Types intersections non-idempotents pour raffiner la normalisation forte avec des informations quantitatives

Alexis Bernadet

▶ **To cite this version:**

Alexis Bernadet. Types intersections non-idempotents pour raffiner la normalisation forte avec des informations quantitatives. Langage de programmation [cs.PL]. Ecole Doctorale Polytechnique, 2014. Français. ⟨NNT : ⟩. ⟨tel-01099657⟩

HAL Id: tel-01099657

https://pastel.hal.science/tel-01099657v1

Submitted on 5 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

École Polytechnique

PhD Thesis

Non idempotent-intersection types to refine strong normalisation with quantitative information

Author: Alexis Bernadet

Jury:

Mme Simona Ronchi Della Rocca	Rapporteur
Mr Lorenzo Tortora de Falco	Rapporteur
Mme Mariangiola Dezani-Ciancaglini	Examinateur
Mr Stéphane Lengrand	Directeur de thèse
Mr Benjamin Werner	Directeur de thèse
Mme Delia Kesner	Président

Defended: October 6, 2014

Abstract

We study systems of non-idempotent intersection types for different variants of the λ-calculus and we discuss properties and applications. Besides the pure λ-calculus itself, the variants are a λ-calculus with explicit substitutions and a λ-calculus with constructors, matching and a fixpoint operator. The typing systems we introduce for these calculi all characterize strongly normalising terms. But we also show that, by dropping idempotency of intersections, typing a term provides quantitative information about it: a trivial measure on its typing tree gives a bound on the size of the longest β-reduction sequence from this term to its normal form. We explore how to refine this approach to obtain finer results: some of the typing systems, under certain conditions, even provide the exact measure of this longest β-reduction sequence, and the type of a term gives information on the normal form of this term. Moreover, by using filters, these typing systems can be used to define a denotational semantics.

Acknowledgements

I would like to thank Stéphane Lengrand, Benjamin Werner, Mariangiola Dezani-Ciancaglini, Delia Kesner, Simona Ronchi Della Rocca, Lorenzo Tortora de Falco, David Monniaux, Thierry Coquand, Philippe Chassignet, Benjamin Smith, Sylvie Putot, Julie Bernauer, Jean Goubault-Larrecq, Paul-André Melliès, Pierre-Louis Curien, Alain Prouté and my family.

Contents

Acknowledgements . . . 1

1 Introduction **4**
1.1 Generalities . . . 4
1.2 About the λ-calculus . . . 7
1.3 Brief description of the Chapters . . . 8
1.4 Introduction to α-equivalence . . . 9

2 A simple typing system of non-idempotent intersection types for pure λ-calculus **12**
2.1 Introduction . . . 12
2.2 Syntax and operational semantics . . . 13
2.3 The typing system . . . 14
2.3.1 Types . . . 14
2.3.2 Contexts . . . 16
2.3.3 Rules . . . 17
2.4 Soundness . . . 18
2.5 Semantics and applications . . . 20
2.5.1 Denotational semantics . . . 20
2.5.2 Example: System F . . . 24
2.6 Completeness . . . 26
2.7 Conclusion . . . 33

3 Intersection types with explicit substitutions **35**
3.1 Introduction . . . 35
3.2 Syntax . . . 35
3.3 Typing judgments . . . 37
3.4 Soundness . . . 38
3.5 Special property of typing trees: Optimality . . . 38
3.6 Completeness . . . 39
3.7 Complexity . . . 42
3.8 Conclusion . . . 43

4 A big-step operational semantics via non-idempotent intersection types **45**
4.1 Introduction . . . 45
4.2 Basic definitions and properties . . . 47
4.2.1 Syntax . . . 47
4.2.2 Intersection types and contexts . . . 49
4.2.3 Typing system . . . 51
4.3 Characterisation of the typing system . . . 53
4.3.1 Soundness . . . 53
4.3.2 Completeness . . . 54

4.4 Refined soundness . . . 55
4.4.1 Complexity . . . 55
4.4.2 Viewing optimal typing as a big-step semantics . . . 56
4.5 Alternative systems . . . 58
4.5.1 Variant with no information about the normal form . . . 58
4.5.2 Obtaining the exact normal form . . . 58
4.6 Conclusion . . . 59

5 Strong normalisation in a calculus with constructors and fixpoints via non-idempotent intersection types 60
5.1 Introduction . . . 60
5.2 Calculus . . . 62
5.2.1 Definition of the calculus . . . 62
5.2.2 Refined notion of reduction . . . 64
5.3 Strong normalisation . . . 68
5.3.1 Intersection types . . . 69
5.3.2 Soundness . . . 71
5.3.3 Completeness . . . 71
5.4 Conclusion . . . 72
5.5 Confluence . . . 72
5.6 Example in Caml . . . 74

6 Conclusion 75

A Full proofs 80

B A simple presentation of de Bruijn indices 102
B.1 Definitions . . . 102
B.2 Binding . . . 102
B.3 Substitutions . . . 104
B.4 β-reduction . . . 106
B.5 Semantics . . . 107

Chapter 1

Introduction

1.1 Generalities

What is a programming language ? A programming language is a set of sequences of symbols where each of these sequences describes more or less an action or a computation. Such sequence of symbols is called a program of this programming language. It is needed to build computer software. For real world programs, these sequences of symbols are written in one or more files in the computer.

The most obvious programming language is the machine language which can be directly interpreted by a computer. There are two main reasons why most software are not written in the machine language:

- It is really hard to write anything with it, even small programs.
- There exists a different machine language for each computer architecture. Therefore, we may need to rewrite the entire program if we want to port it on an different platform.

Therefore, almost every other software are written in programming languages that are more human readable, more scalable, more portable, more machine-independent and, sometimes, safer. Programs written in these languages cannot be directly interpreted by a computer.

- There can be a software that transforms a program written in some language to a program written in the machine language. This software is called a compiler.
- There can be a software that interpret a program written in some language and do what the program is supposed to do. This is called an interpreter.

Of course, things can be a little more complicated than that:

- A program in some language can be compiled to another language which is not the machine language and, to be executed, the new program obtained requires an interpreter.
- Compilation can be in several steps: the program can be written in several intermediate languages until we reach the target language. This is useful when creating a programming language and its compiler to not have to worry about technical problems that have already been solved by more competent people. Solving these problems ourselves may be called reinventing the wheel. For examples, lots of high-level programming language are compiled to a language called C. Most compiler can compile C to assembler which is a programming language that is just above the machine language in terms of abstraction. Finally, the assembler program can be compiled to machine language and then, we can execute it. Therefore, if someone wants to write the compiler of

a new high-level language, they just have to be able to compile it to C, and then they can have quite fast executables.

When writing a program, someone might wonder what its meaning is or what it does. We usually have these two approaches:

- We give an intuitive meaning to each construction of the language. The meaning given is not very different from what someone would have written if we were trying to explain in a tutorial or a lesson what a construction does. Of course, this has no mathematical value.
- We say that the meaning of a program is what it does when executed (after compilation if any). This is also not usually suited to do mathematical reasoning on programs.

Therefore, if we want to have mathematical results about programs (such as correctness), we need to define formally the meaning of a program. This is called the semantics of a programming language. Ensuring the correctness of a program is useful when there are lives as stake (programs in a plane, etc...) or even when there is money at stake (for example, the crash of a program used in a bank). Even when it is just saving debugging time (time spent searching and correcting mistakes that occurs during the execution of a program) it is well worth it. There are two main approaches:

- The operational semantics: There are syntactic rules (usually reduction rules) that describe what a program does. In some ways, it is paraphrasing the interpreter.
- The denotational semantics: We create a mathematical object and we use it as a model of the programming language: A program is interpreted as "something" in this mathematical object (for example, it can be an element of a set or a morphism in a category). For a given programming language, there are usually lots of possible models.

The difference between these two is similar to the difference between "what it is" (denotational semantics) and "what it does" (operational semantics).

When studying semantics, we usually start with a minimalistic/toy language. Most of the time, when we are complexifying the language we can easily extend and adapt the definitions, results and proofs. The λ-calculus is one of the most well known minimalistic languages.

Also, when working with terms and programs, we prefer to work with an abstract expression tree instead of a sequence of symbols. For example, the following sequence of 7 symbols:

$$3 \times (4 + 5)$$

can be transformed into the following abstract expression tree:

$$\dfrac{\overline{3} \qquad \dfrac{\overline{4} \qquad \overline{5}}{+}}{\times}$$

Even if we are manipulating abstract expression trees, we usually write them in one line (in the above example we write $3 \times (4+5)$). Transforming a sequence of symbols to an abstract expression tree is done by something called a lexer and another thing called a parser. Usually, when working on semantics we ignore this aspect.

What is typing in a programming language ?

Originally, types are used in a programming language as an indication for the compiler about how to represent data. For example, an integer (which has type "int") has not the same representation as a floating number (which has type "float").

As computers got faster, a certain number of programming languages (and quite a few of them, like Python, became popular), choose to get rid of the types in the

language. This leads to a small cost at runtime (we need to keep information about the types) in exchange of the possibility to write shorter programs.

On the other side, some people (such as the developers of the ML family of programming languages) consider that having information about the types at compilation time is a bonus because it makes the catching of mistakes at compilation time easier.

The argument that not having types makes programs shorter to write is not necessary valid: a compiler can try to guess the types in a program. This is called typing inference. Hence, we have the best of both worlds. Actually, the more the types are expressive, the harder it is to infer them. For example, Girard's System F cannot be inferred [Wel96]. Moreover, most of the systems using intersection types[CD78], such as the ones studied in this thesis cannot be inferred.

What is logic ?

Set Theory has imposed itself as the main language for doing mathematics. However, at the beginning of the 20th century, mathematicians realized that certain reasoning with sets could lead to contradictions (for example, Russel's paradox [vH02]: the set of all the sets is not a set).

Therefore, it seemed necessary to sanitize the mathematical foundations: deciding what was a valid (or sound) proof and what was not. Of course, to do this, we need to give a mathematical definition of a proof.

Such proofs can be represented in a computer. Hence, we can write a Proof Assistant: we write a proof in a computer, and the Proof Assistant checks whether or not the proof is sound. Also, the writing of some parts of the proofs can be automated.

What is intuitionistic logic ?

Alan Turing proved [Tur36] that some functions (defined mathematically) could not be calculated by a computer. It is not a question of time or memory available. It turns out that the reason we are in this situation is that we can use in mathematics the principle of excluded middle (for every formula A, A is true or the negation of A is true). By dropping excluded middle we we are in a new logic called intuitionistic logic and when we have excluded middle we says that we are in classical logic. The idea of intuitionistic logic is from Brouwer [vH02] and has been developed by Heyting [Hey71]. One of the most modern way of defining intuitionistic logic is Topos Theory: an Lawvere's Elementary Topos [LS09] can be used as a model of intuitionistic logic. On the one hand, we can prove fewer theorems in intuitionistic logic (compared to classical logic). On the other hand:

- A definition (resp. theorem) that is used (resp. holds) in classical logic might not be intuitionistically friendly (resp. be proved in intuitionistic logic). However, it may be possible to have a definition (resp. theorem) that is equivalent in classical logic but is also intuitionistically friendly (resp. holds in intuitionistic logic) and not more complicated than the original one.
- When something is proved in intuitionistic logic it gives more information than in classical logic: the proof is constructive. In particular, if we can define a function in intuitionistic logic, then this function is computable. This is the reason intuitionistic logic is also called constructive logic.

Moreover, we can find a correspondence between intuitionistic proofs and functional programs: This is called the Curry-Howard correspondence [GLT89].

What is linear logic ?

