ as illustration. Using the compatibility of $\rightarrow_\beta$ we have

$$x:K \overset{7.5}{\equiv} x\ K \twoheadrightarrow^{+}_{\beta} x\ L = x:L \ .$$

ii. $M \equiv P\,Q$, or $M \equiv \pi_i(P)$, or $M \equiv P\,t$. The cases are similar and we illustrate their proof by the case $M \equiv P\,Q$. Using the compatibility of $\rightarrow_\beta$ and the induction hypothesis, we have

$$P\,Q:K \overset{7.5}{=} P:\lambda k.k\ \underline{Q}\ K \overset{\text{IH}}{\twoheadrightarrow^{+}_{\beta}} P:\lambda k.k\ \underline{Q}\ L \overset{7.5}{=} P\,Q:L \ .$$

iii. $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$, or $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$. The cases are similar and we use $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$ as an example. As $K \twoheadrightarrow^{+}_{\beta} L$, the compatibility of $\rightarrow_\beta$, together with the induction hypothesis, lets us conclude that

$$\lambda k.\mathrm{case}(k,\, x.(Q:K),\, x.(R:K)) \overset{\text{IH}}{\twoheadrightarrow^{+}_{\beta}} \lambda k.\mathrm{case}(k,\, x.(Q:L),\, x.(R:L)) \ . \qquad (\star)$$

Using this we have

$$\begin{aligned}\mathrm{case}(P,\, x.Q,\, x.R):K &\overset{7.5}{=} P:\lambda k.\mathrm{case}(k,\, x.(Q:K),\, x.(R:K)) \\ &\overset{\text{IH \& } (\star)}{\twoheadrightarrow^{+}_{\beta}} P:\lambda k.\mathrm{case}(k,\, x.(Q:L),\, x.(R:L)) \\ &\overset{7.5}{=} \mathrm{case}(P,\, x.Q,\, x.R):L \ .\end{aligned}$$

iv. $M \equiv [P \mid Q_1, \ldots, Q_n]$. By the induction hypothesis we have the following reduction path:

$$\begin{aligned}[P \mid Q_1, \ldots, Q_n]{:}K &\overset{7.5}{=} y\,(P{:}K)\,\underline{\underline{Q_1}} \cdots \underline{\underline{Q_n}} \\ &\overset{\text{IH}}{\twoheadrightarrow}^{+}_{\beta} y\,(P{:}L)\,\underline{\underline{Q_1}} \cdots \underline{\underline{Q_n}} \\ &\overset{7.5}{=} [P \mid Q_1, \ldots, Q_n]{:}L\ .\end{aligned}$$

This exhausts the possibilities. We conclude that reductions in the second argument of $\bullet{:}\bullet$ are conserved under application of $\bullet{:}\bullet$. □

We continue with a lemma stating the crucial property in the simulation: when we substitute a $\underline{\underline{\bullet}}$-translated term we can "remove" the $\underline{\underline{\bullet}}$-translation, in the first argument by a number of β-reductions.

Lemma 7.21 *Let $M, N \in \Lambda^{[]}_{\mathrm{M}}$ be a $\Lambda^{[]}_{\mathrm{M}}$-term and $K \in \Lambda_{\mathrm{M}}$ a Λ_{M}-term. Then*

$$(M{:}K)\{x := \underline{\underline{N}}\} \twoheadrightarrow_{\beta} M\{x := N\}{:}K\{x := \underline{\underline{N}}\}$$

for any term variable x.

Before doing the proof, we do following observation that holds for any P and Q:

$$\underline{\underline{P}}\,Q \overset{7.5}{\equiv} (\lambda k.P{:}k)\,Q \to_{\beta} (P{:}k)\{k := Q\} \overset{7.17}{\equiv} P{:}Q \tag{25}$$

where we use that k is introduced by the translation and hence $k \notin \mathrm{FV}(P)$.

Proof (Lemma 7.21). We will use induction on the structure of M. In the induction steps we have

$$\underline{\underline{M}}\{x := \underline{\underline{N}}\} \overset{7.5}{\equiv} \lambda k.(M{:}k)\{x := \underline{\underline{N}}\} \overset{\text{IH}}{\twoheadrightarrow}_{\beta} \lambda k.(M\{x := N\}{:}k) \overset{7.5}{\equiv} \underline{\underline{M\{x := N\}}}\ . \tag{$*$}$$

We consider the following cases:

i. $M \equiv y$. We divide into two subcases

   (a) $x = y$. We have

$$\begin{aligned}x{:}K\{x := \underline{\underline{N}}\} &\overset{7.5}{\equiv} (x\ K)\{x := \underline{\underline{N}}\} \\ &= \underline{\underline{N}}(K\{x := \underline{\underline{N}}\}) \\ &\overset{(25)}{\to}_{\beta} N{:}K\{x := \underline{\underline{N}}\} \\ &= x\{x := N\}{:}K\{x := \underline{\underline{N}}\}\ .\end{aligned}$$

   (b) $y \neq x$. We have

$$\begin{aligned}(y{:}K)\{x := \underline{\underline{N}}\} &\overset{7.17}{\equiv} y{:}K\{x := \underline{\underline{N}}\} \\ &\overset{7.5}{\equiv} y\{x := N\}{:}K\{x := \underline{\underline{N}}\}\ .\end{aligned}$$

ii. $M \equiv \lambda y.P$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. These cases are similar and we take the most complicated case, $M \equiv \langle P, Q\rangle$, as illustration. Using the induction hypothesis we have

$$\begin{aligned}(\langle P, Q\rangle{:}K)\{x := \underline{\underline{N}}\} &\overset{7.5}{\equiv} (K\ \langle \underline{\underline{P}}, \underline{\underline{Q}}\rangle)\{x := \underline{\underline{N}}\} \\ &\equiv (K\{x := \underline{\underline{N}}\})\ \langle \underline{\underline{P}}\{x := \underline{\underline{N}}\}, \underline{\underline{Q}}\{x := \underline{\underline{N}}\}\rangle \\ &\overset{(*)}{\twoheadrightarrow}_{\beta} (K\{x := \underline{\underline{N}}\})\ \langle \underline{\underline{P\{x := N\}}}, \underline{\underline{Q\{x := N\}}}\rangle \\ &\overset{7.5}{\equiv} \langle P, Q\rangle\{x := N\}{:}\underline{\underline{Q\{x := N\}}}\ .\end{aligned}$$

iii. $M \equiv P\,Q$, or $M \equiv \mathrm{case}(P,\, y.Q,\, y.R)$, or $M \equiv \mathrm{let}\,[\![y, \alpha]\!] = P\,\mathrm{in}\,Q$. These three cases are similar and we illustrate their proof using $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$. We start out noting that by the induction hypothesis and the compatibility of $\to_\beta$ we have

$$\begin{aligned}\lambda k.\mathrm{case}(k,\, y.(Q\!:\!K)\{x := \underline{\underline{N}}\},\, y.(R\!:\!K)\{x := \underline{\underline{N}}\}) &\overset{\mathrm{IH}}{\twoheadrightarrow_\beta}\\ \lambda k.\mathrm{case}(k,\, y.(Q\{x := N\}\!:\!K\{x := \underline{\underline{N}}\}),\, y.(R\{x := N\}\!:\!K\{x := \underline{\underline{N}}\})) &\quad (\star)\end{aligned}$$

With this reduction path at hand, we can do the following reduction:

$$\begin{aligned}&(\mathrm{case}(P,\, x.Q,\, x.R)\!:\!K)\{x := \underline{\underline{N}}\}\\ &\quad\overset{7.5}{\equiv} \big(P\!:\!\lambda k.\mathrm{case}(k,\, y.(Q\!:\!K),\, y.(R\!:\!K))\big)\{x := \underline{\underline{N}}\}\\ &\quad\overset{\mathrm{IH}}{\twoheadrightarrow_\beta} P\{x := N\}\!:\!\lambda k.\mathrm{case}(k,\, u.(Q\!:\!K)\{x := \underline{\underline{N}}\},\, u.(R\!:\!K)\{x := \underline{\underline{N}}\})\\ &\quad\overset{7.20\ \&\ (\star)}{\twoheadrightarrow_\beta} P\{x := N\}\!:\!\lambda k.\mathrm{case}(k,\, y.(Q\{x := N\}\!:\!K\{x := \underline{\underline{N}}\}),\\ &\qquad\qquad y.(R\{x := N\}\!:\!K\{x := \underline{\underline{N}}\}))\\ &\quad\overset{7.5}{\equiv} \mathrm{case}(P\{x := N\},\, y.Q\{x := N\},\, y.R\{x := N\})\!:\!K\{x := \underline{\underline{N}}\}\\ &\quad\equiv \mathrm{case}(P,\, y.Q,\, y.R)\{x := N\}\!:\!K\{x := \underline{\underline{N}}\}\end{aligned}$$

iv. $M \equiv \pi_i(P)$ or $M \equiv P\,t$. The cases are similar and we use $M \equiv P\,t$ as illustration. The proposition follows directly from the induction hypothesis:

$$\begin{aligned}(P\,t\!:\!K)\{x := \underline{\underline{N}}\} &\overset{7.5}{\equiv} (P\!:\!\lambda k.k\,t\,K)\{x := \underline{\underline{N}}\}\\ &\overset{\mathrm{IH}}{\twoheadrightarrow_\beta} P\{x := N\}\!:\!\lambda k.k\,t\,(K\{x := \underline{\underline{N}}\})\\ &\overset{7.5}{\equiv} P\,t\{x := N\}\!:\!K\{x := \underline{\underline{N}}\}\ .\end{aligned}$$

v. $M \equiv [P \mid Q_1, \ldots, Q_n]$. This last case also follows from the induction hypothesis by some simple rewritings:

$$\begin{aligned}&([P \mid Q_1, \ldots, Q_n]\!:\!K)\{x := \underline{\underline{N}}\}\\ &\quad\overset{7.5}{\equiv} (y\,(P\!:\!K)\,\underline{\underline{Q_1}}\,\cdots\,\underline{\underline{Q_n}})\{x := \underline{\underline{N}}\}\\ &\quad= y\,\big((P\!:\!K)\{x := \underline{\underline{N}}\}\big)\,(\underline{\underline{Q_1}}\{x := \underline{\underline{N}}\})\,\cdots\,(\underline{\underline{Q_n}}\{x := \underline{\underline{N}}\})\\ &\quad\overset{\mathrm{IH}}{\twoheadrightarrow_\beta} y\,(P\{x := N\}\!:\!K\{x := \underline{\underline{N}}\})\,(\underline{\underline{Q_1\{x := N\}}})\,\cdots\,(\underline{\underline{Q_n\{x := N\}}})\\ &\quad\overset{7.5}{\equiv} [P\{x := N\} \mid Q_1\{x := N\}, \ldots, Q_n\{x := N\}]\colon\ .\end{aligned}$$

This exhausts the possibilities and concludes the proof that the $\underline{\underline{\bullet}}$-translation of a substitution can be "removed" from the first argument of $\bullet\!:\!\bullet$ by β-reduction. □

When establishing the simulation, we shall prove that we have a reduction path between the translation of a term with a redex and the reduced term. The following lemma establishes that if the reduction is compatible, it is enough to prove that the reduction path exists when the term is the redex.