When a program runs, resources (data in memory) can be duplicated or erased. It is possible to have a calculus/programming language that has restrictions on duplicating/erasing resources. By interpreting this through the Curry-Howard correspondence, we can build a corresponding logic where we have restrictions on using a proof several times or not using it at all: This is called linear logic and it was

invented by Girard [GLT89]. One particularity of linear logic, is that we have two "and" and two "or". Linear logic is strongly related to the topic of this thesis, namely the idea of using intersection types that are not idempotent ($A \cap A$ is not equivalent to A, see the next section).

1.2 About the λ-calculus

The λ-calculus is a minimalistic programming language created by Church [Chu85]. In this programming language there is no distinction between programs and data. A program M in the pure λ-calculus is called a λ-term and is either:

- A variable x, y, z, ...
- An abstraction of the form $\lambda x.M$. Intuitively, $\lambda x.M$ describes the function that maps x to M. For example, $\lambda x.x$ is the identity function.
- An application of the form MN where M is the function and N is the argument.

The fact that this programming language looks simplistic can be balanced as follows:

- It is possible to encode in this calculus, usual data structures (such as booleans, integers, lists, trees, etc ...) and more advanced constructions. For example, the integer n is encoded as the λ-term $\lambda f.\lambda x.f^n x$ (with $f^0 x = x$ and $f^{n+1} x = f(f^n x)$). We can also encode recursion and then prove that the λ-calculus is Turing complete: anything that can be "computed" can be written in λ-calculus. This is both a blessing and a curse.
- It is possible to enrich the λ-calculus with usual data structures and more advanced constructions expressed as new primitives of the language rather as encodings. Most definitions, theorems and proofs of the λ-calculus without this enrichment (also called the pure λ-calculus) can be adapted. By taking a step further, we can design real world programming languages based on the λ-calculus. These languages are called functional programming languages. OCaml and Haskell are among those.

The syntax of λ-calculus is very simple, but we still have to define what a λ-term formally means. This is defining the semantics of the λ-calculus. The main idea is that $(\lambda x.M)N$ has the same meaning as $M\{x := N\}$ where $M\{x := N\}$ is M where we "replaced every x in M by N". Moreover, going from $(\lambda x.M)N$ to $M\{x := N\}$ is a step in the computation. An occurrence $(\lambda x.M_1)M_2$ inside a term M is called a β-redex; and replacing an occurrence of $(\lambda x.M_1)M_2$ by $M_1\{x := M_2\}$ is called a β-reduction. When we cannot do any β-reduction, the computation is over and the term M that cannot be reduced is the result of the computation. If from a term M we can reach a term M' that cannot be reduced, then M' is called the normal form of M.

When defining this semantics formally, several questions and problems arise:

- We have to formally define what the term $M\{x := N\}$.is. The definition is quite subtle and relies on an equivalence relation that identifies some terms (like $\lambda x.x$ and $\lambda y.y$ which both describe the identity function). This relation is called α-equivalence. Usually, λ-terms are considered modulo α-equivalence. The easiest way of defining and manipulating this equivalence is probably by using nominal logic techniques (see Section 1.4).
- If there are several β-redexes in a term M, then there are several ways of doing a β-reduction. Therefore, the normal form does not seem to always be unique. Actually, it is unique, and it is a corollary of the confluence property: If M can reach M_1 and M can reach M_2 then there exists M_3 such that M_1

and M_2 can reach M_3. It is possible to prove confluence by using something called parallel reductions [CR36].

- The question about whether or not the normal form exists: Some λ-terms do not have a normal form. For example, $(\lambda x.xx)(\lambda x.xx)$ can only reduce to itself. By the fact that the λ-calculus is Turing complete, we cannot avoid the fact that the computation of some terms does not terminate and we cannot even decide by a program if a term does terminate or not (that would give a solution to the Halting Problem and contradict Turing's result[Tur36]). Even when the normal form exists, by the fact that a λ-term can be reduced several ways, we can consider the two following cases:
  - There exists a β-reduction sequence to the normal form. This is called weak normalisation.
  - We do not have an infinite β-reduction sequence: Any β-reduction strategy terminates. This is called strong normalisation. By the fact that a λ-terms has a finite number of β-redexes, strong normalisation of M is equivalent to having a longest reduction sequences from M.

  For example, $(\lambda y.z)((\lambda x.xx)(\lambda y.xx))$ (with $y \neq z$) is weakly normalising (the normal form is z) but is not strongly normalising (it reduces to itself). In this thesis, we are more interested in the strong normalisation property.

We have described the operational semantics of the λ-calculus. It is also possible to give a devotional semantics. For example, we can interpret each λ-term as an element of a Scott Domain [Sco82a] (which is a partially ordered set that satisfies some properties).

If we want to easily manipulate λ-terms and ensure they have some properties (such as strong normalisation), we can use types. If M has some type A (written $M : A$) then, by a Soundness Theorem, we can prove that M has some properties described by A. The typing judgment is defined with induction rules.

The most basic typing system for λ-calculus are the simple types where a type is either a constant τ or an arrow $A \to B$. If we have $M : A \to B$, it means that M is a "function" that goes from A to B. The main idea is that:

- If $M : A \to B$ and $N : A$, then $MN : B$.
- If $M : B$ with $x : A$, then $\lambda x.M : A \to B$.

Actually, more formally, the typing judgments are of the form $\Gamma \vdash M : A$ where Γ is something called a context and indicates the type of each variables in M. Simple types have the following interesting property: If M is typable, then M is strongly normalising.

It is possible to enrich the types with an intersection construction: If A is a type and B is a type, then $A \cap B$ is a type. The main idea is that if $M : A$ and $M : B$, then $M : A \cap B$. Usually, we have an equivalence relation $\approx$ on types (to have associativity, commutativity of intersection etc ...). We also usually have $A \cap A \approx A$: the intersection is idempotent.

In most of the thesis, we are going to drop the idempotency of intersection and we are going to see how it can give more information about quantitative properties on a λ-term: If $M : A \cap A$, then M will be used twice as a program of type A.

1.3 Brief description of the Chapters

In **Chapter 2** ,we define a typing system for pure λ-calculus. The types used in this system are intersection types: If M is of type A and M is of type B, then M is of a type written $A \cap B$. The particularity of this typing system is that the intersection is not idempotent: $A \cap A$ is not the same as A. This leads to various

properties and applications that we study: First, similarly to the case of idempotent intersection types, this typing system characterizes strongly normalising terms: a term is typable if and only if it is strongly normalising. Also, we can use this typing system to define a denotational semantics of λ-calculus where values are filters of types. This semantics can serve as a tool to prove strong normalisation in other typing systems. We illustrate this with the example of System F. Finally, we have a complexity result: A trivial measure on a typing tree of a term M gives us an upper bound on the size of the longest β-reduction sequences from M.

The complexity result obtained in Chapter 2 is an inequality result. We would rather have an equality result, extracting from the typing tree of a term the exact size of the longest reduction sequences. For this, in **Chapter 3** we refine the λ-calculus: Instead of the pure λ-calculus, we work with λS which is a λ-calculus with explicit substitutions $M[x := N]$. In this calculus the erasure of a sub-term can be postponed and this simplifies the study of various results. In particular, we extract from typing trees the exact size of the longest reduction sequences, defined in terms of its number of B-steps ($(\lambda x.M)N \to_B M[x := N]$) rather than β-steps (which are not primitive reductions of this calculus). Moreover, for this improved complexity result, we have to restrict ourselves to certain types of typing trees that we call "optimal".

In **Chapter 4**, we also refine the complexity result of Chapter 2, but still for the pure λ-calculus (unlike Chapter 3). However, we now use an enriched grammar for types and we add extra rules to the typing system. This typing system still characterises strongly normalising normal terms and has its own definition of an *optimal* typing tree; these optimal typing trees provide the exact size the longest β-reduction sequences and can be considered as the derivation of a big-step operational semantics. In particular types describe the structure of normal forms.

In **Chapter 5**, we extend the typing system of Chapter 2 to a calculus that is closer to a real world programming language. In particular, this calculus provides constructors and matching to easily use algebraic data types, and a fixpoint operator to write recursive functions. As in the previous chapters, a term is typable if and only if it is strongly normalising.

The chapters can almost be read mostly independently. but we recommended to read Chapter 2 before the other ones.

This thesis is written in the implicit framework of classical set theory (ZFC) with first order logic; however when we work with λ-terms (of the various calculi), we consider them up to α-equivalence. In Section 1.4, we are going to show what this formally means for the pure λ-calculus. We will not explicitly develop these technical points for the other calculi handled in this thesis (other we could), but we will only write "we consider terms up to α-equivalence".

1.4 Introduction to α-equivalence

In this section we show how α-equivalence can be formally defined in the framework of the pure λ-calculus. The main idea is that α-equivalence is defined by using permutations and is inspired by nominal logic [Pit03].

Assume we have an infinite (countable) set Var.

Definition 1 (Permutations)**.**

A permutation π is a bijection from Var to Var such that $\mathsf{Dom}(\pi)$ defined by:

$$\mathsf{Dom}(\pi) := \{x \in Var \mid \pi(x) \neq x\}$$

is finite.

Perm is the set of the permutations and forms a group:
id defined by $id(x) := x$ is a permutation and $id^{-1} = id$ and $\mathsf{Dom}(id) = \emptyset$.

$$\frac{}{x \approx_\alpha x} \qquad \frac{M \approx_\alpha M' \quad N \approx_\alpha N'}{MN \approx_\alpha M'N'}$$

$$\frac{\pi \in \text{Perm} \quad \pi(x) = y \quad \pi.M \approx_\alpha M' \quad \text{fv}(M) \cap \mathsf{Dom}(\pi) \subseteq \{x\}}{\lambda x.M \approx_\alpha \lambda y.M'} \ (\lambda)$$

Figure 1.1: α-equivalence

For all $\pi_1, \pi_2 \in Perm$, $\pi_2 \circ \pi_1$ defined by $(\pi_2 \circ \pi_1)(x) = \pi_2(\pi_1(x))$ is a permutation and $(\pi_1\pi_2)^{-1} = \pi_2^{-1}\pi_1^{-1}$ and $\mathsf{Dom}(\pi_2 \circ \pi_1) \subseteq \mathsf{Dom}(\pi_1) \cup \mathsf{Dom}(\pi_2)$.

For all $x, y \in Var$, $< x, y >$ defined by:

$$\begin{array}{lcll} < x, y > (x) & = & y & \\ < x, y > (y) & = & x & \\ < x, y > (z) & = & z & (z \neq x, z \neq y) \end{array}$$

is a permutation and $< x, y >^{-1} =< x, y >$ and $\mathsf{Dom}(< x, y >) = \{x, y\}$.

Moreover, $< x, y > =< y, x >$.

Definition 2 (λ-terms and α-equivalence)**.**

λ-terms are defined with the following grammar:

$$\begin{array}{lcl} M, N & ::= & x \mid \lambda x.M \mid MN \\ x & \in & Var \end{array}$$

Free variables of M is a finite set of variables defined by induction on M as follows:

$$\begin{array}{lcl} fv(x) & := & \{x\} \\ fv(MN) & := & fv(M) \cup fv(N) \\ fv(\lambda x.M) & := & fv(M) - \{x\} \end{array}$$

The action of π on M is defined by induction on M as follows:

$$\begin{array}{lcl} \pi.x & := & \pi(x) \\ \pi.(MN) & := & \pi.M\pi.N \\ \pi.(\lambda x.M) & := & \lambda\pi(x).\pi.M \end{array}$$

The size of a term M is defined by induction on M as follows:

$$\begin{array}{lcl} |x| & := & 1 \\ |MN| & := & |M| + |N| \\ |\lambda x.M| & := & |M| + 1 \end{array}$$

α-equivalence is defined with the rules of Figure 1.1. In rule (λ) the permutation π is a witness of the α-equivalence.

Lemma 1 (Properties of λ-terms)**.**

1. *$id.M = M$ and $\pi_2.\pi_1.M = (\pi_2 \circ \pi_1).M$.*
2. *$fv(\pi.M) = \{\pi(x) \mid x \in fv(M)\}$. Therefore if $x \in fv(\pi.M)$, then $\pi^{-1}(x) \in fv(M)$.*
3. *If $M \approx_\alpha M'$ then $\pi.M \approx_\alpha \pi.M'$, $fv(M) = fv(M')$ and $|M| = |M'|$.*
4. *If $M \approx_\alpha M'$, then $\lambda x.M \approx_\alpha \lambda x.M'$.*
5. *$\approx_\alpha$ is an equivalence relation.*
6. *If for all $x \in fv(M)$, $\pi(x) = x$, then $M \approx_\alpha \pi.M$.*
7. *If for all $x \in fv(M)$, $\pi_1(x) = \pi_2(x)$, then $\pi_1.M \approx_\alpha \pi_2.M$*
8. *If $x \neq y$ and $y \notin fv(M)$ and $< x, y > .M \approx_\alpha M'$, then $\lambda x.M \approx_\alpha \lambda y.M'$.*
9. *If $\lambda x.M \approx_\alpha \lambda x.M'$, then $M \approx_\alpha M'$.*

10. *If* $\lambda x.M \approx_\alpha \lambda y.M'$ *and* $x \neq y$*, then* $y \notin fv(M)$ *and* $< x, y > .M \approx_\alpha M'$.

Proof. 1. By induction on M.

2. By induction on M.

3. By induction on $M \approx_\alpha M'$. Only one case is not trivial: We have $\lambda x.M_1 \approx_\alpha \lambda y.M_2$ with π_1 as a witness. We then prove that $\lambda\pi(x).\pi.M_1 \approx_\alpha \lambda\pi(y).\pi.M_2$ with the rule (λ) and the permutation $\pi \circ \pi_1 \circ \pi^{-1}$ as witness.

4. We use rule (λ) with the permutation id.

5. - We prove reflexivity by induction on M. For the $\lambda x.M_1$ case, we use the item 4.
   - We prove symmetry by induction on $M \approx_\alpha M'$. For the (λ) rule with π as a witness of $\lambda x.M_1 \approx_\alpha \lambda y.M_2$, we use π^{-1} as a witness of $\lambda y.M_2 \approx_\alpha \lambda x.M_1$.
   - We prove transitivity by induction on M_1 that if $M_1 \approx_\alpha M_2$ and $M_2 \approx_\alpha M_3$, then $M_1 \approx_\alpha M_3$. There is only one case that is not trivial: We have $\lambda x.M_1 \approx_\alpha \lambda y.M_2$ with π_1 as a witness and $\lambda y.M_2 \approx_\alpha \lambda z.M_3$ with π_2 as a witness. We use $\pi_2 \circ \pi_1$ as a witness of $\lambda x.M_1 \approx_\alpha \lambda z.M_3$.

     In particular, we have $\pi_1.M_1 \approx_\alpha M_2$ and $\pi_2.M_2 \approx_\alpha M_3$. Therefore, $M_1 \approx_\alpha \pi^{-1}.M_2$ and $\pi_1^{-1}.M_2 \approx_\alpha \pi_1^{-1}.\pi_2^{-1}.M_3$. By induction hypothesis on M_1, we have $M_1 \approx_\alpha \pi_1^{-1}.\pi_2^{-1}M_3$. Hence $(\pi_2 \circ \pi_1).M_1 \approx_\alpha M_3$.

6. By induction on M. In particular, for $\lambda x.M_1$, we use the rule (λ) with π as a witness.

7. For all $x \in \mathrm{fv}(M)$, we have $\pi_1(x) = \pi_2(x)$. Hence, $(\pi_2^{-1} \circ \pi_1)(x) = \pi_2^{-1}(\pi_1(x)) = x$. By item 6, we have $(\pi_2^{-1} \circ \pi_1).M \approx_\alpha M$. Therefore, $\pi_1.M \approx_\alpha \pi_2.M$.

8. We use the rule (λ) with $< x, y >$ as a witness.

9. Corollary of item 6.

10. Corollary of item 7.

□

Lemma 1.5 allows us to quotient the grammar of λ-terms by α-equivalence; more generally the properties will be used implicitly throughout the thesis.

In the rest of the thesis, the work concerning α-equivalence, even for other calculi, is assumed and terms are considered up to α-equivalence.

Another possibility is to work with De Bruijn indices and then we do not have to define the α-equivalence. However, the proofs and formalism are heavier. A glimpse on how to work with De Bruijn is given in Appendix B. It would have been possible to write this thesis with De Bruijn indices.

Chapter 2

A simple typing system of non-idempotent intersection types for pure λ-calculus

2.1 Introduction

Intersection types were introduced in [CD78], extending the simply-typed λ-calculus with a notion of finite polymorphism. This is achieved by a new construct $A \cap B$ in the syntax of types and new typing rules such as:

$$\frac{M : A \quad M : B}{M : A \cap B}$$

where $M : A$ denotes that a term M is of type A.

One of the motivations was to characterise strongly normalising (SN) λ-terms, namely the property that a λ-term can be typed if and only if it is strongly normalising. Variants of systems using intersection types have been studied to characterise other evaluation properties of λ-terms and served as the basis of corresponding semantics [Lei86, Ghi96, DCHM00, CS07].

This chapter refines with quantitative information the property that typability characterises strong normalisation. Since strong normalisation ensures that all reduction sequences are finite, we are naturally interested in identifying the length of the longest reduction sequences. We do this with a typing system that is very sensitive to the usage of resources when λ-terms are reduced.

This system results from a long line of research inspired by Linear Logic [Gir87]. The usual logical connectives of, say, classical and intuitionistic logic, are decomposed therein into finer-grained connectives, separating a *linear* part from a part that controls how and when the structural rules of *contraction* and *weakening* are used in proofs. This can be seen as resource management when hypotheses, or more generally logical formulae, are considered as resource.

The Curry-Howard correspondence, which originated in the context of intuitionistic logic [How80], can be adapted to Linear Logic [Abr93, BBdH93], whose resource-awareness translates to a control of resources in the execution of programs (in the usual computational sense). From this, have emerged some versions of linear logic that capture pastime functions [BM03, Laf04, GR07]. Also from this has emerged a theory of λ-calculus with resource, with semantical support (such as the differential λ-calculus) [ER03, BEM10].

In this line of research, de Carvalho [dC05, dC09] obtained interesting measures

$$\frac{}{(\lambda x.M)N \to_\beta M\{x := N\}} \qquad \frac{M \to_\beta M'}{\lambda x.M \to_\beta \lambda x.M'}$$

$$\frac{M \to_\beta M'}{MN \to_\beta M'N} \qquad \frac{N \to_\beta N'}{MN \to_\beta MN'}$$

Figure 2.1: β-reduction

of reduction lengths in the λ-calculus by means of *non-idempotent* intersection types (as pioneered by [KW99, NM04]).

Intersections were originally introduced as idempotent, with the equation $A \cap A = A$ either as an explicit quotient or as a consequence of the system. This corresponds to the understanding of the judgement $M : A \cap B$ as follows: M can be used as data of type A or data of type B. But the meaning of $M : A \cap B$ can be strengthened in that M **will** be used **once** as data of type A and **once** as data of type B. With this understanding, $A \cap A \neq A$, and dropping idempotency of intersections is thus a natural way to study control of resources and complexity. Using this, de Carvalho [dC09] has shown a correspondence between the size of the typing derivation tree and the number of steps taken by a Krivine machine to reduce the term. This relates to the length of linear head-reductions, but if we remain in the realm of intersection systems that characterise strong normalisation, then the more interesting measure is the length of the longest reduction sequences. In this chapter we get a result similar to de Carvalho's, but with the measure corresponding to strong normalisation.

In Section 2.2 we formally give the syntax and operational semantics of the λ-calculus. In Section 2.3, we define a system with non-idempotent intersection types. In Section 2.4, we prove that if a term is typable then it is SN (soundness). In Section 2.5, we use the typing system to define a denotational semantics and we show examples of applications. In Section 2.6, we prove that if a term is strongly normalising then it is typable.

2.2 Syntax and operational semantics

In this section we formally give the syntax and operational semantics of the λ-calculus.

Definition 3 (λ-terms)**.** *Terms are defined in the usual way with the following grammar:*

$$M, N ::= x \mid \lambda x.M \mid MN$$

Free variables $fv(M)$ and substitutions $M\{x := N\}$ are defined in the usual way and terms are considered up to α-equivalence.

The notion of reduction is the usual β-reduction:

Definition 4 (β-reduction)**.** *$M \to_\beta M'$ is defined inductively by the rules of Figure 2.1.*

To make some proofs and theorems about strong normalisation more readable, we use the following notations:

Definition 5 (Strongly normalising terms)**.** *Assume n is an integer.*

*We write **SN** for the set of strongly normalising terms for $\to_\beta$.*

$$\frac{}{F \approx F} \qquad \frac{}{A \cap B \approx B \cap A} \qquad \frac{}{(A \cap B) \cap C \approx A \cap (B \cap C)}$$

$$\frac{}{A \cap (B \cap C) \approx (A \cap B) \cap C} \qquad \frac{A \approx A' \quad B \approx B'}{A \cap B \approx A' \cap B'} \qquad \frac{}{\omega \approx \omega} \qquad \frac{A \approx B \quad B \approx C}{A \approx C}$$

Figure 2.2: Equivalence between types

We write $\boldsymbol{SN}_{=n}$ *for the set of strongly normalising terms for* $\to_\beta$ *such that the length of longest* β*-reduction sequences is equal to* n*.*

The set $SN_{\leq n}$ *is defined by:*

$$SN_{\leq n} := \bigcup_{m \leq n} \boldsymbol{SN}_{=m}$$

2.3 The typing system

In Section 2.3.1, we define the intersection types. In Section 2.3.2, we define contexts. In Section 2.3.3, we define the typing system and give basic properties.

2.3.1 Types

In this chapter, intersection types are defined as follows:

Definition 6 (Intersection types)**.**

F-types, A-types and U-types are defined with the following grammar:

$$\begin{array}{lcl} F, G & ::= & \tau \mid A \to F \\ A, B, C & ::= & F \mid A \cap B \\ U, V & ::= & A \mid \omega \end{array}$$

where τ *ranges over an infinite set of atomic types.*

With this grammar, $U \cap V$ *is defined if and only if* U *and* V *are A-types.*

Therefore, by defining $A \cap \omega := A$, $\omega \cap A := A$ *and* $\omega \cap \omega := \omega$*, we have* $U \cap V$ *defined for all* U *and* V*.*

Some remarks:

- The property that, in an arrow $A \to B$, B is not an intersection, is the standard restriction of *strict* types [vB95] and is used here to prove Lemma 5.1.
- ω is a device that allows synthetic formulations of definitions, properties and proofs: U-types are not defined by mutual induction with A-types and F-types, but separately, and we could have written the chapter without them (only with more cases in statements and proofs).

For example, $(\tau \to \tau) \cap \tau$ is an A-type.