Lemma 7.22 *Let $\mathbf{R}$ be a notion of reduction on $\Lambda^{[]}_{\mathrm{M}}$. Let $M, N \in \Lambda^{[]}_{\mathrm{M}}$ be $\Lambda^{[]}_{\mathrm{M}}$-terms where $M \to_{\mathbf{R}} N$. If $M\!:\!K \twoheadrightarrow^+_\beta N\!:\!K$ for any term $K \in \Lambda_{\mathrm{M}}$, then*

$$C[M]\!:\!K \twoheadrightarrow^+_\beta C[N]\!:\!K$$

where C is a context and K is any Λ_{M}-term.

Proof. We do the proof by induction on the structure of the context.

i. $C \equiv []$. This is trivially true by the assumptions.

ii. $C \equiv \lambda x.C'$, or $C \equiv \langle C', Q\rangle$, or $C \equiv \langle P, C'\rangle$, or $C \equiv \mathrm{in}_i(C')$, or $C \equiv \Lambda\alpha.C'$, or $C \equiv [\![C', t]\!]$. These cases are similar and we use $C \equiv \langle C', Q\rangle$ as illustration. The proposition follows directly from the induction hypothesis using compatibility:

$$\langle C'[M], Q\rangle : K \overset{7.5}{\equiv} K\ \langle \underline{\underline{C'[M]}}, Q\rangle \overset{\mathrm{IH}}{\twoheadrightarrow_\beta} K\ \langle \underline{\underline{C'[N]}}, Q\rangle \overset{7.5}{\equiv} \langle C'[N], Q\rangle : K\ .$$

iii. $C \equiv C'\,Q$, or $C \equiv \pi_i(C')$, or $C \equiv \mathrm{case}(C', x.Q, x.R)$, or $C \equiv \mathrm{let}\,[\![x, \alpha]\!] = C'\,\mathrm{in}\,Q$, or $C \equiv C'\,t$, or $C \equiv [C' \mid Q_1, \ldots, Q_n]$, or $C \equiv [P \mid Q_1, \ldots, C', \ldots, Q_n]$. These cases are similar and we use $C \equiv C'\,Q$ as example. We have

$$C'[M]\,Q : K \overset{7.5}{\equiv} C'[M] : \lambda k.k\ \underline{Q}\ K \overset{\mathrm{IH}}{\twoheadrightarrow_\beta} C'[N] : \lambda k.k\ \underline{Q}\ K \overset{7.5}{\equiv} C'[N]\,Q : K$$

using the compatibility of $\to_\beta$.

iv. $C \equiv \mathrm{case}(P, x.C', x.R)$, or $C \equiv \mathrm{case}(P, x.Q, x.C')$, or $C \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,C'$, or $C \equiv P\,C'$. The cases are similar and we take $C \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,C'$ as an illustrating example. We first note that

$$\lambda k.\mathrm{let}\,[\![x, \alpha]\!] = k\,\mathrm{in}\,(C'[M] : K) \overset{\mathrm{IH}}{\twoheadrightarrow_\beta} \lambda k.\mathrm{let}\,[\![x, \alpha]\!] = k\,\mathrm{in}\,(C'[N] : K) \qquad (*)$$

using the compatibility of $\to_\beta$. We then use this as a stepping stone to make the reduction path

$$\begin{aligned}\mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,C'[M] : K &\overset{7.5}{\equiv} P : \lambda k.\mathrm{let}\,[\![x, \alpha]\!] = k\,\mathrm{in}\,(C'[M] : K)\\ &\overset{(*)\ \&\ 7.20}{\twoheadrightarrow_\beta} P : \lambda k.\mathrm{let}\,[\![x, \alpha]\!] = k\,\mathrm{in}\,(C'[N] : K)\\ &\quad \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,C'[N] : K\ .\end{aligned}$$

This exhausts the possibilities and concludes the proof that the $\bullet : \bullet$-translation encodes compatibility. □

We see that the only reduction steps taken are the reduction steps from M to N. We therefore state the following corollary:

Corollary 7.23 *Let $\mathbf{R}$ be a notion of reduction on $\Lambda^{[]}_{\mathrm{M}}$. Let $M, N \in \Lambda^{[]}_{\mathrm{M}}$ be $\Lambda^{[]}_{\mathrm{M}}$-terms where $M \to_{\mathbf{R}} N$. If $\underline{\underline{M}} \equiv \underline{\underline{N}}$, then*

$$C[M] : K \equiv C[N] : K$$

where C is any context and K is any Λ_{M}-term.

We are now able to prove that the CPS-transformations of Definition 7.4 and 7.5 fulfil the conditions of the Simulation Proposition 7.15 with $\psi = \underline{\bullet}$ and $\chi = \underline{\underline{\bullet}}$.

Proposition 7.24 *For all $M, N \in \Lambda^{[]}_{\mathrm{M}}$ and $K \in \Lambda_{\mathrm{M}}$:*

i. $\underline{M} \twoheadrightarrow_\beta \underline{\underline{M}}$

ii. $M \to_\beta N \Longrightarrow M : K \twoheadrightarrow^+_\beta N : K$

iii. $M \to_{\mathrm{p}\pi} N \Longrightarrow M : K \equiv N : K$

Proof. We prove Case i by induction on M.

i. $M \equiv x$. We have

$$\underline{x} \overset{7.4}{\equiv} \lambda k.x\, k \overset{7.5}{\equiv} \lambda k.x : k \overset{7.5}{\equiv} \underline{x} \ .$$

ii. $M \equiv \lambda x.P$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv [\![P, t]\!]$. These cases are straight-forward using the induction hypothesis. We take $M \equiv \langle P, Q\rangle$ as an example

$$\begin{aligned}\underline{\langle P, Q\rangle} &\overset{7.4}{\equiv} \lambda k.k\ \langle \underline{P}, \underline{Q}\rangle \\ &\overset{\text{IH}}{\twoheadrightarrow_\beta} \lambda k.k\ \langle \underline{\underline{P}}, \underline{\underline{Q}}\rangle \\ &\overset{7.5}{\equiv} \lambda k.(\langle P, Q\rangle : k) \\ &\overset{7.5}{\equiv} \underline{\langle P, Q\rangle} \ .\end{aligned}$$

iii. $M \equiv P\,Q$, or $M \equiv \pi_i(P)$, or $M \equiv P\,t$. These cases are alike and we illustrate their proof by the case $M \equiv P\,Q$:

$$\begin{aligned}\underline{P\,Q} &\overset{7.4}{\equiv} \lambda k.\underline{P}\ \lambda m.m\ \underline{Q}\ k \\ &\overset{\text{IH}}{\twoheadrightarrow_\beta} \lambda k.\underline{\underline{P}}\ \lambda m.m\ \underline{\underline{Q}}\ k \\ &\overset{(25)}{\rightarrow_\beta} \lambda k.(P : \lambda m.m\ \underline{\underline{Q}}\ k) \\ &\overset{7.5}{\equiv} \lambda k.(P\,Q : k) \\ &\overset{7.5}{\equiv} \underline{\underline{P\,Q}} \ .\end{aligned}$$

iv. $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$ or $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P \,\mathrm{in}\, Q$. These two cases are similar and we take $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P \,\mathrm{in}\, Q$ as an example. We have

$$\begin{aligned}\underline{\mathrm{let}\,[\![x, \alpha]\!] = P \,\mathrm{in}\, Q} &\overset{7.4}{\equiv} \lambda k.\underline{P}\ \lambda m.\mathrm{let}\,[\![x, \alpha]\!] = m \,\mathrm{in}\, \underline{Q}\ k \\ &\overset{\text{IH}}{\twoheadrightarrow_\beta} \lambda k.\underline{\underline{P}}\ \lambda m.\mathrm{let}\,[\![x, \alpha]\!] = m \,\mathrm{in}\, \underline{\underline{Q}}\ k \\ &\overset{(25)}{\twoheadrightarrow^2_\beta} \lambda k.(P : \lambda m.\mathrm{let}\,[\![x, \alpha]\!] = m \,\mathrm{in}\, (Q : k)) \\ &\overset{7.5}{\equiv} \lambda k.(\mathrm{let}\,[\![x, \alpha]\!] = P \,\mathrm{in}\, Q : k) \\ &\overset{7.5}{\equiv} \underline{\underline{\mathrm{let}\,[\![x, \alpha]\!] = P \,\mathrm{in}\, Q}} \ .\end{aligned}$$

v. $M \equiv [P \mid Q_1, \ldots, Q_n]$. We have the following reduction path:

$$\begin{aligned}\underline{[P \mid Q_1, \ldots, Q_n]} &\overset{7.4}{\equiv} \lambda k.y\ (\underline{P}\ k)\ \underline{Q_1}\ \cdots\ \underline{Q_n} \\ &\overset{\text{IH}}{\twoheadrightarrow_\beta} \lambda k.y\ (\underline{\underline{P}}\ k)\ \underline{\underline{Q_1}}\ \cdots\ \underline{\underline{Q_n}} \\ &\overset{(25)}{\rightarrow_\beta} \lambda k.y\ (P : k)\ \underline{\underline{Q_1}}\ \cdots\ \underline{\underline{Q_n}} \\ &\overset{7.5}{\equiv} \lambda k.([P \mid Q_1, \ldots, Q_n] : k) \\ &\overset{7.5}{\equiv} \underline{\underline{[P \mid Q_1, \ldots, Q_n]}} \ .\end{aligned}$$

This completes the proof that $\underline{M} \twoheadrightarrow_\beta \underline{\underline{M}}$.