To prove Subject Reduction and Subject Expansion (Theorems 1 and 5) by using Lemmas 6 and 12, we have to define equivalence $\approx$ and inclusion $\subseteq$ between types. Here is the formal definitions and basic properties (notice that we do not have $A \approx A \cap A$):

Definition 7 (Equivalence between types)**.**

Assume U *and* V *are* U*-types. We define* $U \approx V$ *with the rules given in Figure 2.2.*

The fact that $\approx$ is an equivalence relation can be easily proved (Lemma 2.4). Therefore, adding rules for reflexivity, symmetry and transitivity is superfluous and

only adds more cases to treat in the proofs of statements where such a relation is assumed (e.g. Lemma 5.2).

Lemma 2 (Properties of $\approx$)**.**

1. *Neutrality of ω:* $U \cap \omega = \omega \cap U = U$.
2. *Strictness of F-types: If* $U \approx F$, *then* $U = F$.
3. *Strictness of ω: If* $U \approx \omega$, *then* $U = \omega$.
4. $\approx$ *is an equivalence relation.*
5. *Commutativity of $\cap$:* $U \cap V \approx V \cap U$
6. *Associativity of $\cap$:* $U_1 \cap (U_2 \cap U_3) \approx (U_1 \cap U_2) \cap U_3$
7. *Stability of $\cap$: If* $U \approx U'$ *and* $V \approx V'$, *then* $U \cap V \approx U' \cap V'$.
8. *If* $U \cap V = \omega$, *then* $U = V = \omega$.
9. *If* $U \cap V \approx U$, *then* $V = \omega$.

Proof.

1. Straightforward.
2. By induction on $U \approx F$.
3. Straightforward.
4.
   - Reflexivity: We prove that $U \approx U$ by induction on U.
   - Symmetry: We prove by induction on $U \approx V$ that if $U \approx V$, then $V \approx U$.
   - Transitivity: Straightforward.
5. Straightforward.
6. Straightforward.
7. Straightforward.
8. Straightforward.
9. For all U, we construct $\varphi(U)$ defined by induction on U as follows:
   $$\begin{array}{lcl} \varphi(F) & := & 1 \\ \varphi(A \cap B) & := & \varphi(A) + \varphi(B) \\ \varphi(\omega) & := & 0 \end{array}$$
   By induction on U, if $\varphi(U) = 0$, then $U = \omega$
   We also have $\varphi(U \cap V) = \varphi(U) + \varphi(V)$.
   By induction on $U \approx V$, if $U \approx V$, then $\varphi(U) = \varphi(V)$.
   If $U \cap V \approx U$: Then, $\varphi(U \cap V) = \varphi(U)$. Therefore, $\varphi(U) + \varphi(V) = \varphi(U)$. Hence, $\varphi(V) = 0$. Therefore, $V = \omega$.

□

Definition 8 (Sub-typing)**.**
Assume U and V are U-types. We write $U \subseteq V$ if and only if there exists a U-type U' such that $U \approx V \cap U'$.

Lemma 3 (Properties of $\subseteq$)**.**

1. $\subseteq$ *is a partial pre-order and* $\approx$ *is the equivalence relation associated to it:* $U \subseteq V$ *and* $V \subseteq U$ *if and only if* $U \approx V$.
2. *Projections:* $U \cap V \subseteq U$ *and* $U \cap V \subseteq V$.

3. *Stability of* $\cap$*: If* $U \subseteq U'$ *and* $V \subseteq V'$*, then* $U \cap V \subseteq U' \cap V'$*.*
4. *Greatest element:* $U \subseteq \omega$*.*

Proof. Straightforward. □

We could have represented intersection types with multisets (and correspondingly used the type inclusion symbol $\subseteq$ the other way round). We chose to keep the standard notation $A \cap B$, with the corresponding inclusion satisfying $A \cap B \subseteq A$, since these are interpreted as set intersection and set inclusion in some models (e.g. realisability candidates). This way, we can also keep the equivalence relation explicit in the rest of the chapter, which gives finer-grained results. For example, the equivalence only appears where it is necessary and the proof of Lemma 5.2 shows the mechanism that propagates the equivalence through the typing trees. These presentation choices are irrelevant to the key ideas of the chapter.

2.3.2 Contexts

To define typing judgements (Definition 10), we need to define contexts and give their basic properties. We naturally define pointwise the notion of equivalence and inclusion for contexts. More formally:

Definition 9 (Contexts)**.**
A context Γ *is a total map from the set of variables to the set of* U*-types such that the domain of* Γ *defined by:*

$$\mathsf{Dom}(\Gamma) := \{x \mid \Gamma(x) \neq \omega\}$$

is finite.

$\cap$, $\approx$ *and* $\subseteq$ *for contexts are defined pointwise:*

$$\begin{array}{lcl} (\Gamma \cap \Delta)(x) & := & \Gamma(x) \cap \Delta(x) \\ \Gamma \approx \Delta & \Leftrightarrow & \forall x, \Gamma(x) \approx \Delta(x) \\ \Gamma \subseteq \Delta & \Leftrightarrow & \forall x, \Gamma(x) \subseteq \Delta(x) \end{array}$$

Notice that if $\mathsf{Dom}(\Gamma)$ *and* $\mathsf{Dom}(\Delta)$ *are finite, then* $\mathsf{Dom}(\Gamma \cap \Delta)$ *is finite. Therefore* $\Gamma \cap \Delta$ *is indeed a context in the case where* Γ *and* Δ *are contexts.*

The empty context $()$ *is defined as follows:* $()(x) := \omega$ *for all* x*.*

Assume Γ *is a context,* $x_1, \ldots, x_n$ *are distinct variables and* $U_1, \ldots U_n$ *are* U*-types such that for all* i*,* $x_i \notin \mathsf{Dom}(\Gamma)$*. Then, the context* $(\Gamma, x_1 : U_1, \ldots, x_n : U_n)$ *is defined as follows:*

$$\begin{array}{lcll} (\Gamma, x_1 : U_1, \ldots, x_n : U_n)(x_i) & := & U_i & \\ (\Gamma, x_1 : U_1, \ldots, x_n : U_n)(y) & := & \Gamma(y) & (\forall i, y \neq x_i) \end{array}$$

$(\Gamma, x_1 : U_1, \ldots, x_n : U_n)$ *is indeed a context and* $((), x_1 : U_1, \ldots, x_n : U_n)$ *is written* $(x_1 : U_1, \ldots, x_n : U_n)$*.*

Lemma 4 (Properties of contexts)**.**

1. $\approx$ *for contexts is an equivalence relation.*
2. $\subseteq$ *for contexts is a partial pre-order and* $\approx$ *is its associated equivalence relation:* $\Gamma \subseteq \Delta$ *and* $\Delta \subseteq \Gamma$ *if and only if* $\Gamma \approx \Delta$*.*
3. *Projections:* $\Gamma \cap \Delta \subseteq \Gamma$ *and* $\Gamma \cap \Delta \subseteq \Delta$*.*
4. *Alternative definition:* $\Gamma \subseteq \Delta$ *if and only if there exists a context* Γ' *such that* $\Gamma \approx \Delta \cap \Gamma'$*.*
5. *Commutativity of* $\cap$*:* $\Gamma \cap \Delta \approx \Delta \cap \Gamma$*.*
6. *Associativity of* $\cap$*:* $(\Gamma_1 \cap \Gamma_2) \cap \Gamma_3 \approx \Gamma_1 \cap (\Gamma_2 \cap \Gamma_3)$*.*
7. *Stability of* $\cap$*: If* R *is either* $\approx$ *or* $\subseteq$*,* $\Gamma \; R \; \Gamma'$ *and* $\Delta \; R \; \Delta'$*, then* $\Gamma \cap \Delta \; R \; \Gamma' \cap \Delta'$*.*

$$\frac{}{x : F \vdash^0 x : F}\,(Var) \qquad \frac{\Gamma \vdash^n M : A \quad \Delta \vdash^m M : B}{\Gamma \cap \Delta \vdash^{n+m} M : A \cap B}\,(\cap) \qquad \frac{}{\vdash^0 M : \omega}\,(\omega)$$

$$\frac{\Gamma, x : U \vdash^n M : F \quad A \subseteq U}{\Gamma \vdash^n \lambda x.M : A \to F}\,(Fun) \qquad \frac{\Gamma \vdash^n M : A \to F \quad \Delta \vdash^m N : A}{\Gamma \cap \Delta \vdash^{n+m+1} MN : F}\,(App)$$

Figure 2.3: Typing rules

8. *Greatest context:* $\Gamma \subseteq ()$.
9. $(\Gamma, x : U) \subseteq \Gamma$.

Proof. Straightforward. □

2.3.3 Rules

We now have all the elements to present the typing system:

Definition 10 (Typing system)**.**
Assume Γ *is a context,* M *is a term,* n *is an integer, and* U *is a* U*-type. The judgement* $\Gamma \vdash^n M : U$ *is inductively defined by the rules given in Figure 2.3.*
We write $\Gamma \vdash M : U$ *if there exists* n *such that* $\Gamma \vdash^n M : U$.

Some remarks:

- In $\Gamma \vdash^n M : U$, n is the number of uses of the rule (App) and it is the trivial measure on typing trees that we use.
- The rule (ω) is a device to simplify some definitions, proofs and properties: it is independent from the other rules and this chapter could have been written without it (only with more cases in statements and proofs). In particular, $\vdash M : \omega$ gives no information about M and, for example, Theorem 2 uses A instead of U.
- Condition $A \subseteq U$ in rule (Fun) is called *subsumption* and is used to make Subject Reduction (Theorem 1) hold when the reduction is under a λ: After a subject reduction, U can turn into a different U' without changing A.
- Another advantage in having the type ω for the presentation of the typing system: Without the notation U or V, we would have to duplicate the abstraction rule that types $\lambda x.M$ (one case where $x \in \mathrm{fv}(M)$ and one case where $x \notin \mathrm{fv}(M)$). That would make two rules instead of one.
- Like the other judgements defined with induction rules, when we write "By induction on $\Gamma \vdash M : U$", means "By induction on the derivation proving $\Gamma \vdash M : U$".

Lemma 5 (Basic properties of typing)**.**

1. $\Gamma \vdash^n M : U \cap V$ *if and only if there exist* Γ_1, Γ_2, n_1 *and* n_2 *such that* $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} M : U$ *and* $\Gamma_2 \vdash^{n_2} M : V$.
2. *If* $\Gamma \vdash^n M : U$ *and* $U \approx V$, *then there exists* Δ *such that* $\Gamma \approx \Delta$ *and* $\Delta \vdash^n M : V$.
3. *If* $\Gamma \vdash^n M : U$ *and* $U \subseteq V$, *then there exist* Δ *and* m *such that* $\Gamma \subseteq \Delta$, $m \leq n$ *and* $\Delta \vdash^m M : V$.
4. *If* $\Gamma \vdash M : A$, *then* $\mathsf{Dom}(\Gamma) = fv(M)$.

5. *If* $\Gamma \vdash M : U$, *then* $\mathsf{Dom}(\Gamma) \subseteq fv(M)$.

Proof.

1. Straightforward.
2. By induction on $U \approx V$.
3. Corollary of 1 and 2.
4. By induction on $\Gamma \vdash M : A$.
5. Corollary of 4.

□

2.4 Soundness

As usual for the proof of Subject Reduction, we first prove a substitution lemma:

Lemma 6 (Substitution lemma)**.**
If $\Gamma, x : U \vdash^n M : A$ *and* $\Delta \vdash^m N : U$, *then there exists* Γ' *such that* $\Gamma' \approx \Gamma \cap \Delta$ *and* $\Gamma' \vdash^{n+m} M\{x := N\} : A$.

Proof. By induction on $\Gamma, x : U \vdash M : A$. The measure of the final typing tree is $n + m$ because, by the fact that the intersection types are non-idempotent, this proof does not do any duplications of the rule (*App*).

More precisely:

- For $\dfrac{}{x : F \vdash^0 x : F}$ with $\Gamma = ()$, $n = 0$, $M = x$, $U = F$ and $A = F$: We have $x\{x := N\} = N$. By hypothesis, $\Delta \vdash^m N : U$. Therefore, $\Delta \vdash^m M\{x := N\} : F$ with $n + m = m$ and $\Gamma \cap \Delta = () \cap \Delta = \Delta$.
- For $\dfrac{}{y : F \vdash^0 y : F}$ with $y \neq x$, $\Gamma = (y : F)$, $n = 0$, $M = y$, $U = \omega$, and $A = F$: By hypothesis, $\Delta \vdash^m N : \omega$. Hence, $\Delta = ()$ and $m = 0$. We have $y\{x := N\} = y$. Therefore, $y : F \vdash^0 M\{x := N\} : F$ with $n + m = 0$ and $\Gamma \cap \Delta = (y : F) \cap () = (y : F)$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M : A_1 \quad \Gamma_2, x : U_2 \vdash^{n_2} M : A_2}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2} M : A_1 \cap A_2}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$ and $A = A_1 \cap A_2$: By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$. By Lemma 5.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M\{x := N\} : A_1$ and $\Gamma'_2 \vdash^{n_2+m_2} M\{x := N\} : A_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2} M\{x := N\} : A_1 \cap A_2$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 = n + m$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n M_1 : F \quad B \subseteq V}{\Gamma, x : U \vdash^n \lambda y.M_1 : B \to F}$ with $M = \lambda y.M_1$, $x \neq y$, $y \notin \mathrm{fv}(N)$ and $A = B \to F$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exists Γ' such that $\Gamma' \approx (\Gamma, y : V) \cap \Delta$ and $\Gamma' \vdash^{n+m} M_1\{y := N\} : F$. By Lemma 5.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V \approx V'$. Hence, $B \subseteq V'$. Therefore, $\Gamma'' \vdash^{n+m} \lambda y.M_1\{x := N\} : B \to F$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M_1 : B \to F \quad \Gamma_2, x : U_2 \vdash^{n_2} M_2 : B}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2+1} M_1 M_2 : F}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $U = U_1 \cap U_2$, $M = M_1 M_2$ and $A = F$: We have $(M_1 M_2)\{x :=$

$N\} = M_1\{x := N\}M_2\{x := N\}$. By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$. By Lemma 5.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M_1\{x := N\} : B \to F$ and $\Gamma'_2 \vdash^{n_2+m_2} M_2\{x := N\} : B$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2+1} M_1M_2 : F$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 + 1 = n + m$. □

Theorem 1 (Subject Reduction)**.**
If $\Gamma \vdash^n M : A$ *and* $M \to_\beta M'$*, then there exist* Γ' *and* n' *such that* $\Gamma \subseteq \Gamma'$*,* $n > n'$ *and* $\Gamma' \vdash^{n'} M' : A$*.*

Proof. First by induction on $M \to_\beta M'$, then by induction on A.

In particular, for the base case of the β reduction ($(\lambda x.M1)M2 \to_\beta M_1\{x := M_2\}$), we use Lemma 6.

For the case $\lambda x.M_1 \to_\beta \lambda x.M'_1$ with $M_1 \to_\beta M'_1$, we use the fact that in the rule (Fun), there can be a subsumption. Therefore, the change of the type of x can be caught by the rule (App).

More precisely:

- If A is of the form $A_1 \cap A_2$: Then, there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} M : A_1$ and $\Gamma_2 \vdash^{n_2} M : A_2$. By induction hypothesis on $(M \to_\beta M', A_1)$ and $(M \to_\beta M', A_2)$, there exist Γ'_1, Γ'_2, n'_1 and n'_2 such that $\Gamma_1 \subseteq \Gamma'_1$, $\Gamma_2 \subseteq \Gamma'_2$, $n_1 > n'_1$, $n_2 > n'_2$, $\Gamma'_1 \vdash^{n'_1} M' : A_1$ and $\Gamma'_2 \vdash^{n'_2} M' : A_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n'_1+n'_2} M' : A_1 \cap A_2$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma'_2$ and $n = n_1 + n_2 > n'_1 + n'_2$.

- For $\dfrac{}{(\lambda x.M_1)M_2 \to_\beta M_1\{x := M_2\}}$ with $M = (\lambda x.M_1)M_2$ and A is of the form F:

  Therefore, there exist Γ_1, Γ_2, n_1, n_2, B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} \lambda x.M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$: Then, there exists U such that $B \subseteq U$ and $\Gamma_1, x : U \vdash^{n_1} M_1 : F$. By Lemma 5.3, there exist Γ'_2 and n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n_2 \geq n'_2$ and $\Gamma'_2 \vdash^{n'_2} M_2 : U$. By Lemma 6, there exists Γ' such that $\Gamma' \approx \Gamma_1 \cap \Gamma'_2$ and $\Gamma' \vdash^{n_1+n'_2} M_1\{x := M_2\} : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2 \approx \Gamma'$ and $n = n_1 + n_2 + 1 > n_1 + n_2 \geq n_1 + n'_2$.

- For $\dfrac{M_1 \to_\beta M'_1}{\lambda x.M_1 \to_\beta \lambda x.M'_1}$ with $M = \lambda x.M_1$ and A is of the form F: There exist B, G and U such that $F = B \to G$, $B \subseteq U$ and $\Gamma, x : U \vdash^n M_1 : G$. By induction hypothesis, there exists Γ'_1 and n' such that $(\Gamma, x : U) \subseteq \Gamma'_1$, $n > n'$ and $\Gamma'_1 \vdash^{n'} M'_1 : G$. There exist a unique Γ' and a unique U' such that $\Gamma'_1 = (\Gamma', x : U')$. Therefore, $\Gamma \subseteq \Gamma'$ and $U \subseteq U'$. Hence, $B \subseteq U'$. Therefore $\Gamma' \vdash^{n'} \lambda x.M'_1 : B \to G$.

- For $\dfrac{M_1 \to_\beta M'_1}{M_1M_2 \to_\beta M'_1M_2}$ with $M = M_1M_2$ and A is of the form F: There exist $\Gamma_1, \Gamma_2, n_1, n_2$ and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$. By induction hypothesis, there exist Γ'_1 and n'_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $n_1 > n'_1$ and $\Gamma'_1 \vdash^{n'_1} M'_1 : B \to F$. Therefore, $\Gamma'_1 \cap \Gamma_2 \vdash^{n'_1+n_2+1} M'_1M_2 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma_2$ and $n = n_1 + n_2 + 1 > n'_1 + n_2 + 1$.

- For $\dfrac{M_2 \to_\beta M'_2}{M_1M_2 \to_\beta M_1M'_2}$ with $M = M_1M_2$ and A is of the form F: There exist $\Gamma_1, \Gamma_2, n_1, n_2$ and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} M_1 : B \to F$

and $\Gamma_2 \vdash^{n_2} M_2 : B$. By induction hypothesis, there exist Γ_2' and n_2' such that $\Gamma_2 \subseteq \Gamma_2'$, $n_2 > n_2'$ and $\Gamma_2' \vdash^{n_2'} M_2' : B$. Therefore, $\Gamma_1 \cap \Gamma_2' \vdash^{n_1+n_2'+1} M_1 M_2' : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2'$ and $n = n_1 + n_2 + 1 > n_1 + n_2' + 1$.

□

In Theorem 1, we have $n > n'$ because, by the fact that types are non-idempotent, we do not do any duplications in the proof of Subject Reduction. Therefore, by Subject Reduction, for each β-reduction, the measure of the typing tree strictly decreases and then, we have Soundness as a corollary.

Theorem 2 (Soundness)**.**
If $\Gamma \vdash^n M : A$*, then* $M \in SN_{\leq n}$*.*

Proof. Corollary of Theorem 1: We prove by induction on n that if $\Gamma \vdash^n M : A$ then $M \in \mathrm{SN}_{\leq n}$.

Let M' be a term such that $M \to_\beta M'$. By Theorem 1, there exist Γ' and n' such that $n' < n$ and $\Gamma' \vdash^{n'} M' : A$. By induction hypothesis, $M' \in \mathrm{SN}_{\leq n'}$. Hence, $M' \in \mathrm{SN}_{\leq n-1}$ because $n' \leq n - 1$.

Therefore, $M \in \mathrm{SN}_{\leq n}$. □

Theorem 2 gives more information than a usual soundness theorem: If a term is typable, the term is not only strongly normalising, but the measure on a typing tree is a bound on the size longest β-reduction sequences from this term to its normal form. One of the reasons we have just a bound is that the measure of the typing tree can decrease more than 1. This is usually what happens when the β-reduction erases a sub-term. Having a better result than just a bound is discussed in Chapters 3 and 4.

2.5 Semantics and applications

In this section we show how to use non-idempotent intersection types to simplify the methodology of [CS07], which we briefly review here:

The goal is to produce modular proofs of strong normalisation for various *source typing systems*. The problem is reduced to the strong normalisation of a unique *target system* of intersection types, chosen once and for all. This is done by interpreting each term t as the set $[\![t]\!]$ of the intersection types that can be assigned to t in the target system. Two facts then remain to be proved:

1. if t can be typed in the source system, then $[\![t]\!]$ is not empty
2. the target system is strongly normalising

The first point is the only part that is specific to the source typing system: it amounts to turning the interpretation of terms into a filter model of the source typing system. The second point depends on the chosen target system: as [CS07] uses a system of *idempotent* intersection types (extending the simply-typed λ-calculus), their proof involves the usual reducibility technique [Gir72, Tai75]. But this is somewhat redundant with point 1 which uses similar techniques to prove the correctness of the filter model with respect to the source system.

In this chapter we propose to use *non-idempotent* intersection types for the target system, so that point 2 can be proved with simpler techniques than in [CS07] while point 1 is not impacted by the move.

2.5.1 Denotational semantics

The following filter constructions only involve the syntax of types and are independent from the chosen target system.

Definition 11 (Values)**.**
A value v is a set of U-types such that:

- $\omega \in v$.
- *If* $U \in v$ *and* $U \subseteq V$*, then* $V \in v$.
- *If* $U \in v$ *and* $V \in v$*, then* $U \cap V \in v$.

We write $\mathscr{D}$ the set of values.

While our intersection types differ from those in [CS07] (in that idempotency is dropped), the stability of a filter under type intersections makes it validate idempotency (it contains A if and only if it contains $A \cap A$, etc). This makes our filters very similar to those in [CS07], so we can plug-in the rest of the methodology with minimal change.

Definition 12 (Examples of values)**.**

1. *Let* $\bot := \{\omega\}$
2. *Let* $\top$*, the set of all* U*-types.*
3. *Assume* α *is a set of* F*-types. Then* $< \alpha >$ *is the set of* U*-types defined with the following rules:*
$$\frac{F \in \alpha}{F \in < \alpha >} \qquad \frac{}{\omega \in < \alpha >} \qquad \frac{A \in < \alpha > \quad B \in < \alpha >}{A \cap B \in < \alpha >}$$
4. *Assume* $u, v \in \mathscr{D}$*. Let* $uv := < \{F \mid \exists A \in v, (A \to F) \in u\} >$.

Lemma 7 (Properties of values)**.**
Assume $u, v \in \mathscr{D}$.