By Lemma 7.22 it is sufficient to consider the five different forms of β-redexes to prove Case ii of the proposition. We do the following case analysis:

i. $(\lambda x.P)\,Q \to_{\beta_\lambda} P\{x := Q\}$. We note that

$$\underline{\underline{M}}\{k := L\} \overset{7.5}{=} \lambda l.(M : l)\{k := L\} \overset{7.17}{=} (\lambda l.M : l) \overset{7.5}{=} \underline{\underline{M}} \qquad (*)$$

by Lemma 7.17 and

$$\underline{\underline{M}}\{x := \underline{N}\} \overset{7.5}{\equiv} \lambda k.(M : k)\{x := \underline{N}\} \overset{7.21}{\twoheadrightarrow_\beta} \lambda k.(M\{x := N\} : k) \overset{7.5}{\equiv} \underline{\underline{M\{x := N\}}} \qquad (\star)$$

by Lemma 7.21. We use these two observations to do the following reduction:

$$\begin{aligned}(\lambda x.P)\,Q : K &\overset{(*)}{\equiv} (\lambda k.k\,\underline{\underline{Q}}\,K)\,\lambda x.\underline{\underline{P}} \\ &\to_{\beta_\lambda} (\lambda x.\underline{\underline{P}})\,\underline{\underline{Q}}\,K \\ &\to_{\beta_\lambda} (\underline{\underline{P}}\{x := \underline{\underline{Q}}\})\,K \\ &\overset{(\star)}{\twoheadrightarrow_\beta} \underline{\underline{P\{x := Q\}}}\,K \\ &\overset{(25)}{\to_\beta} P\{x := Q\} : K\ .\end{aligned}$$

ii. $M \equiv \pi_i(\langle P_1, P_2\rangle) \to_{\beta_{\langle\rangle}} P_i \equiv N$. We then have

$$\begin{aligned}\pi_i(\langle P_1, P_2\rangle) : K &\overset{7.5}{\equiv} (\lambda k.\pi_i(k)\,K)\,\langle \underline{\underline{P_1}}, \underline{\underline{P_2}}\rangle \\ &\to_{\beta_\lambda} \pi_i(\langle \underline{\underline{P_1}}, \underline{\underline{P_2}}\rangle)\,K \\ &\to_{\beta_{\langle\rangle}} \underline{\underline{P_i}}\,K \\ &\overset{(25)}{\to_\beta} P_i : K\ .\end{aligned}$$

iii. $M \equiv \mathrm{case}(\mathrm{in}_i(P),\, x.Q_1,\, x.Q_2) \to_{\beta_{\mathrm{in}}} Q_i\{x := P\}$. We make the following reductions:

$$\begin{aligned}\mathrm{case}(\mathrm{in}_i(P),\, x.Q_1,\, x.Q_2) : K &\overset{7.5}{\equiv} (\lambda k.\mathrm{case}(k,\, x.(Q_1 : K),\, x.(Q_2 : K)))\,\mathrm{in}_i(\underline{\underline{P}}) \\ &\to_{\beta_\lambda} \mathrm{case}(\mathrm{in}_i(\underline{\underline{P}}),\, x.(Q_1 : K),\, x.(Q_2 : K)) \\ &\to_{\beta_{\mathrm{in}}} (Q_i : K)\{x := \underline{\underline{P}}\} \\ &\overset{7.21}{\twoheadrightarrow_\beta} Q_i\{x := P\} : K\ .\end{aligned}$$

iv. $M \equiv (\Lambda\alpha.P)\,t \to_{\beta_\lambda} P\{\alpha := t\}$. Using Lemma 7.18 we have

$$\begin{aligned}(\Lambda\alpha.P)\,t : K &\overset{7.5}{\equiv} (\lambda k.k\,t\,K)\,\Lambda\alpha.\underline{\underline{P}} \\ &\to_{\beta_\lambda} (\Lambda\alpha.\underline{\underline{P}})\,t\,K \\ &\overset{7.18}{\to_{\beta_\Lambda}} \underline{\underline{P\{\alpha := t\}}}\,K \\ &\overset{(25)}{\to_\beta} P\{\alpha := t\} : K\ .\end{aligned}$$

v. $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![P, t]\!]\,\mathrm{in}\,Q \to_{\beta_\lambda} Q\{x := P, \alpha := t\}$. Using Lemmata 7.18 and 7.21 we have

$$\begin{aligned}\mathrm{let}\,[\![x, \alpha]\!] = [\![P, t]\!]\,\mathrm{in}\,Q : K &\overset{7.5}{\equiv} (\lambda k.\mathrm{let}\,[\![x, \alpha]\!] = k\,\mathrm{in}\,(Q : K))\,[\![\underline{\underline{P}}, t]\!] \\ &\to_{\beta_\lambda} \mathrm{let}\,[\![x, \alpha]\!] = [\![\underline{\underline{P}}, t]\!]\,\mathrm{in}\,(Q : K) \\ &\to_{\beta_{[\![]\!]}} (Q_1 : K)\{x := \underline{\underline{P}}, \alpha := t\} \\ &\overset{7.18\ \&\ 7.21}{\twoheadrightarrow_\beta} Q\{x := P, \alpha := t\} : K\ .\end{aligned}$$

This exhausts the possibilities and that $M \to_\beta N$ implies $M : K \twoheadrightarrow_\beta^+ N : K$ is concluded by Lemma 7.22.

In Case iii the p and π-reductions are defined using a one-level destructor context. In the compacting CPS-translation the subterm that in the contractum is filled into the one-level destructor context, is translated to $Q : K$. The three notion of reductions are treated similarly in the proof and we only consider the case for $\mathrm{p}_{[\![]\!]}$. By Corollary 7.23 it is only necessary to prove Case iii for the five different forms of redex:

i. $(\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q)\;R \to_{\mathrm{p}[\,]} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q\;R$ or
$(\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q)\;t \to_{\mathrm{p}[\,]} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q\;t$. The cases are similar, and we use the latter as illustration:

$$\begin{aligned}(\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q)\;t : K &\overset{7.5}{\equiv} P : \lambda k.\mathrm{let}\,\llbracket x,\alpha\rrbracket = k\,\mathrm{in}\,(Q : \lambda l.l\;t\;K)\\ &\overset{7.5}{\equiv} P : \lambda k.\mathrm{let}\,\llbracket x,\alpha\rrbracket = k\,\mathrm{in}\,(Q\;t : K)\\ &\overset{7.5}{\equiv} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q\;t : K\ .\end{aligned}$$

ii. $\pi_i(\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q) \to_{\mathrm{p}[\,]} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,\pi_i(Q)$. A simple rewriting using Definition 7.5 gives us the desired result:

$$\begin{aligned}\pi_i(\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q) : K &\overset{7.5}{\equiv} P : \lambda k.\mathrm{let}\,\llbracket x,\alpha\rrbracket = k\,\mathrm{in}\,(Q : \lambda l.\pi_i(l)\;K)\\ &\overset{7.5}{\equiv} P : \lambda k.\mathrm{let}\,\llbracket x,\alpha\rrbracket = k\,\mathrm{in}\,(\pi_i(Q) : K)\\ &\overset{7.5}{\equiv} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,\pi_i(Q) : K\ .\end{aligned}$$

iii. The reduction

$$\mathrm{case}(\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q,\ y.R_1,\ y.R_2) \to_{\mathrm{p}[\,]} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,\mathrm{case}(Q,\ y.R_1,\ y.R_2)$$

or

$$\mathrm{let}\,\llbracket y,\beta\rrbracket = (\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q)\,\mathrm{in}\,R \to_{\mathrm{p}[\,]} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,\mathrm{let}\,\llbracket y,\beta\rrbracket = Q\,\mathrm{in}\,R\ .$$

The cases are similar and we use the latter as illustration:

$$\begin{aligned}\mathrm{let}\,\llbracket y,\beta\rrbracket = (\mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,Q)\,\mathrm{in}\,R : K &\\ \overset{7.5}{\equiv} P : \lambda k.\mathrm{let}\,\llbracket x,\alpha\rrbracket = k\,\mathrm{in}\,(Q : \lambda l.\mathrm{let}\,\llbracket y,\beta\rrbracket = l\,\mathrm{in}\,R : K)&\\ \overset{7.5}{\equiv} P : \lambda k.\mathrm{let}\,\llbracket x,\alpha\rrbracket = k\,\mathrm{in}\,(\mathrm{let}\,\llbracket y,\beta\rrbracket = Q\,\mathrm{in}\,R : K)&\\ \overset{7.5}{\equiv} \mathrm{let}\,\llbracket x,\alpha\rrbracket = P\,\mathrm{in}\,\mathrm{let}\,\llbracket y,\beta\rrbracket = Q\,\mathrm{in}\,R : K\ .&\end{aligned}$$

This exhausts the possibilities and hence $M \to_{\mathrm{p}[\,]} N \Rightarrow M : K \equiv N : K$ by Corollary 7.23. The other two other notions of reductions are treated similarly, we conclude that $M \to_{\mathrm{p}\pi} N \Longrightarrow M : K \equiv N : K$ □

Corollary 7.25 *For all* $M \in \Lambda_{\mathrm{M}}$

$$\underline{\iota(M)} \in \mathrm{SN}_{\beta} \Longrightarrow \iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi}$$

We see from the proof of Case ii of Proposition 7.24 that not all internal redexes are removed by the :-translation. A closer study of the proof and the proof of Lemma 7.18 reveals that the pattern is that a reduction is simulated by a number of internal reductions, the reduction itself, and then a number of internal reductions.

7.4 Strong Normalisation simulation of Λ_{M} by $\Lambda_{\mathrm{M}}^{[]}$

The last step in our proof is to prove that Klop's system with memory simulates Λ_{M} wrt. strong normalisation, *ie*,

$$\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi} \Longrightarrow M \in \mathrm{SN}_{\beta\mathrm{p}}\ .$$

We follow an idea from [Sør97] and prove it by the three implications

$$\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi} \Longrightarrow \iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi\kappa} \Longrightarrow \iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\kappa} \Longrightarrow M \in \mathrm{SN}_{\beta\mathrm{p}} \ . \tag{26}$$

The first implication is caused by the possibility to postpone κ-reductions. The second implication is trivial as a $\beta\mathrm{p}\kappa$-reduction is also a $\beta\mathrm{p}\pi\kappa$-reduction. The third implication follows from $\iota(M) \twoheadrightarrow_\kappa M$. We prove the first and the third implication below. It is mainly an extension of the proofs found in [Sør97, MN99].

7.4.1 Postponement of κ-redexes

To prove the first implication of (26) we need some lemmata. We start with a substitution lemma.

Lemma 7.26 (Substitution lemma for κ-reductions) *Let M, N, and O be $\Lambda^{[]}_{\mathrm{M}}$-terms. If $M \to_\kappa N$, then the following hold:*

i. $M\{z := O\} \to_\kappa N\{z := O\}$,

ii. $M\{\alpha := t\} \to_\kappa N\{\alpha := t\}$, *and*

iii. $O\{z := M\} \twoheadrightarrow_\kappa O\{z := N\}$

where z is any variable, α is any type variable, and t is any eigenterm.

Corollary 7.27 *Let z be a variable and K, L, M, and N be $\Lambda^{[]}_{\mathrm{M}}$-terms such that $K \twoheadrightarrow_\kappa L$ and $M \twoheadrightarrow_\kappa N$. Then the following holds:*

$$M\{z := K\} \twoheadrightarrow_\kappa N\{z := L\} \ .$$

Proof (Lemma 7.26). In Case i we should prove

$$M \to_\kappa N \Longrightarrow M\{x := O\} \to_\kappa N\{x := O\} \ . \tag{$*$}$$

We use induction on the structure of M:

i. $M \equiv x$. We have no N such that $x \to_\kappa N$. Hence, ($*$) holds trivially.

ii. $M \equiv [P_0 \mid P_1, \ldots, P_n]$. We divide into two subcases:

   (a) $N \equiv P_0$, *ie*, we are considering the κ-redex. Then we have

$$\begin{aligned} M\{z := O\} &\equiv [P_0 \mid P_1, \ldots, P_n]\{z := O\} \\ &= [P_0\{z := O\} \mid P_1\{z := O\}, \ldots, P_n\{z := O\}] \\ &\to_\kappa P_0\{z := O\} \\ &\equiv N\{z := O\} \ . \end{aligned}$$

   (b) $N \equiv [Q_0 \mid Q_1, \ldots, Q_n]$ where $P_i \to_\kappa Q_i$ for one i satisfying $0 \leq i \leq n$ and $Q_j \equiv P_j$ for all $j \neq i$. Using the induction hypothesis and the compatibility of $\to_\kappa$ we have:

$$\begin{aligned} M\{z := O\} &\equiv [P_0 \mid P_1, \ldots, P_n]\{z := O\} \\ &= [P_0\{z := O\} \mid P_1\{z := O\}, \ldots, P_n\{z := O\}] \\ &\overset{\mathrm{IH}}{\to}_\kappa [Q_0\{z := O\} \mid Q_1\{z := O\}, \ldots, Q_n\{z := O\}] \\ &\equiv N\{z := O\} \ . \end{aligned}$$

iii. $M \equiv \lambda x.P$, or $M \equiv P\,Q$, or $M \equiv \langle P, Q\rangle$, or $M \equiv \pi_i(P)$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \Lambda\alpha.P$, or $M \equiv P\,t$, or $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$, or $M \equiv [\![Q, t]\!]$, or $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$. In all these cases the κ-reduction is done in a subterm. They are thus similar to Case ii(b) above.