1. *If for all* $F \in u$ *we have* $F \in v$*, then* $u \subseteq v$.
2. *If for all* F*,* $F \in u$ *if and only if* $F \in v$*, then* $u = v$.
3. $\bot$ *is the smallest element of* $\mathscr{D}$ *and* $\top$ *is the biggest element of* $\mathscr{D}$.
4. *If* $U \in v$ *and* $U \approx V$*, then* $V \in v$.
5. $< \alpha >$ *is the smallest element* v *of* $\mathscr{D}$ *such that* $\alpha \subseteq v$.
6. $F \in < \alpha >$ *if and only if* $F \in \alpha$.
7. $uv \in \mathscr{D}$ *and* $F \in uv$ *if and only if there exists* A *such that* $(A \to F) \in u$ *and* $A \in v$.
8. *We have* $u\bot = \bot = \bot u$.
9. *If* $v \neq \bot$*, then* $\top v = \top$.

Proof.

1. We prove by induction on U, that for all $U \in u$, we have $U \in v$:
   - For ω: By Definition 11, we have $\omega \in v$.
   - For F: By hypothesis, we have $F \in v$.
   - For $A \cap B$: We have $A \cap B \subseteq A$ and $A \cap B \subseteq B$. By Definition 11, we have $A \in u$ and $B \in u$. By induction hypothesis, we have $A \in v$ and $B \in v$. By Definition 11, we have $A \cap B \in v$.
2. Corollary of 1.
3. First, we prove that $\bot \in \mathscr{D}$:
   - We have $\omega \in \bot$.

- Assume $U \in \bot$ and $U \subseteq V$. Therefore, $U = \omega$ and $\omega \subseteq V$. We also have $V \subseteq \omega$. Hence, $V \approx \omega$. Therefore, $V = \omega$ and $V \in \bot$.
- Assume $U \in \bot$ and $V \in \bot$. Therefore, $U = \omega$ and $V = \omega$. Hence, $U \cap V = \omega \cap \omega = \omega \in \bot$.

Therefore, $\bot \in \mathscr{D}$.

Assume $v \in \mathscr{D}$ and $U \in \bot$. Then, $U = \omega$ and $U \in v$. Hence, $\bot \subseteq v$. Therefore, $\bot$ is the smallest element of $\mathscr{D}$.

It is straightforward to prove that $\top$ is the biggest element of $\mathscr{D}$.

4. Assume $U \in v$ and $U \approx V$. Then, $U \subseteq V$. Therefore, by Definition 11, $V \in v$.
5. To prove that $< \alpha > \in \mathscr{D}$ we first need the following:
   - We can notice that $U \cap V \in < \alpha >$ if and only if $U \in < \alpha >$ and $V \in < \alpha >$.
   - We can prove by induction $U \approx V$, that if $U \in < \alpha >$ and $U \approx V$ then $V \in < \alpha >$.

   Therefore, we can prove that $< \alpha > \in \mathscr{D}$:
   - We have $\omega \in < \alpha >$.
   - Assume $U \in < \alpha >$ and $U \subseteq V$. Hence, there exists U' such that $U \approx V \cap U'$. Therefore, $V \cap U' \in < \alpha >$. Hence, $V \in < \alpha >$.
   - If $U \in < \alpha >$ and $V \in < \alpha >$, then $U \cap V \in < \alpha >$.

   Therefore, $< \alpha > \in \mathscr{D}$ and by definition we have $\alpha \subseteq < \alpha >$.

   Assume $v \in \mathscr{D}$ such that $\alpha \subseteq v$. We can prove by induction on $U \in < \alpha >$ that for all $U \in < \alpha >$, we have $U \in v$. Hence, $< \alpha > \subseteq v$.

   Therefore, $< \alpha >$ is the smallest element v of $\mathscr{D}$ such that $\alpha \subseteq v$.
6. Straightforward.
7. The fact that $uv \in \mathscr{D}$ is a corollary of 5. The rest is a corollary of 6.
8. By 3, we have $\bot \subseteq u\bot$ and $\bot \subseteq \bot u$.
   - Assume $F \in u\bot$. Then, by 7, there exists A such that $(A \to F) \in u$ and $A \in \bot$. Contradiction. Therefore, $F \in \bot$.

     By 1, $u\bot \subseteq \bot$.
   - By a similar proof, $\bot u \subseteq \bot$.

   Therefore, $u\bot = \bot u = \bot$.
9. By 3, we have $\top v \subseteq \top$.

   Assume $F \in \top$. By the fact that $v \neq \bot$, there exists $A \in v$. Therefore, $(A \to F) \in \top$ and $F \in \top v$.

   By 1, we have $\top \subseteq \top v$.

   Then, we can conclude.

□

Definition 13 (Environements)**.**
An environment ρ *is a total map from the set of variables to* $\mathscr{D}$.
Assume Γ *is a context, then we write* $\Gamma \in \rho$ *if and only if:*

$$\forall x, \Gamma(x) \in \rho(x)$$

The environment $(\rho, x \leftarrow v)$ *is defined as follow:*

$$\begin{array}{lcll} (\rho, x \leftarrow v)(x) & := & v & \\ (\rho, x \leftarrow v)(y) & := & \rho(y) & (x \neq y) \end{array}$$

Lemma 8 (Properties of environments)**.**

1. *If* $\Gamma \in \rho$ *and* $U \in v$*, then* $(\Gamma, x : U) \in (\rho, x \leftarrow v)$.

2. *If* $\Gamma \in (\rho, x \leftarrow v)$, *then there exist* Γ' *and* U *such that* $\Gamma = (\Gamma', x : U)$, $\Gamma' \in \rho$ *and* $U \in v$.
3. *We have* $() \in \rho$.
4. *If* $\Gamma \in \rho$ *and* $\Gamma \subseteq \Delta$, *then* $\Delta \in \rho$.
5. *If* $\Gamma \in \rho$ *and* $\Delta \in \rho$, *then* $\Gamma \cap \Delta \in \rho$.

Proof. By the fact that $\Gamma \in \rho$ is defined point-wise. □

Definition 14 (Semantics of a term)**.**
Assume M *is a term and* ρ *an environment.*
Let $[\![M]\!]_\rho := \{U \mid \exists \Gamma \in \rho, \Gamma \vdash M : U\}$.

Lemma 9 (Properties of the semantics of a term)**.**

1. *We have* $[\![M]\!]_\rho \in \mathscr{D}$.
2. *If* $[\![M]\!]_\rho \neq \bot$, *then* M *is strongly normalising.*
3. *We have* $[\![x]\!]_\rho = \rho(x)$.
4. *We have* $[\![MN]\!]_\rho = [\![M]\!]_\rho [\![N]\!]_\rho$.
5. *We have* $[\![\lambda x.M]\!]_\rho v \subseteq [\![M]\!]_{(\rho, x \leftarrow v)}$.
6. *If* $x \in fv(M)$ *or* $v \neq \bot$, *then* $[\![\lambda x.M]\!]_\rho v = [\![M]\!]_{(\rho, x \leftarrow v)}$.
7. *If* $M \to_\beta M'$, *then* $[\![M]\!]_\rho \subseteq [\![M']\!]_\rho$.

Proof.

1. We prove that $[\![M]\!]_\rho \in \mathscr{D}$:
   - We have $\vdash M : \omega$ and $() \in \rho$ (by Lemma 8.3). Therefore, $\omega \in [\![M]\!]_\rho$.
   - Assume $U \in [\![M]\!]_\rho$ and $U \subseteq V$. Then, there exists $\Gamma \in \rho$ such that $\Gamma \vdash M : U$. Therefore, there exists Γ' such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash M : V$. By Lemma 8.4, we have $\Gamma' \in \rho$. Hence, $V \in [\![M]\!]_\rho$.
   - Assume $U \in [\![M]\!]_\rho$ and $V \in [\![M]\!]_\rho$. Then, there exist $\Gamma, \Delta \in \rho$ such that $\Gamma \vdash M : U$ and $\Delta \vdash M : V$. Therefore, $\Gamma \cap \Delta \vdash M : U \cap V$ and by Lemma 8.5, $\Gamma \cap \Delta \in \rho$. Hence, $U \cap V \in [\![M]\!]_\rho$.

   Therefore, $[\![M]\!]_\rho \in \mathscr{D}$.
2. Assume $[\![M]\!]_\rho \neq \bot$. Then, there exists $A \in [\![M]\!]_\rho$. Therefore, there exists $\Gamma \in \rho$ such that $\Gamma \vdash M : A$. By Theorem 2, M is strongly normalising.
3.
   - Assume $F \in [\![x]\!]_\rho$. Then, there exists $\Gamma \in \rho$ such that $\Gamma \vdash x : F$. Therefore, $\Gamma = (x : F)$ and $F = \Gamma(x)$. We also have $\Gamma(x) \in \rho(x)$. Hence, $F \in \rho(x)$.
   - Assume $F \in \rho(x)$. Then, $(x : F) \in \rho$ and $x : F \vdash x : F$. Therefore, $F \in [\![x]\!]_\rho$.

   By Lemma 7.2, $[\![x]\!]_\rho = \rho(x)$.
4.
   - Assume $F \in [\![MN]\!]_\rho$. Then, there exists $\Gamma \in \rho$ such that $\Gamma \vdash MN : F$. Therefore, there exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M : A \to F$ and $\Gamma_2 \vdash N : A$. We have $\Gamma \subseteq \Gamma_1$ and $\Gamma \subseteq \Gamma_2$. By Lemma 8.4, $\Gamma_1 \in \rho$ and $\Gamma_2 \in \rho$. Hence, $(A \to F) \in [\![M]\!]_\rho$ and $A \in [\![N]\!]_\rho$. By Lemma 7.7, $F \in [\![M]\!]_\rho [\![N]\!]_\rho$.

$$\frac{x \in \mathsf{Dom}(\mathfrak{G}) \quad \mathfrak{G}(x) = \mathfrak{A}}{\mathfrak{G} \vdash_F x : \mathfrak{A}} \qquad \frac{\mathfrak{G} \vdash_F M : \mathfrak{A} \to \mathfrak{B} \quad \mathfrak{G} \vdash_F N : \mathfrak{A}}{\mathfrak{G} \vdash_F MN : \mathfrak{B}} \qquad \frac{\mathfrak{G}, x : \mathfrak{A} \vdash_F M : \mathfrak{B}}{\mathfrak{G} \vdash_F \lambda x.M : \mathfrak{A} \to \mathfrak{B}}$$

$$\frac{\mathfrak{G} \vdash_F M : \mathfrak{A} \quad \alpha \notin \mathrm{fv}(\mathfrak{G})}{\mathfrak{G} \vdash_F M : \forall \alpha.\mathfrak{A}} \qquad \frac{\mathfrak{G} \vdash_F M : \forall \alpha.\mathfrak{A}}{\Gamma \vdash_F M : \mathfrak{A}\{\alpha := \mathfrak{B}\}}$$

Figure 2.4: Typing rules of System F

- Assume $F \in [\![M]\!]_\rho[\![N]\!]_\rho$. By Lemma 7.7, there exists A such that $(A \to F) \in [\![M]\!]_\rho$ and $A \in [\![N]\!]_\rho$. Then, there exist $\Gamma_1, \Gamma_2 \in \rho$ such that $\Gamma_1 \vdash M : A \to F$ and $\Gamma_2 \vdash N : A$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash MN : F$. By Lemma 8.5, $\Gamma_1 \cap \Gamma_2 \in \rho$. Hence, $F \in [\![MN]\!]_\rho$.

By Lemma 7.2, $[\![MN]\!]_\rho = [\![M]\!]_\rho[\![N]\!]_\rho$.

5. Assume $F \in [\![\lambda x.M]\!]_\rho v$. By Lemma 8.5, there exists A such that $(A \to F) \in [\![\lambda x.M]\!]_\rho$ and $A \in v$. Then, there exists $\Gamma \in \rho$ such that $\Gamma \vdash \lambda x.M : A \to F$. Hence, there exist U such that $A \subseteq U$ and $\Gamma, x : U \vdash M : F$. Therefore, $U \in v$. By Lemma 8.1, $(\Gamma, x : U) \in (\rho, x \leftarrow v)$. Hence, $F \in [\![M]\!]_{(\rho, x \leftarrow v)}$.

   By Lemma 7.1, $[\![\lambda x.M]\!]_\rho v \subseteq [\![M]\!]_{(\rho, x \leftarrow v)}$.

6. By 5, we have $[\![\lambda x.M]\!]_\rho v \subseteq [\![M]\!]_{(\rho, x \leftarrow v)}$.

   Assume $F \in [\![M]\!]_{(\rho, x \leftarrow v)}$. Then, there exists $\Gamma \in (\rho, x \leftarrow v)$ such that $\Gamma \vdash M : F$. By Lemma 7.2, there exist $\Gamma_1 \in \rho$ and $U \in v$ such that $\Gamma = (\Gamma_1, x : U)$. Therefore, $\Gamma_1, x : U \vdash M : F$. By hypothesis, we are in one of the following cases:

   - We have $x \in \mathrm{fv}(M)$: Therefore, U is of the form A.
   - We have $v \neq \bot$ and U is of the form A.
   - We have $v \neq \bot$ and $U = \omega$: Then, there exists $A \in v$ and we have $A \subseteq \omega$.

   In all cases, there exists $A \in v$ such that $A \subseteq U$. Hence, $\Gamma_1 \vdash \lambda x.M : A \to F$. Therefore, $(A \to F) \in [\![\lambda x.M]\!]_\rho$. By Lemma 7.7, $F \in [\![\lambda x.M]\!]_\rho v$.

   By Lemma 7.1, $[\![M]\!]_{(\rho, x \leftarrow v)} \subseteq [\![\lambda x.M]\!]_\rho v$.

   Then, we can conclude.

7. Assume $F \in [\![M]\!]_\rho$. Then, there exists $\Gamma \in \rho$ such that $\Gamma \vdash M : F$. By Theorem 1, there exists Γ' such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash M' : F$. By Lemma 8.4, $\Gamma' \in \rho$. Therefore, $F \in [\![M']\!]_\rho$.

   By Lemma 7.1, $[\![M]\!]_\rho \subseteq [\![M']\!]_\rho$.

□

2.5.2 Example: System F

Definition 15 (System F)**.**

Types of System F are defined with the following grammar:

$$\mathfrak{A}, \mathfrak{B}, \mathfrak{C} ::= \alpha \mid \mathfrak{A} \to \mathfrak{B} \mid \forall \alpha.\mathfrak{A}$$

where α ranges over an infinite set of type variables*. Free variables and substitution is defined the usual way.*

A context $\mathfrak{G}$ in System is a partial map from type variables to types of System F.

The typing judgement $\mathfrak{G} \vdash_F M : \mathfrak{A}$ is defined with the rules of Figure 2.4.

Definition 16 (Model of System F)**.**

We define TP($\mathscr{D}$) the set of subsets X of $\mathscr{D}$ such that:

- $\top \in X$.
- $\bot \notin X$.

Assume $X, Y \in TP(\mathscr{D})$.
Let $X \to Y := \{u \in \mathscr{D} \mid \forall v \in X, uv \in Y\}$.

Lemma 10 (Examples of elements of TP($\mathscr{D}$))**.**

1. *We have* $\{\top\} \in TP(\mathscr{D})$.
2. *Assume* $(X_i)_{i\in I}$ *a non-empty family of* $TP(\mathscr{D})$. *Then,* $\bigcap_{i\in I} X_i \in TP(\mathscr{D})$.
3. *Assume* $X, Y \in TP(\mathscr{D})$. *Then* $X \to Y \in TP(\mathscr{D})$.

Proof.

1. Straightforward.
2. Straightforward.
3. - Assume $v \in X$. Then, by Definition 16, we have $v \neq \bot$. By Lemma 7.9, $\top v = \top$. By Definition 16, we have $\top \in Y$. Therefore, $\top \in X \to Y$.
   - Assume $\bot \in X \to Y$. We have $\top \in X$. Therefore, we have $\bot\top \in Y$. By Lemma 7.8, $\bot\top = \bot$. Therefore, $\bot \in Y$. Contradiction.
     Hence, $\bot \notin X \to Y$.

   Therefore, $X \to Y \in \mathrm{TP}(\mathscr{D})$.

□

Definition 17 (Interpretation of System F in the TP($\mathscr{D}$))**.**
Assume σ *a total map from type variables* α *to* $TP(\mathscr{D})$ *(*type environment*).*
We define $[\mathfrak{A}]_\sigma \in TP(\mathscr{D})$ *by induction on* $\mathfrak{A}$ *as follows:*

$$\begin{array}{lcl} [\alpha]_\sigma & := & \sigma(\alpha) \\ [\mathfrak{A} \to \mathfrak{B}]_\sigma & := & [\mathfrak{A}]_\sigma \to [\mathfrak{B}]_\sigma \\ [\forall \alpha.\mathfrak{A}]_\sigma & := & \bigcap_{X \in TP(\mathscr{D})} [\mathfrak{A}]_{(\sigma, \alpha \leftarrow X)} \end{array}$$

We define $[\mathfrak{G}]_\sigma$ *as the set of environment* ρ *such that:*

$$\forall x \in \mathsf{Dom}(\mathfrak{G}), \rho(x) \in [\mathfrak{G}(x)]_\sigma$$

Theorem 3 (Soundness of the model)**.**
Assume $\mathfrak{G} \vdash_F M : \mathfrak{A}$.
Then, for all σ *type environment we have:*

$$\forall \rho \in [\mathfrak{G}]_\sigma, [\![M]\!]_\rho \in [\mathfrak{A}]_\sigma$$

Proof. By induction on $\mathfrak{G} \vdash_F M : \mathfrak{A}$.

- For $\dfrac{x \in \mathsf{Dom}(\mathfrak{G}) \quad \mathfrak{G}(x) = \mathfrak{A}}{\mathfrak{G} \vdash_F x : \mathfrak{A}}$ with $M = x$: Therefore, $[\![x]\!]_\rho = \rho(x)$ (by Lemma 9.3).
  And we have $\rho(x) \in [\mathfrak{G}(x)]_\sigma = [\mathfrak{A}]_\sigma$.
- For $\dfrac{\mathfrak{G} \vdash_F M_1 : \mathfrak{B} \to \mathfrak{A} \quad \mathfrak{G} \vdash_F M_2 : \mathfrak{B}}{\mathfrak{G} \vdash_F M_1 M_2 : \mathfrak{A}}$ with $M = M_1 M_2$: By Lemma 9.4, $[\![M_1 M_2]\!]_\rho = [\![M_1]\!]_\rho [\![M_2]\!]_\rho$. By induction hypothesis, $[\![M_1]\!]_\rho \in [\mathfrak{B} \to \mathfrak{A}]_\sigma = [\mathfrak{B}]_\sigma \to [\mathfrak{A}]_\sigma$ and $[\![M_2]\!]_\rho \in [\mathfrak{B}]_\sigma$. Therefore, $[\![M_1]\!]_\sigma [\![M_2]\!]_\sigma \in [\mathfrak{A}]_\sigma$.
- For $\dfrac{\mathfrak{G}, x : \mathfrak{A}_1 \vdash_F M_1 : \mathfrak{A}_2}{\mathfrak{G} \vdash_F \lambda x.M_1 : \mathfrak{A}_1 \to \mathfrak{A}_2}$ with $M = \lambda x.M_1$ and $\mathfrak{A} = \mathfrak{A}_1 \to \mathfrak{A}_2$:
  Assume $v \in [\mathfrak{A}_1]_\sigma$. Then, $v \neq \bot$. By Lemma 9.6, $[\![\lambda x.M_1]\!]_\rho v = [\![M_1]\!]_{(\rho, x \leftarrow v)}$.
  We also have $(\rho, x \leftarrow v) \in [\mathfrak{G}, x : \mathfrak{A}_1]_\sigma$. By induction hypothesis, $[\![M_1]\!]_{(\rho, x \leftarrow v)} \in [\mathfrak{A}_2]_\sigma$. Therefore, $[\![\lambda x.M_1]\!]_\rho v \in [\mathfrak{A}_2]_\sigma$.
  Hence, $[\![\lambda x.M_1]\!]_\rho \in [\mathfrak{A}_1]_\sigma \to [\mathfrak{A}_2]_\sigma$. Therefore, $[\![M]\!]_\rho \in [\mathfrak{A}]_\sigma$.

- For $\dfrac{\mathfrak{G} \vdash_F M : \mathfrak{A}_1 \quad \alpha \notin \mathrm{fv}(\mathfrak{G})}{\mathfrak{G} \vdash_F M : \forall\alpha.\mathfrak{A}_1}$ with $\mathfrak{A} = \forall\alpha.\mathfrak{A}_1$:

  Assume $X \in \mathrm{TP}(\mathscr{D})$. By the fact that $\alpha \notin \mathrm{fv}(\mathfrak{G})$, we have $[\mathfrak{G}]_{(\sigma,\alpha\leftarrow X)} = [\mathfrak{G}]_\sigma$. Therefore, $\rho \in [\mathfrak{G}]_{(\sigma,\alpha\leftarrow X)}$. By induction hypothesis, $[\![M]\!]_\rho \in [\mathfrak{A}_1]_{(\sigma,\alpha\leftarrow X)}$.

  Hence, $[\![M]\!]_\rho \in \bigcap_{X\in\mathrm{TP}(\mathscr{D})}[\mathfrak{A}_1]_{(\sigma,\alpha\leftarrow X)} = [\forall\alpha.\mathfrak{A}_1]_\sigma = [\mathfrak{A}]_\sigma$.

- For $\dfrac{\mathfrak{G} \vdash_F M : \forall\alpha.\mathfrak{A}_1}{\mathfrak{G} \vdash_F M : \mathfrak{A}_1\{\alpha := \mathfrak{A}_2\}}$ with $\mathfrak{A} = \mathfrak{A}_1\{\alpha := \mathfrak{A}_2\}$:

  By induction hypothesis, $[\![M]\!]_\rho \in [\forall\alpha.\mathfrak{A}_1]_\sigma = \bigcap_{X\in\mathrm{TP}(\mathscr{D})}[\mathfrak{A}_1]_{(\sigma,\alpha\leftarrow X)}$. In particular, $[\![M]\!]_\rho \in [\mathfrak{A}_1]_{(\sigma,\alpha\leftarrow[\mathfrak{A}_2]_\sigma)}$. We can prove by induction on $\mathfrak{A}_1$ that $[\mathfrak{A}_1\{\alpha := \mathfrak{A}_2\}]_\sigma = [\mathfrak{A}_1]_{(\sigma,\alpha\leftarrow[\mathfrak{A}_2]_\sigma)}$. Therefore, $[\![M]\!]_\rho \in [\mathfrak{A}]_\sigma$.

□

Theorem 4 (Strong normalisation of System F)**.**
If $\mathfrak{G} \vdash_F M : \mathfrak{A}$, then M is strongly normalising.

Proof.
Let σ defined by: $\forall\alpha, \sigma(\alpha) := \{\top\}$.
Let ρ defined by: $\forall x, \rho(x) := \top$. Therefore, $\rho \in [\mathfrak{G}]_\sigma$.
By Theorem 3, we have $[\![M]\!]_\rho \in [\mathfrak{A}]_\sigma$. Therefore, $[\![M]\!]_\rho \neq \bot$.
By Lemma 9.2, M is strongly normalising.