This completes the proof that we can substitute a term under a κ-reduction and hence completes Case i of Lemma 7.26. Case ii of the lemma states that we can substitute a type under a κ-reduction. The proof is similar to the proof above: one should simply replace z by α and O by t.

In Case iii of the lemma we should prove

$$M \to_\kappa N \Longrightarrow O\{z := M\} \twoheadrightarrow_\kappa O\{z := N\}\ . \qquad (\star)$$

We do this by induction on the structure of O:

i. $O \equiv x$. We have two subcases:

   (a) $x = z$. We then have

   $$O\{z := M\} = M \to_\kappa N = O\{z := N\}\ .$$

   (b) $x \neq z$. In this case we have

   $$O\{z := M\} = x = O\{z := N\}\ ,$$

   and thus $O\{z := M\} \twoheadrightarrow_\kappa O\{z := N\}$ in zero steps.

ii. $O \equiv \lambda x.P$, $O \equiv P\,Q$, $O \equiv \langle P, Q\rangle$, $O \equiv \pi_i(P)$, $O \equiv \mathrm{in}_i(P)$, $O \equiv \mathrm{case}(P,\, x.Q,\, x.R)$, $O \equiv \Lambda\alpha.P$, $O \equiv Pt$, $O \equiv [\![Q, t]\!]$, $O \equiv \mathrm{let}\,[\![x, \alpha]\!] = P\,\mathrm{in}\,Q$, or $O \equiv [P \mid \vec{Q}]$. In all these cases the substitution is propagated to the subterms and $(\star)$ follows from the induction hypothesis and the compatibility of $\to_\kappa$. We take $O \equiv \mathrm{case}(P,\, x.Q,\, x.R)$ as an illustration:

$$\begin{aligned} O\{z := M\} &\equiv \mathrm{case}(P\{z := M\},\, x.Q\{z := M\},\, x.R\{z := M\}) \\ &\overset{\text{IH}}{\twoheadrightarrow}_\kappa \mathrm{case}(P\{z := N\},\, x.Q\{z := N\},\, x.R\{z := N\}) \\ &\equiv O\{z := N\}\ . \end{aligned}$$

This exhausts the possibilities and completes the proof of Case iii stating that a reduction in a substitution results in zero or more reductions in the resulting term. It also completes the proof of the Substitution Lemma 7.26. □

Using the substitution lemma we can prove the possibility of postponing κ-reductions. Compared to the proof in [Sør97] the proof is much more involved as we have to consider eight different forms of reduction representing 22 reductions.

Lemma 7.28 *Let $M, N, O \in \Lambda^{[]}_{\mathrm{M}}$ be terms, where $M \to_\kappa N \to_{\beta\mathrm{p}\pi} O$. Then we have a $K \in \Lambda^{[]}_{\mathrm{M}}$ such that $M \twoheadrightarrow^{+}_{\beta\mathrm{p}\pi} K \twoheadrightarrow_\kappa O$.*

The lemma is illustrated by the following diagram:

Proof. We should prove that

$$M \to_\kappa N \to_{\beta\mathrm{p}\pi} O \Longrightarrow M \twoheadrightarrow^+_{\beta\mathrm{p}\pi} K \twoheadrightarrow_\kappa O \qquad (*)$$

for some $\Lambda^{[]}_{\mathrm{M}}$-term K. The proof is by induction on $M \to_\kappa N$. We use induction on the structure of M using an extension of the splitting (7) on page 39 with the case $[P \mid \vec{Q}]$. Within each major case we split on how $M \to_\kappa N \to_{\beta\mathrm{p}\pi} O$.

i. $M \equiv x$. We have no N such that $x \to_\kappa N$. Therefore $(*)$ follows trivially.

ii. $M \equiv D_1[S]$. In this case the κ-reduction is done in a subterm, while N might be the $\beta\mathrm{p}\pi$-redex. We split into subcases depending on where the $\beta\mathrm{p}\pi$-redex is in N. The first five cases cover the situation where the $\beta\mathrm{p}\pi$-reductions are a β-reduction, and N is the β-redex being reduced:

   (a) $M \equiv (\lambda x.T)\,P \to_\kappa (\lambda x.T')\,P' \to_{\beta_\lambda} T'\{x := P'\} \equiv O$, *ie*, $S \equiv \lambda x.T$ and $D_1 \equiv []\,P$. Using Corollary 7.27 we can then do the reduction

   $$(\lambda x.T)\,P \to_{\beta_\lambda} T\{x := P\} \overset{7.27}{\twoheadrightarrow_\kappa} T'\{x := P'\}\ .$$

   We thus take $K \equiv T\{x := P\}$.

   (b) $M \equiv \pi_i(\langle T_1, T_2\rangle) \to_\kappa \pi_i(\langle T_1', T_2'\rangle) \to_{\beta_\langle\rangle} T_i' \equiv O$, *ie*, we have $S \equiv \langle T_1, T_2\rangle$ and $D_1 \equiv \pi_i([])$. We reduce $\pi_i(\langle T_1, T_2\rangle)$ to T_i' by

   $$\pi_i(\langle T_1, T_2\rangle) \to_{\beta_\langle\rangle} T_i \twoheadrightarrow_\kappa T_i'\ .$$

   (c) $M \equiv \mathrm{case}(\mathrm{in}_i(T),\, x.U_1,\, x.U_2)$ reducing to $N \equiv \mathrm{case}(\mathrm{in}_i(T'),\, x.U_1',\, x.U_2')$ reducing $O \equiv U_i'\{x := T'\}$, *ie*, $S \equiv \mathrm{in}_i(T)$ and $D_1 \equiv \mathrm{case}([],\, x.U_1,\, x.U_2)$. Using Corollary 7.27 we can reduce $\mathrm{case}(\mathrm{in}_i(T),\, x.U_1,\, x.U_2)$ in the following way:

   $$\mathrm{case}(\mathrm{in}_i(T),\, x.U_1,\, x.U_2) \to_{\beta_{\mathrm{in}}} U_i\{x := T\} \overset{7.27}{\twoheadrightarrow_\kappa} U_i'\{x := T'\}\ .$$

   (d) $M \equiv (\Lambda\alpha.T)\,t \to_\kappa (\Lambda\alpha.T')\,t \to_{\beta_\Lambda} T'\{\alpha := t\} \equiv O$, *ie*, $S \equiv \Lambda\alpha.T$ and $D_1 \equiv []\,t$. Using Lemma 7.26.ii we can then do the reduction

   $$(\Lambda\alpha.T)\,t \to_{\beta_\Lambda} T\{\alpha := t\} \overset{7.26}{\twoheadrightarrow_\kappa} T'\{\alpha := t\}\ .$$

   (e) $M \equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![T, t]\!]\,\mathrm{in}\,U$ reducing to $N \equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![T', t]\!]\,\mathrm{in}\,U'$ reducing to $O \equiv \mathrm{let}\,[\![x, \alpha]\!] = [\![T', t]\!]\,\mathrm{in}\,U' \equiv O$, *ie*, $D_1 \equiv \mathrm{let}\,[\![x, \alpha]\!] = []\,\mathrm{in}\,U$ and $S \equiv [\![T, t]\!]$. Using Lemma 7.26(ii) and Corollary 7.27 we can reduce M as follows:

   $$\begin{aligned}\mathrm{let}\,[\![x, \alpha]\!] = [\![T, t]\!]\,\mathrm{in}\,U &\to_{\beta_{[\!]}} U\{\alpha := t, x := T\} \\ &\overset{7.26(\mathrm{ii})\ \&\ 7.27}{\twoheadrightarrow_\kappa} U'\{\alpha := t, x := T'\}\ .\end{aligned}$$

This exhausts the possible β-reductions. The following three cases deal with the reductions described by a one-level destructor context, *ie*, the permutative and the π-reductions.

   (f) $M \equiv D_1[\mathrm{case}(T,\, x.U_1,\, x.U_2)]$ reducing to $N \equiv D_1'[\mathrm{case}(T',\, x.U_1',\, x.U_2')]$ reducing to $O \equiv \mathrm{case}(T,\, x.D_1'[U_1'],\, x.D_1'[U_2'])$, *ie*, $S \equiv \mathrm{case}(T,\, x.U_1,\, x.U_2)$ and the reduction:

   $$\begin{aligned}D_1[\mathrm{case}(T,\, x.U_1,\, x.U_2)] &\to_\kappa D_1'[\mathrm{case}(T',\, x.U_1',\, x.U_2')] \\ &\to_{\mathrm{p}} \mathrm{case}(T,\, x.D_1'[U_1'],\, x.D_1'[U_2'])\ .\end{aligned}$$

   Using the compatibility of $\to_\kappa$ we can do the following reduction of M:

   $$\begin{aligned}D_1[\mathrm{case}(T,\, x.U_1,\, x.U_2)] &\to_{\mathrm{p}} \mathrm{case}(T,\, x.D_1[U_1],\, x.D_1[U_2]) \\ &\to_\kappa \mathrm{case}(T,\, x.D_1'[U_1'],\, x.D_1'[U_2'])\ .\end{aligned}$$

(g) $M \equiv D_1[\mathrm{let}\,\langle\!\langle x, \alpha \rangle\!\rangle = T \,\mathrm{in}\, U]$ reducing to $N \equiv D_1'[\mathrm{let}\,\langle\!\langle x, \alpha \rangle\!\rangle = T' \,\mathrm{in}\, U']$ reducing to $O\mathrm{let}\,\langle\!\langle x, \alpha \rangle\!\rangle = T' \,\mathrm{in}\, D_1'[U']$, *ie*, $S \equiv \mathrm{let}\,\langle\!\langle x, \alpha \rangle\!\rangle = T \,\mathrm{in}\, U$. Exploiting the compatibility of $\to_\kappa$ we do the reduction

$$\begin{aligned} D_1[\mathrm{let}\,\langle\!\langle x, \alpha \rangle\!\rangle = T' \,\mathrm{in}\, U'] &\to_{\mathrm{p}} \mathrm{let}\,\langle\!\langle x, \alpha \rangle\!\rangle = T \,\mathrm{in}\, D_1[U] \\ &\to_\kappa \mathrm{let}\,\langle\!\langle x, \alpha \rangle\!\rangle = T' \,\mathrm{in}\, D_1'[U'] \ . \end{aligned}$$

(h) $M \equiv D_1[[T \mid \vec{U}]] \to_\pi D_1'[[T' \mid \vec{U'}]] \to_{\mathrm{p}} [D_1'[T'] \mid \vec{U'}] \equiv O$, *ie*, $S \equiv [T \mid \vec{U}]$. As $\to_\kappa$ is compatible by definition, M can be reduced in the following way:

$$D_1[[T \mid \vec{U}]] \to_\pi [D_1[T] \mid \vec{U}] \to_\kappa [D_1'[T'] \mid \vec{U'}] \ .$$

This covers all the ways N can be the $\beta\mathrm{p}\pi$-redex. The last four subcases consider the situation where the $\beta\mathrm{p}\pi$-reduction is done in a subterm.

(i) $M \equiv D_1[S] \to_\kappa D_1'[S] \to_{\beta\mathrm{p}\pi} D_1''[S] \equiv O$, *ie*, both of the reductions are done in the one-level destructor context. Using the compatibility of $\to_\kappa$ and the induction hypothesis we have a D_1''' such that $D_1 \to_{\beta\mathrm{p}\pi} D_1''' \to_\kappa D_1''$. Using the compatibility of $\to_{\beta\mathrm{p}\pi}$ we do the following reduction of M:

$$D_1[S] \to_{\beta\mathrm{p}\pi} D_1'''[S] \to_\kappa D_1'[S] \ .$$

(j) $M \equiv D_1[S] \to_\kappa D_1[S'] \to_{\beta\mathrm{p}\pi} D_1[S''] \equiv O$, *ie*, we do the reductions in the subterm S and its contractum. By the induction hypothesis we can find a T such that $S \to_{\beta\mathrm{p}\pi} T \to_\kappa S''$ and using the compatibility of $\to_\kappa$ we can reduce M as follows:

$$D_1[S] \to_{\beta\mathrm{p}\pi} D_1[T] \to_\kappa D_1[S''] \ .$$

(k) $M \equiv D_1[S] \to_\kappa D_1'[S] \to_{\beta\mathrm{p}\pi} D_1'[S'] \equiv O$, *ie*, we do the κ-reduction in the one-level destructor context and the $\beta\mathrm{p}\pi$-reduction in the subterm S. By the compatibility of $\to_{\beta\mathrm{p}\pi}$ and $\to_\kappa$ we can simply interchange the order of the reductions, *ie*,

$$D_1[S] \to_{\beta\mathrm{p}\pi} D_1[S'] \to_\kappa D_1'[S'] \ .$$

(l) $M \equiv D_1[S] \to_\kappa D_1[S'] \to_{\beta\mathrm{p}\pi} D_1'[S'] \equiv O$, *ie*, we do κ-reduction in the subterm and the $\beta\mathrm{p}\pi$-reduction in the one-level destructor context. This is similar to Case ii(k) above.

This exhausts the possibilities of doing a κ-reduction followed by a $\beta\mathrm{p}\pi$-reduction in a filled one-level destructor context.

iii. $M \equiv \lambda x.P$, or $M \equiv \langle P, Q \rangle$, or $M \equiv \mathrm{in}_i(P)$, or $M \equiv \langle\!\langle P, t \rangle\!\rangle$. In all these cases the reductions can only be done in a subterm. The cases are hence similar to the Subcases ii(i)–ii(l).

iv. $M \equiv [P_0 \mid P_1, \ldots, P_n]$. We have three subcases:

(a) $M \equiv [P_0 \mid P_1, \ldots, P_n] \to_\kappa P_0 \to_{\beta\mathrm{p}\pi} Q$, *ie*, M is the κ-redex. By the compatibility of $\to_{\beta\mathrm{p}\pi}$ we can reduce M as follows:

$$[P_0 \mid P_1, \ldots, P_n] \to_{\beta\mathrm{p}\pi} [Q \mid P_1, \ldots, P_n] \to_\kappa Q \ .$$

(b) $M \equiv [P_0 \mid P_1, \ldots, P_n] \to_\kappa [Q_0 \mid Q_1, \ldots, Q_n] \to_{\beta\mathrm{p}\pi} [R_0 \mid R_1, \ldots, R_n] \equiv O$ where $P_i \to_\kappa Q_i \to_{\beta\mathrm{p}\pi} R_i$ for a i with $0 \leq i \leq n$ and $P_j \equiv Q_j \equiv R_j$ for all $j \neq i$. Using the induction hypothesis we can find a term T such that

$P_i \to_{\beta\mathrm{p}\pi} T \to_\kappa R_i$. If $i = 0$ we can reduce M in the following way using compatibility of $\to_\kappa$ and $\to_{\beta\mathrm{p}\pi}$:

$$[P_0 \mid P_1, \ldots, P_n] \to^{\mathrm{IH}}_{\beta\mathrm{p}\pi} [T \mid P_1, \ldots, P_n] \to_\kappa [R_0 \mid P_1, \ldots, P_n] \equiv O \ .$$

If $i \geq 1$ we can do the following reduction by the compatibility of $\to_\kappa$ and $\to_{\beta\mathrm{p}\pi}$:

$$\begin{aligned}[P_0 \mid P_1, \ldots, P_i, \ldots, P_n] &\to^{\mathrm{IH}}_{\beta\mathrm{p}\pi} [P_0 \mid P_1, \ldots, T, \ldots, P_n] \\ &\to_\kappa [P_0 \mid P_1, \ldots, R_i, \ldots, P_n] \equiv O \ .\end{aligned}$$

(c) $M \equiv [P_0 \mid P_1, \ldots, P_n] \to_\kappa [Q_0 \mid Q_1, \ldots, Q_n] \to_{\beta\mathrm{p}\pi} [R_0 \mid R_1, \ldots, R_n] \equiv O$ where $P_i \to_\kappa Q_i \equiv R_i$ for a i with $0 \leq i \leq n$ and $P_j \equiv Q_j \to_{\beta\mathrm{p}\pi} R_j$ for a $j \neq i$ and $0 \leq j \leq n$. For all other k, we take $P_k \equiv Q_k \equiv R_k$. We have four subcases depending on $i = 0$ or $j = 0$, and whether $j > i$. We take the case where $1 \leq i < j$ as an illustration and do the following reduction on M:

$$\begin{aligned}[P_0 \mid P_1, \ldots, P_i, \ldots, P_j, \ldots, P_n] &\to_{\beta\mathrm{p}\pi} [P_0 \mid P_1, \ldots, P_i, \ldots, R_j, \ldots, P_n] \\ &\to_\kappa [P_0 \mid P_1, \ldots, R_i, \ldots, R_j, \ldots, P_n] \equiv O \ .\end{aligned}$$

This exhausts the possibilities and completes the proof that κ-reductions can be postponed to after $\beta\mathrm{p}\pi$-reductions. □

We now use this lemma to prove the following

Proposition 7.29 *Let M be a $\Lambda^{[]}_{\mathrm{M}}$-term. If $M \in \mathrm{SN}_{\beta\mathrm{p}\pi}$, then $M \in \mathrm{SN}_{\beta\mathrm{p}\pi\kappa}$.*

Proof. We proceed by contraposition and assume $M \in \infty_{\beta\mathrm{p}\pi\kappa}$. We must prove that $M \in \infty_{\beta\mathrm{p}\pi}$. We do this by induction on n. We prove that for all $n \geq 0$ we can make an infinite reduction path such that the first n reductions are $\beta\mathrm{p}\pi$-reductions, *ie*,

$$M \to_{\beta\mathrm{p}\pi} M_1 \to_{\beta\mathrm{p}\pi} \ldots \to_{\beta\mathrm{p}\pi} M_{n-1} \to_{\beta\mathrm{p}\pi} M_n \to_{\beta\mathrm{p}\pi\kappa} M_{n+1} \to_{\beta\mathrm{p}\pi\kappa} \cdots \ .$$

The case $n = 0$ follows trivially from $M \in \infty_{\beta\mathrm{p}\pi\kappa}$. For $n > 0$ we know by the induction hypothesis that we have an infinite path where the first $n - 1$ reductions are $\beta\mathrm{p}\pi$-reductions, *ie*,

$$M \to_{\beta\mathrm{p}\pi} M_1 \to_{\beta\mathrm{p}\pi} \ldots \to_{\beta\mathrm{p}\pi} M_{n-1} \to_{\beta\mathrm{p}\pi\kappa} M_n \to_{\beta\mathrm{p}\pi\kappa} \cdots \ .$$

Since κ-reductions strictly decreases the term size we cannot have

$$M_{n-1} \to_\kappa M_n \to_\kappa \cdots$$

with only κ-reductions by the well-foundedness of $\mathbb{N}$. Thus we have a smallest $k \geq n-1$ such that $M_k \to_{\beta\mathrm{p}\pi} M_{k+1}$. By $k - (n-1)$ applications of Lemma 7.28 we establish that

$$M \to_{\beta\mathrm{p}\pi} M_1 \to_{\beta\mathrm{p}\pi} \cdots \to_{\beta\mathrm{p}\pi} M_{n-1} \twoheadrightarrow^{+}_{\beta\mathrm{p}\pi} M_n \to_{\beta\mathrm{p}\pi\kappa} M_{n+1} \to_{\beta\mathrm{p}\pi\kappa} \cdots \ .$$

The first n reductions of this reduction path are $\beta\mathrm{p}\pi$-reductions.

As we for an arbitrary n can find a $\beta\mathrm{p}\pi$-reduction path of length n from M we conclude that M is infinite under $\beta\mathrm{p}\pi$-reduction. This completes the proof that strong normalisation of a term M under $\beta\mathrm{p}\pi$-reduction implies strong normalisation under $\beta\mathrm{p}\pi\kappa$-reduction. □

We have now proved that we can delay the κ-reduction. The following lemma establish that we can regain the original term by κ-reductions.

Lemma 7.30 *Let $M \in \Lambda_{\mathrm{M}}$ be a term, then $\iota(M) \twoheadrightarrow_\kappa M$.*

Proof. We proof this by induction over the structure of M.

i. $M \equiv x$. We have

$$\iota(x) \overset{7.2}{=} x \to_\kappa^0 x \ .$$

ii. $M \equiv \lambda x.P$. We have by the induction hypothesis and the compatibility of $\to_\kappa$ that

$$\iota(\lambda x.P) \overset{7.2}{=} \lambda x.[\iota(P) \mid x] \overset{\text{IH}}{\twoheadrightarrow_\kappa} \lambda x.[P \mid x] \to_\kappa \lambda x.P \ .$$

iii. $M \equiv PQ$. Using the induction hypothesis and the compatibility of $\to_\kappa$, we have

$$\iota(P\,Q) \overset{7.2}{=} \iota(P)\,\iota(Q) \overset{\text{IH}}{\twoheadrightarrow_\kappa} P\,Q \ .$$

iv. $M \equiv \langle P, Q\rangle$. We have by the induction hypothesis and the compatibility of $\to_\kappa$ that

$$\iota(\langle P, Q\rangle) \overset{7.2}{=} \langle [\iota(P) \mid \iota(Q)], [\iota(Q) \mid \iota(P)]\rangle \overset{\text{IH}}{\twoheadrightarrow_\kappa} \langle [P \mid Q], [Q \mid P]\rangle \to_\kappa^2 \langle P, Q\rangle$$

v. $M \equiv \pi_i(P)$. We have by the induction hypothesis and the compatibility of $\to_\kappa$ that

$$\iota(\pi_i(P)) \overset{7.2}{=} \pi_i(\iota(P)) \overset{\text{IH}}{\twoheadrightarrow_\kappa} \pi_i(P) \ .$$

vi. $M \equiv \mathrm{in}_i(P)$. From the induction hypothesis and the compatibility of $\to_\kappa$ we have

$$\iota(\mathrm{in}_i(M)) \overset{7.2}{=} \mathrm{in}_i(\iota(M)) \overset{\text{IH}}{\twoheadrightarrow_\kappa} \mathrm{in}_i(M) \ .$$

vii. $M \equiv \mathrm{case}(P,\, x.Q,\, x.R)$. Using the induction hypothesis and the compatibility of $\to_\kappa$ We have

$$\begin{aligned}\iota(\mathrm{case}(P,\, x.Q,\, x.R)) &\overset{7.2}{=} \mathrm{case}(\iota(P),\, x.[\iota(Q) \mid x, \lambda x.\iota(R)],\, x.[\iota(R) \mid x, \lambda x.\iota(Q)]) \\ &\overset{\text{IH}}{\twoheadrightarrow_\kappa} \mathrm{case}(P,\, x.[Q \mid x, R],\, x.[R \mid x, Q]) \\ &\to_\kappa^2 \mathrm{case}(P,\, x.Q,\, x.R) \ .\end{aligned}$$

viii. $M \equiv \Lambda\alpha.P$. We do the following reduction path by the induction hypothesis and the compatibility of $\to_\kappa$:

$$\iota(\Lambda\alpha.P) \overset{7.2}{=} \Lambda\alpha.\iota(P) \overset{\text{IH}}{\twoheadrightarrow_\kappa} \Lambda\alpha.P \ .$$

ix. $M \equiv Pt$. Using the induction hypothesis and the compatibility of $\to_\kappa$ we note that

$$\iota(Pt) \overset{7.2}{=} \iota(P)t \overset{\text{IH}}{\twoheadrightarrow_\kappa} Pt \ .$$

x. $M \equiv [\![P, t]\!]$. We reduce as follows using the induction hypothesis and the compatibility of $\to_\kappa$:

$$\iota([\![P, t]\!]) \overset{7.2}{=} [\![\iota(P), t]\!] \overset{\text{IH}}{\twoheadrightarrow_\kappa} [\![P, t]\!] \ .$$

xi. $M \equiv \mathrm{let}\,\llbracket x, \alpha \rrbracket = P \,\mathrm{in}\, Q$. We can κ-reduce in the following way

$$\begin{aligned}\iota(\mathrm{let}\,\llbracket x, \alpha \rrbracket = P \,\mathrm{in}\, Q) &\overset{7.2}{=} \mathrm{let}\,\llbracket x, \alpha \rrbracket = \iota(P) \,\mathrm{in}\, [\iota(Q) \mid x] \\ &\overset{\text{IH}}{\twoheadrightarrow_\kappa} \mathrm{let}\,\llbracket x, \alpha \rrbracket = P \,\mathrm{in}\, [Q \mid x] \\ &\to_\kappa \mathrm{let}\,\llbracket x, \alpha \rrbracket = P \,\mathrm{in}\, Q\end{aligned}$$

by the induction hypothesis and the compatibility of $\to_\kappa$.

This completes the proof that $\iota(M)$ can be κ-reduced to M. □

Proposition 7.31 *Let $M \in \Lambda_{\mathrm{M}}^{[]}$ be a term. If $\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\kappa}$ then $M \in \mathrm{SN}_{\beta\mathrm{p}}$.*

Proof. We prove this proposition by contraposition. We assume that $M \in \infty_{\beta\mathrm{p}}$. We can hence find the following infinite reduction path of M

$$M \to_{\beta\mathrm{p}} M_1 \to_{\beta\mathrm{p}} M_2 \to_{\beta\mathrm{p}} \cdots \ .$$

By the previous lemma $\iota(M) \twoheadrightarrow_\kappa M$. Thus the reduction path

$$\iota(M) \twoheadrightarrow_\kappa M \to_{\beta\mathrm{p}} M_1 \to_{\beta\mathrm{p}} M_2 \to_{\beta\mathrm{p}} \cdots$$

is an infinite $\beta\mathrm{p}\kappa$-reduction of $\iota(M)$. This completes the proof that strong normalisation of $\iota(M)$ under $\beta\mathrm{p}\kappa$-reductions implies strong normalisation of M under $\beta\mathrm{p}$-reductions. □

We have now proved the three implications in (26). We collect the result in the following proposition:

Proposition 7.32 *Let $M \in \Lambda_{\mathrm{M}}$ be a term with $\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi}$. Then $M \in \mathrm{SN}_{\beta\mathrm{p}}$.*

Proof. As $\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi}$ we have by Proposition 7.29 that $\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi\kappa}$ and then also $\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi}$. Using Proposition 7.31 we conclude that $M \in \mathrm{SN}_{\beta\mathrm{p}}$. □

We can now conclude the main theorem that we are after:

Theorem 7.33 *Let M be a Λ_{M}-term. If $\underline{\iota(M)} \in \mathrm{WN}_\beta$, then $M \in \mathrm{SN}_{\beta\mathrm{p}}$.*

Proof. We assume that $\underline{\iota(M)} \in \mathrm{WN}_\beta$. By Proposition 7.13 we conclude that $\underline{\iota(M)} \in \mathrm{SN}_\beta$. Using Corollary 7.25 this implies that $\iota(M) \in \mathrm{SN}_{\beta\mathrm{p}\pi}$. By Proposition 7.32 we conclude that $M \in \mathrm{SN}_\beta$. □

The following corollary is an instance of the *Barendregt-Geuwers-Klop conjecture*:

Corollary 7.34 *If all typable terms in Λ_{M} are β-weakly normalising, then all typable terms of Λ_{M} are strongly $\beta\mathrm{p}$-normalising.*

7.5 Discussion and Conclusion

In this chapter, we have extended the technique by Xi and Sørensen to Λ_{M}, the λ-calculus corresponding to first-order minimal logic. The main achievement in this extension is the ability to handle the case-construction. There were two challenges: the erasing nature of the reduction of case-constructions and that the permutative reductions increase the term size. The former was handled in the previous chapter by extending the technique of relevant pairs previously introduced in [MN99]; the latter was done by introducing an exponential weight function in the proof of strong normalisation of the permutative reductions. We also see the crucial rôle of the uniformly normalising subset: it replaces the subset Λ^I in the extended calculus.

It should be noted that we have not reduced the question of strong normalisation to the question of weak normalisation of the *same* notion of reduction as, strictly speaking, required by the Barendregt-Geuvers-Klop conjecture. Instead, we have reduced βp-strong normalisation to weak normalisation of the simpler notion of β-reduction as we have left out the permutative reductions. A priori, this is not necessarily an achievement. To understand this, consider the extension of the ordinary λ-calculus with the constant $\mathbf{L}$ and the reduction rule $\mathbf{L}\, M \to \mathbf{L}$. In this system, the term $\mathbf{L}\, \Omega$ is not weakly normalising under β-reduction, but weakly normalising with the extra reduction rule. However, as the permutative reductions only rearrange the terms, it is reasonable to expect weak β-normalisation to follow from weak βp-normalisation. I have not proved this.

In Chapter 4 we discussed the full system of reduction rules used in Stålmarck's proof of strong normalisation of natural deduction of first-order logic and its corresponding λ-calculus. After the positive result of applying the technique by Sørensen and Xi to first-order minimal logic, it is reasonable to see why we have not considered the full system of Stålmarck. We will discuss this in this section, and also the limitations of the technique by Xi and Sørensen.

7.5.1 Classical Logic

At the heart of the technique by Sørensen and Xi is the use of a CPS-translation to embed the system with memory in the original system, such that reductions are simulated. CPS-translations of control operations, either call-by-name or call-by-value, are presented several places in the literature: Barthe et al [BHS97b], Griffin [Gri90], Murthy [Mur91], Sabry et al. [SF93], and Lawall et al. [LM00]. However, none of these translations are applicable for our purpose. The problem lies in the simulation.

To see this, we shall study the behaviour of the CPS-translations in more detail. From Definition 7.4 and Definition 7.5 of the CPS-translations it is seen that each entry consists of recursive application of the translation on the proper subterms. The results of the recursive application is filled into a context. The filling is defined such that the β-reduction on the translated term simulates the reductions.

Let us now for a moment assume that we have a compacting CPS-translation of the Δ-operator $\Delta x.M$ defined along the same pattern. Let ϕ be the function that fills the translation of M into the right context, *ie*, $M : K \equiv \phi(x, M : L, K)$ where either $L \supseteq K$ or L is a fixed term. Let us consider the situation for the Δ_λ-reduction

$$(\Delta x.x\ y)\ P \to_\Delta \Delta u.x\ y\{x := \lambda z.u\ (z\ P)\} \equiv \Delta u.(\lambda z.u\ (z\ P))\ y\ .$$

(Though this redex cannot appear as a closed term, it could be part of bigger term.) The left-hand side of the Δ_λ-redex is

$$(\Delta x.x\ y)\ P : K = \phi(x, x\ \lambda k.k\ \underline{y}\ \lambda l.l\ \underline{\underline{P}}\ L, \lambda l.l\ \underline{\underline{P}}\ K)\ , \tag{27}$$

while the right-hand side of the redex is

$$\Delta u.(\lambda z.u\ (z\ P))\ y : K = \phi(u, \left(\lambda k.k\ \underline{y}\ L'\right)\ \left(\lambda z.\lambda m.u\ \lambda n.n\ (\lambda o.z\ \lambda p.p\ \underline{\underline{P}}\ o)\ m\right), K)\ . \quad (28)$$

We have either $L = L'$ or $L \supseteq \lambda l.l\ \underline{\underline{P}}\ K$ and $L' \supseteq K$. If ϕ fulfilled the simulation requirement, (27) would β-reduce to (28). When comparing the two equations, it is quite unlikely that a function ϕ with this property can be found, if ϕ is only allowed to fill a specific context. Even if we found a context that filled with the three arguments of (27) reduced to the second argument of (28), we would be faced with the problem that the second argument of (28) would be filled into the same context (as ϕ is also applied in (28)). One could try to get around this problem by considering other CPS-translations than those of Definition 7.4 and Definition 7.5, translating to a system with $\beta\Delta$-reduction, or using another control operator. However, all the attempts I have done in these ways fails for reasons similar to the one described above.

It seems like the problem is of fundamental nature. When reducing a Δ-operator we do not substitute a subterm of the redex, but rather a term containing a subterm of the redex. As we might create a new Δ-redex when Δ-reducing we cannot simply translate a Δ-operator such that the extra term is included in the translation. The conclusion of this informal reasoning is that no matter how we define the :-translation, we cannot expect it to remove enough redexes to simulate Δ-reduction. The technique by Sørensen and Xi is therefore not extendible to classical first-order logic.

We can partially work around this problem. Rehof and Sørensen [RS94] consider the restriction $\Lambda_{\rightarrow}$ of Λ_{M} to a calculus with only application and abstraction; $\Lambda_{\rightarrow}$ is thus the simply typed λ-calculus extended with the special type constant $\bot$. For this language they prove:

Proposition 7.35 *If $\Lambda_{\rightarrow}$ is strongly β-normalising, then $\Lambda_{\rightarrow\Delta}$ is strongly $\beta\Delta$-normalising.*

With this proposition at hand, we can infer strong normalisation of $\Lambda_{\rightarrow}$ from weak normalisation of $\Lambda_{\rightarrow}$, and then use the proposition to infer strong normalisation of $\Lambda_{\rightarrow\Delta}$. We can thus infer strong normalisation of $\Lambda_{\rightarrow\Delta}$ under $\beta\Delta$-reductions from weak normalisation of β-reduction. As pointed at the start of the section, it might be easier to establish weak $\beta\Delta$-normalisation than weak β-normalisation. I have not had the time to consider whether one can deduce weak β-normalisation from weak $\beta\Delta$-normalisation. Furthermore, as stressed in [RS94], the proposition above does not extend to Λ_{M}.

A stronger technique can be found in unpublished works by Barthe et al. [BHS97c] where the technique of permutative inner interpretations [Sør97] is used to prove Proposition 7.35 using . The main idea is to reduce all internal redexes in the translated terms. In this way more redexes are removed than by the ':'-translation, and a Δ-redex becomes syntactically equal to its contractum. It should be possible to extend their proof to Λ_{M}. The extension is non-trivial as their proof exploits the fact that the internal variables occur only once in the body of an internal redex in the CPS-translations of $\Lambda_{\rightarrow}$. This does not hold when including the case-expression. I expect that some of the ideas used to prove strong normalisation of permutative reductions can be used to extend the proof. I have not had the time to check this. It might even be possible to include this step in the second step of our proof, *ie*, one could prove $\mathrm{SN}_{\beta} \Rightarrow \mathrm{SN}_{\beta\Delta\mathrm{p}\pi}$. If simply extending the reasoning above, this requires a proof that $\Delta\mathrm{p}\pi$-reductions are strongly normalising, and this is rather complicated, as Δ-reductions use substitutions (see for instance the strong normalisation proof of labelled β-reductions in [Bar84, chp. 11]). Therefore, it might be easier to keep the reasoning about the Δ-operator as the last step last step of our reasoning.

As a closing remark, it should be noted that Griffin [Gri90] and Murthy [Mur91] consider Felleisen's $\mathcal{C}$-operator [FH92], is stronger than the Δ-operator. They define

CPS-translations such that the translation of a redex reduces to the translation of a contractum, but only under specific reduction strategies, like call-by-value and call-by-name. In fact, Sabry et al. states it as an open question whether this property can be re-established for call-by-value CPS-translations [SF93, p. 315]. Private communication states that it also an open question whether it can be re-established for call-by-name.

7.5.2 Reduction of Redundant Double Negation

Stålmarck's system also has two rules for the contraction of redundant applications of the negation rule. As discussed in Section 4.3.2, their translation to a λ-calculus requires a Church-style calculus. However, even if we did so, we would not be able to extend our technique to handle these rules: The translation of the rules is

$$\begin{aligned} x^{\bot\to\bot}\, M^{\bot} &\to M^{\bot} \\ \Delta x : \bot \to \bot . M^{\bot} &\to M^{\bot} \end{aligned} \tag{29}$$

where the last rule is subject to the restriction that $x \notin \mathrm{FV}(M)$. Redexes of the second of these two rules are not preserved under ι-translation. A reasonable extension of the ι-translation to the Δ-operator will have to save a copy of the abstraction variable x, *ie*,

$$\iota(\Delta x.M) = \Delta x.[\iota(M) \mid x] \ .$$

The ι-translation of the left-hand side of the second reduction in (29) is

$$\iota(\Delta x : \bot \to \bot . M) = \Delta x : \bot \to \bot . [\iota(M) \mid x] \not\to [\iota(M) \mid x] \to_{\kappa} \iota(M) \ .$$

Apparently, there is no work around for this problem and we conclude that it is not possible to handle these reductions by our technique.

Chapter 8

Conclusion

There are several contributions of this master's thesis:

i. We have made explicit the notion of uniform normalisation, being implicit in many proofs of inferring strong normalisation from weak normalisation. In particular, we have studied the non-trivial distinction between the conservation property and uniform normalisation appearing when considering erasing reductions.

ii. We have relaxed the conditions in the characterisation by Bergstra-Klop of perpetual redexes. We no longer have an exact characterisation, but we have found a simpler proof.

iii. Furthermore, the relaxed conditions are strong enough to be used to characterise a uniformly normalising subset of the λ-calculus Λ^K. This extends the well-known subset Λ^I to include some erasing λ-reductions at the cost of using a semantic condition.

iv. We have used the understanding of uniform normalisation to extend the technique by Xi and Sørensen to infer strong normalisation from weak normalisation of Λ_{M}, the λ-calculus with a type system corresponding to first-order minimal logic as.

v. We have considered the possibilities of extending this technique to full classical first-order logic. It is my conclusion that the compacting CPS-translation is insufficient to do this. The reason is found in the fact that the Δ introduces subterms in the contractum that are not subterms of the redex. The same problems apply to similar control operators. We are therefore in need of a translation removing all internal redexes. We have indicated a way to do this by using the stronger technique of permutative inner interpretations.

To the best of my knowledge, none of the first four results have appeared previously in the literature. The fifth point has been presented before, but the problems have not been made as explicit as in this thesis. When referring to the project description in Appendix A, it is my clear opinion that I have fulfilled the objectives.

Naturally, the main direction of further work indicated from this master's thesis is to try to include the Δ-operator in the proof by adapting the technique of Permutative Inner Interpretations. It is unclear whether it is easiest to remove the Δ-reductions at the same point as the permutative reductions, or do so in an independent step.

Appendix A

Project Description of "Weak and Strong Normalization, K-redexes, and First Order Logic"

For some typed λ-calculi it is easier to prove weak normalization than strong normalization. Techniques to infer the latter from the former have been invented over the last twenty years by Nederpelt, Klop, Khasidashvili, Karr, de Groote, and Kfoury and Wells. However, these techniques infer strong normalization of one notion of reduction from weak normalization of a *more complicated* notion of reduction.

In recent papers [Sør97] techniques have been developed to infer strong normalization of a notion of reduction in a typed λ-calculus from weak normalization of the *same* notion of reduction. However, these techniques have thus far been applied only to pure λ-calculi (e.g. without pairs and sums). In this project we consider a lambda-calculus, λF, corresponding to the natural deduction presentation of full first-order classical logic including implication, conjunction, disjunction, and existential and universal quantification.

In a previous project the reduction of strong to weak normalization has been extended to the fragment of the above mentioned system containing implication and conjunction.

One of the main ideas of the reduction is a technique which also appears to be useful for other purposes, e.g. to simplify known proofs of the Bergstra-Klop characterization of perpetual K-redexes, and to give versions of the so-called conservation theorem that deal with K-redexes.

The first aim of the project is to explore these applications.

The second aim of the project is to analyze and discuss the problems involved with extending the reduction of strong to weak normalization to the full system including disjunction and quantifiers.

Bibliography

[App92] Andrew W. Appel. *Compiling with Continuations*. Cambridge University Press, Cambridge, UK, 1992. Cited on page 80.

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, second edition, 1984. Cited on pages 4, 5, 7, 12, 13, 14, 22, 49, and 111.

[Bar92] H. P. Barendregt. Lambda calculi with types. In S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 117–309. Oxford University Press, 1992. Cited on pages 2 and 33.

[BHS97a] Gilles Barthe, John Hatcliff, and Morten Heine Sørensen. CPS translations and applications: the cube and beyond. In O. Danvy, editor, *ACM SIGPLAN Workshop on Continuations*, number NS-96-13 in BRICS Notes Series, pages 4:1–31, 1997. Cited on page 82.

[BHS97b] Gilles Barthe, John Hatcliff, and Morten Heine Sørensen. A notion of classical pure type systems. In S. Brookes, M. Main, A. Melton, and M. Mislove, editors, *Mathematical foundations of programming semantics*, volume 6 of *Electronic Notes in Computer Science*. Elsevier, 1997. Cited on page 110.

[BHS97c] Gilles Barthe, John Hatcliff, and Morten Heine Sørensen. A notion of classical pure type systems. Unpublished manuscript, October 1997. Cited on page 111.

[BK82] Jan A. Bergstra and Jan Willem Klop. Strong normalization and perpetual reductions in the lambda calculus. *Elektronische Informationsverabeitung und Kybernetik*, 18:403–417, 1982. Cited on pages 3, 11, and 13.

[CC91] Felice Cardone and Mario Coppo. Type inference with recursive types: Syntax and semantics. *Information and Computation*, 92(1):48–80, May 1991. Cited on page 37.

[CD91] Charles Consel and Olivier Danvy. For a better support of static data flow. In J. Hughes, editor, *Functional Programming Languages and Computer Architecture*, volume 523 of *Lecture Notes in Computer Science*, pages 496–519, Cambridge, MA, USA, June 1991. Springer-Verlag. Cited on pages 78 and 80.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic*, volume 1 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, 1958. Cited on pages 22, 25, and 49.

[Chu41] Alonzo Church. *The Calculi of Lambda-Conversion*. Princeton University Press, 1941. Cited on page 1.

[dB72] N. G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indagationes Math.*, 34(5):381–392, 1972. Reprinted in [NGdV94]. Cited on page 4.

[dG93] Philippe de Groote. The conservation theorem revisited. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications, TLCA '93*, volume 664 of *Lecture Notes in Computer Science*, pages 163–178. Springer-Verlag, Utrecht, The Netherlands, March 1993. Cited on page 2.

[FH92] Matthias Felleisen and Robert Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science*, 103(2):235–271, September 1992. Cited on pages 34 and 111.

[FHK84] Daniel P. Friedman, Christopher T. Haynes, and Eugene Kohlbecker. Programming with continuations. In P. Pepper, editor, *Program Transformation and Programming Environments*, NATO ASI Series F V8, pages 263–274. Springer Verlag, 1984. Cited on page 81.

[Fis93] Michael J. Fischer. Lambda-calculus schemata. *Lisp and Symbolic Computation*, 6(3/4):259–288, November 1993. Cited on page 80.

[Gal93] Jean Gallier. Constructive logics Part I: A tutorial on proof systems and typed λ-calculi. *Theoretical Computer Science*, 110(2):249–339, 29 March 1993. Cited on page 25.

[Gan80] R. O. Gandy. An early proof of normalization by A. M. Turing. In Seldin and Hindley [SH80], pages 453–455. Cited on page 2.

[Gen34] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934. Cited on pages 25 and 27.

[Geu93] J. H. Geuvers. *Logics and Type Systems*. PhD thesis, University of Nijmegen, 1993. Cited on page 2.

[Gir72] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. PhD thesis, Thèse d'État, Université Paris VII, 1972. Cited on page 2.

[Gir86] Jean-Yves Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986. Cited on page 37.

[GLT89] Jean-Yves Girard, Yves Lafont, and P. Taylor. *Proofs and types*, volume 7 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 1989. Cited on pages 8, 30, 33, 34, 37, and 43.

[Gri90] Timothy G. Griffin. A formulae-as-type notion of control. In *POPL '90. Proceedings of the seventeenth annual ACM symposium on Principles of programming languages*, pages 47–58, San Francisco, CA, 17–19 January 1990. ACM Press. Cited on pages 33, 34, 110, and 111.

[HB34] D. Hilbert and P. Bernays. *Grundlagen der Mathematik I*, volume XL of *Die Grundlehren der Matematischen Wissenschaften in Einzeldarstellungen mit Besonderer Berücksichtigung der Anwendungsgebiete*. Verlag von Julius Springer, Berlin, 1934. Cited on page 25.

[HD94] John Hatcliff and Olivier Danvy. A generic account of continuation-passing styles. In *Conference Record of the 21st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 458–471, Portland, Oregon, 17–21January 1994. ACM Press. Cited on page 80.

[Hof79] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Harvester Studies in Cognitive Science. The Harvester Press, Stanford Terrace, Hassocks, Sussex, 1979. Cited on page ii.

[Hof99] Martin Hofmann. Linear types and non-size-increasing polynomial time computation. In *Proceedings, 14$^{\text{th}}$ Annual IEEE Symposium on Logic in Computer Science*, 1999. To appear in LICS '99. Cited on page 2.

[How80] W. A. Howard. The formulae-as-types notion of construction. In Seldin and Hindley [SH80], pages 479–490. Cited on page 25.

[JGS93] Neil D. Jones, Carsten K. Gomard, and Peter Sestoft. *Partial Evaluation and Automatic Program Generation*. International Series in Computer Science. Prentice-Hall International, 1993. Cited on page 80.

[Jør98] Jesper Jørgensen. Similix: A self-applicable partial evaluator for Scheme. In *Practice and experience using partial evaluators*, volume 1 of *Partial Evaluation: Practice and Theory, DIKU International Summer School*, pages 91–115. Department of Computer Science, University of Copenhagen, July 1998. Cited on page 80.

[Kar85] Michael Karr. "Delayability" in proofs of strong normalizability in the typed lambda calculus. In Hartnut Ehrig, Christiane Floyd, Mauria Nivat, and James Thatcher, editors, *Mathematical Foundations of Computer Software*, volume 185 of *Lecture Notes in Computer Science*, pages 208–222. Springer-Verlag, 1985. Cited on page 2.

[Kha91] Z. Khasidashvili. *Form Reduction Systems and Reductions of Contracted Forms and Lambda-Terms*. PhD thesis, Tbilisi State University, 1991. In Russian. Cited on page 2.

[KKR$^+$86] David Kranz, Richard Kelsey, Jonathan Rees, Paul Hudak, James Philbin, and Norman Adams. ORBIT: an optimising compiler for Scheme. *ACM SIGPLAN Notices (Proceedings of the SIGPLAN '86 Symposium on Compiler Construction, Palo Alto, California, June 25–27)*, 21(7):219–233, 1986. Cited on page 80.

[Klo80] J. W. Klop. *Combinatory Reduction Systems*. PhD thesis, Utrecht University, Amsterdam, 1980. Volume 127 of CWI Tract. Cited on pages 2, 77, and 78.

[KLR88] Donald E. Knuth, Tracy Larrabee, and Paul M. Roberts. Mathematical writing. Technical report, Stanford University, 1988. Cited on page iii.

[KW95] Assaf J. Kfoury and Joe B. Wells. New notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi. In Dexter Kozen, editor, *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 311–321, San Diego, California, 26–29 June 1995. IEEE Computer Society Press. Cited on page 2.

[LM00] Julia L. Lawall and Harry G. Mairson. Sharing continuatins: proofnets for languages with explicit control. In *The 27$^{\text{rd}}$ ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Boston, January 2000. ACM Press. To appear. Cited on page 110.

[Maz71] Antoni W. Mazurkiewicz. Proving algorithms by tail functions. *Information and Control*, 18:220–226, 1971. Cited on page 80.

[Men97] Elliott Mendelson. *Introduction to Mathematical Logic*. Wadsworth Mathematical Series. Chapman & Hall, London, fourth edition, 1997. Cited on page 27.

[Mit96] John C. Mitchell. *Foundations for Programming Languages*. MIT Press, Cambridge, 1996. Cited on pages 30, 33, and 37.

[MN99] Peter Møller Neergaard. Towards inferring strong normalization from weak normalizaiton for classical first-order logic. Manuscript, Department of Computer Science, University of Copenhagen, January 1999. Cited on pages ii, 3, 67, 79, 103, and 110.

[MNS99] Peter Møller Neergaard and Morten Heine B. Sørensen. Conservation and uniform normalisation in lambda calculi with erasing reductions. Submitted for publication, 1999. Cited on page ii.

[Mur91] Chetan R. Murthy. Classical proofs as programs: How, what and why. In J. P. Meyers, Jr. and M. J. O'Donnell, editors, *Constructivity in Computer Science, Summer Symposium*, volume 613 of *Lecture Notes in Computer Science*, pages 72–88, San Antonio, TX, June 1991. Springer-Verlag. Cited on pages 110 and 111.

[MW85] Albert R. Meyer and Mitchell Wand. Continuation semantics in typed lambda-calculi (summary). In Rohit Parikh, editor, *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 219–224, Brooklyn, June 1985. Springer-Verlag. Cited on page 84.

[Ned73] R. Nederpelt. *Strong Normalization in a Typed Lambda Calculus With Lambda Structured Types*. PhD thesis, Eindhoven, 1973. Cited on page 2.

[Neu27] J. v Neumann. Zur Hilbertschen beweistheorie. *Mathematische Zeitschrift*, 26:1–46, 1927. Cited on page 25.

[NGdV94] R. P. Nederpelt, J. H. Geuvers, and Roel C. de Vrijer, editors. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, 1994.

[Pau91] L. C. Paulson. *ML for the Working Programmer.* Cambridge University Press, first edition, paperback edition, 1991. Cited on page 30.

[Plo75] G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975. Cited on pages 78, 80, and 83.

[Pra65] Dag Prawitz. *Natural Deduction: A Proof-Theoretic Study*, volume 3 of *Stockholm Studies in Philosophy*. Almqvist & Wiksell, Stockholm, 1965. Cited on pages 2, 8, 25, 26, 28, 29, 30, and 42.

[Pra71] Dag Prawitz. Ideas and results in proof theory. In Jens Erik Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 235–307, Amsterdam, 1971. North-Holland. Cited on pages 8, 25, 26, and 77.

[Rey74] John C. Reynolds. Towards a theory of type structure. In B. Robinet, editor, *Programming Symposium, Proceedings*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425. Springer-Verlag, 1974. Cited on page 37.

[Rey93] John C. Reynolds. The discoveries of continuations. *Lisp and Symbolic Computation*, 6(3/4):233–248, November 1993. Cited on pages 78 and 80.

[RS93] Niels Jakob Rehof and Morten Heine Sørensen. The λ_Δ-calculus. Manuscript, Department of Computer Science, University of Copenhagen, October 1993. Cited on pages 33, 34, and 38.

[RS94] Niels Jakob Rehof and Morten Heine Sørensen. The λ_Δ-calculus. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software, TACS*, volume 789 of *Lecture Notes in Computer Science*, Sendai, Japan, April 1994. Springer-Verlag. Cited on pages 39, 42, 48, and 111.

[SF93] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360, 1993. Cited on pages 78, 81, 110, and 112.

[SH80] J. P. Seldin and J. R. Hindley, editors. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press Limited, 1980.

[Sør97] Morten Heine Sørensen. Strong normalization from weak normalization in typed λ-calculi. *Information and Computation*, 133(1):35–71, February 1997. Cited on pages 2, 3, 78, 84, 92, 103, 104, and 111.

[Sta74] R. Statman. *Structural Complexity of Proofs*. PhD thesis, Stanford University, Stanford, California, May 1974. Cited on page 30.

[Sta78] Richard Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, 15:225–287, 1978. Cited on pages 30 and 77.

[Stå91] Gunnar Stålmarck. Normalization theorems for full first order classical natural deduction. *Journal of Symbolic Logic*, 56(1):129–149, March 1991. Cited on pages 3, 26, 29, 30, 31, 32, and 77.

[Str97] Bjarne Stroustrup. *The C++ Programming Language: Third Edition*. Addison-Wesley, Reading, Mass., 1997. Cited on page 33.

[SU98] Morten Heine B. Sørensen and Pawel Urzycyn. Lectures on the Curry-Howard isomorphism. Technical Report 98/14, Department of Computer Science, University of Copenhagen, 1998. ISS 0107-8283. Cited on pages ii, 4, 8, 25, and 30.

[Tai67] W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967. Cited on page 2.

[Tai75] W. W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 240–251. Springer-Verlag, 1975. Cited on page 2.

[Tak95] Masako Takahashi. Parallel reductions in λ-calculus. *Information and Computation*, 118(1):120–127, April 1995. Cited on pages 3, 49, 54, 57, and 65.

[TS96] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1996. Cited on pages 25, 30, 33, and 34.

[Tv88] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics, Vol. 1*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, 1988. Cited on page 26.

[van86] Dirk van Dalen. Intuitionistic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume III: Alternatives to Classical Logic*, volume 166 of *Synthese Library*, pages 225–339. D. Reidel Publ. Co., Dordrecht, 1986. Cited on page 26.

[vRSSX99] Femke van Raamsdonk, Paula Severi, Morten Heine B. Sørensen, and Hongwei Xi. Perpetual reductions in λ-calculus. *Information and Computation*, 149(2):173–225, 1999. Cited on pages 11, 12, 13, 15, and 20.

[Wat95] Bill Watterson. *The Calvin and Hobbes Tenth Anniversary Book*. Warner Books, 1995. Cited on page iii.

[Xi97] Hongwei Xi. Weak and strong beta normalisations in typed λ-calculi. In Phillippe de Groote and J. Roger Hindley, editors, *Third International Conference on Typed Lambda Calculi and Applications, TLCA '97*, volume 1210 of *Lecture Notes in Computer Science*, Nancy, France, April 1997. Springer-Verlag. Cited on pages 2, 3, and 78.