□

2.6 Completeness

The goal of this section is to prove that if a term is strongly normalising, then it is typable. We use the same method usually done to prove this result with idempotent intersection type:

- We exhibit a β-reduction sequence from M to its normal form M'.
- We type M'.
- To get a typing of M we backtrack from M': For each β-step $M_1 \to_\beta M_2$, we get a typing of M_1 from a typing of M_2.

What we do in each β-step is called Subject Expansion (Theorem 5). Unfortunately, Subject Expansion does not work for every reduction $M \to_\beta M'$. In particular, it does not hold when $M \notin \mathbf{SN}$ and $M' \in \mathbf{SN}$ (by Theorem 2). Here are a few examples bellow:

- Case where the argument of the β-redex is not strongly normalising and is erased:
$$(\lambda x.y)((\lambda z.zz)(\lambda z.zz)) \to_\beta y$$
- More subtle case:
$$(\lambda w.(\lambda x.y)(ww))(\lambda z.zz) \to_\beta (\lambda w.y)(\lambda z.zz)$$
  Here, the erased term ww is strongly normalising but w, one of its free variable is caught by the abstraction λw.

Therefore, we need a restricted reduction relation such that:

- It can reach the normal form.
- It satisfies Subject Expansion. Therefore, it preserves strong normalisation both ways.

The restricted reduction relation that we give here is not the smallest one that satisfies those properties. On the contrary, we try to define the biggest one that satisfies them. This gives us a better understanding about operational semantics and intersection types and it gives more weight to some theorems such as Theorem 2.6 (which states that this strategy is *perpetual*).

First, we have to understand why we cannot directly adapt the proof of Subject Reduction to prove Subject Expansion:

- For $M_1 \to_\beta M_2$: When an argument N of a β-redex is erased, to type M_1, we need to type N and we have no information about how to type N from the typing of M_2. (first example)
- When a term is erased, the type of a context changes after Subject Expansion (it gets bigger). When dealing with the (*App*) rule, we are in a similar situation that we were in Subject Reduction. To avoid this problem, in Subject Reduction, we can have a subsumption but here the subsumption is on the wrong side.

  Therefore, when doing an erasure, we have to make sure that the variables of the erased term are not caught by a lambda like in the second example.

Hence, it seems logic to keep information about the set of variables E of an erased terms when defining the strategy.

We actually introduce, two restricted reduction relations that are mutally defined: $\rightsquigarrow_E$ and $\Rightarrow_E$. The second one is more general than the first. This definition uses an "accumulator property" denoted $\mathrm{acc}_x(M)$ ($\mathrm{acc}_x(M)$ if and only if M is of the form $xM_1 \dots M_n$; we also say that x is the head variable of M). The rules are given in Figure 2.5 and are purely syntactic. However, their motivation and meaning is that:

- If M cannot be reduced by $\to_\beta$, then M is typable.
- If $\mathrm{acc}_x(M)$ and M is typable, then we can give M any type by just changing the type of x.
- If $M \rightsquigarrow_E M'$ and M' is typable with a type A, then M has type A, just by changing the types of each x in E.
- If $M \Rightarrow_E M'$ and M' is typable with a type A, then M is typable with a type B (which can be different from A), just by changing the types of each x in E.

In more simple and common perpetual strategies (such as the one defined in Chapter 4) we do not need E.

More formally:

Definition 18 (Accumulators)**.**

Assume M is a term and x a variable. Then, $acc_x(M)$ is defined with the following rules:

$$\frac{}{acc_x(x)} \qquad \frac{acc_x(M)}{acc_x(MN)}$$

We write $acc(M)$ if there exists x such that $acc_x(M)$.

Definition 19 (Special reductions)**.**

Assume M and M' are terms and E is a finite set of variables. The reductions $M \rightsquigarrow_E M'$ and $M \Rightarrow_E M'$ are defined with the rules of Figure 2.5.

Lemma 11 (Execution of a λ-term)**.**

1. *If M cannot be reduced by $\to_\beta$, then we are in one of the following cases:*
   - *M is of the form $\lambda x.M_1$.*
   - *There exists x such that $acc_x(M)$.*

$$\frac{x \in \mathrm{fv}(M)}{(\lambda x.M)N \rightsquigarrow_\emptyset M\{x := N\}} \qquad \frac{x \notin \mathrm{fv}(M) \quad N \Rightarrow_E N'}{(\lambda x.M)N \rightsquigarrow_E (\lambda x.M)N'}$$

$$\frac{x \notin \mathrm{fv}(N) \quad N \text{ cannot be reduced by } \rightarrow_\beta}{(\lambda x.M)N \rightsquigarrow_{\mathrm{fv}(N)} M}$$

$$\frac{M \rightsquigarrow_E M'}{MN \rightsquigarrow_E M'N} \qquad \frac{N \rightsquigarrow_E N'}{MN \rightsquigarrow_E MN'} \qquad \frac{M \rightsquigarrow_E M' \quad x \notin E}{\lambda x.M \rightsquigarrow_E \lambda x.M'}$$

$$\frac{M \rightsquigarrow_E M'}{M \Rightarrow_E M'} \qquad \frac{M \Rightarrow_E M'}{\lambda x.M \Rightarrow_{E-\{x\}} \lambda x.M'} \qquad \frac{\mathrm{acc}_x(M) \quad N \Rightarrow_E N'}{MN \rightsquigarrow_{E\cup\{x\}} MN'}$$

Figure 2.5: Rules of $\rightsquigarrow_E$ and $\Rightarrow_E$

2. *If M can be reduced by $\rightarrow_\beta$, then there exists E and M' such that $M \Rightarrow_E M'$. Moreover, if M is not of the form $\lambda x.M$, then there exist E and M' such that $M \rightsquigarrow_E M'$.*
3. *If $M \rightsquigarrow_E M'$, then $M \Rightarrow_E M'$.*
4. *If $M \Rightarrow_E M'$ then $M \rightarrow_\beta M'$.*

Proof.

1. By induction on M - see e.g. [Böh68].
2. By induction on M: We are in one of the following cases:
   - M is of the form x: Then, M cannot be reduced by $\rightarrow_\beta$. Contradiction.
   - M is of the form $\lambda x.M_1$: Then, M_1 can be reduced by $\rightarrow_\beta$. By induction hypothesis, there exist E and M_1' such that $M_1 \Rightarrow_E M_1'$. Therefore, $M \Rightarrow_E \lambda x.M_1'$.
   - M is of the form $(\lambda x.M_1)M_2$ and $x \in \mathrm{fv}(M_1)$: Therefore, $M \rightsquigarrow_\emptyset M_1\{x := M_2\}$.
   - M is of the form $(\lambda x.M_1)M_2$, $x \notin \mathrm{fv}(M_1)$ and M_2 can be reduced by $\rightarrow_\beta$: By induction hypothesis, there exist E and M_2' such that $M_2 \Rightarrow_E M_2'$. Therefore, $M \rightsquigarrow_E (\lambda x.M_1)M_2'$.
   - M is of the form $(\lambda x.M_1)M_2$, $x \notin \mathrm{fv}(M_1)$ and M_2 cannot be reduced by $\rightarrow_\beta$: Therefore, $M \rightsquigarrow_{\mathrm{fv}(M_2)} M_1$.
   - M is of the form M_1M_2, M_1 is not of the form $\lambda x.M_3$ and M_1 can be reduced by $\rightarrow_\beta$: By induction hypothesis, there exist E and M_1' such that $M_1 \rightsquigarrow_E M_1'$. Therefore, $M \rightsquigarrow_E M_1'M_2$.
   - M is of the form M_1M_2, M_1 is not of the form $\lambda x.M_3$ and M_1 cannot be reduced by $\rightarrow_\beta$: By 1, there exists x such that $\mathrm{acc}_x(M)$.

     If M_2 cannot be reduced by $\rightarrow_\beta$, then M cannot be reduced by $\rightarrow_\beta$. Contradiction. Therefore, M_2 can be reduced by $\rightarrow_\beta$.

     By induction hypothesis, there exist E and M_2' such that $M_2 \Rightarrow_E M_2'$. Therefore, $M \rightsquigarrow_{E\cup\{x\}} M_1M_2'$.

   In all cases, if $M \rightsquigarrow_E M'$, then $M \Rightarrow_E M'$.
3. Trivial.
4. We prove the following by induction on $M \rightsquigarrow_E M'$ and by induction on $M \Rightarrow_E M'$ (mutual recursion):

- If $M \rightsquigarrow_E M'$, then $M \to_\beta M'$.
- If $M \Rightarrow_E M'$, then $M \to_\beta M'$.

□

As usual for the proof of Subject Expansion, we first prove an anti-substitution lemma:

Lemma 12 (Anti-substitution lemma)**.**
If $\Gamma \vdash M\{x := N\} : A$*, then there exist* Γ_1, Γ_2 *and* U *such that* $\Gamma \approx \Gamma_1 \cap \Gamma_2$, $\Gamma_1, x : U \vdash M : A$ *and* $\Gamma_2 \vdash N : U$*.*

Proof. First by induction on M, then by induction on A. We adapt the proof of Lemma 6.

We are in one of the following cases:

- A is of the form $A_1 \cap A_2$: Then, there exist Γ_1 and Γ_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M\{x := N\} : A_1$ and $\Gamma_2 \vdash M\{x := N\} : A_2$. By induction hypothesis on (M_1, A_1) and on (M_2, A_2), there exist Γ'_1, Δ_1, Γ'_2, Δ_2, U_1 and U_2 such that $\Gamma_1 \approx \Gamma'_1 \cap \Delta_1$, $\Gamma_2 \approx \Gamma'_2 \cap \Delta_2$, $\Gamma'_1, x : U_1 \vdash M : A_1$, $\Gamma'_2, x : U_2 \vdash M : A_2$, $\Delta_1 \vdash N : U_1$, and $\Delta_2 \vdash N : U_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2, x : U_1 \cap U_2 \vdash M : A_1 \cap A_2$, $\Delta_1 \cap \Delta_2 \vdash N : U_1 \cap U_2$ and $(\Gamma'_1 \cap \Gamma'_2) \cap (\Delta_1 \cap \Delta_2) \approx (\Gamma'_1 \cap \Delta_1) \cap (\Gamma'_2 \cap \Delta_2) \approx \Gamma_1 \cap \Gamma_2 = \Gamma$.
- M is of the form x and A is of the form F: Then, $M\{x := N\} = N$. Hence, $x : F \vdash x : F$, $\Gamma \vdash N : F$ and $() \cap \Gamma = \Gamma$.
- M is of the form y and A is of the form F with $x \neq y$: Then, $M\{x := N\} = y$. Hence, $\Gamma = (y : F) = (y : F, x : \omega)$. Therefore, $y : F, x : \omega \vdash M : F$, $\vdash N : \omega$ and $(y : F) \cap () = (y : F) = \Gamma$.
- M is of the form $M_1 M_2$ and A is of the form F: Then, $M\{x := N\} = M_1\{x := N\}M_2\{x := N\}$. Therefore, there exist Γ_1, Γ_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M_1\{x := N\} : B \to F$ and $\Gamma_2 \vdash M_2\{x := N\} : B$. By induction hypothesis there exist Γ'_1, Δ_1, Γ'_2, Δ_2, U_1 and U_2 such that $\Gamma_1 \approx \Gamma'_1 \cap \Delta_1$, $\Gamma_2 \approx \Gamma'_2 \cap \Delta_2$, $\Gamma'_1, x : U_1 \vdash M_1 : B \to F$, $\Gamma'_2, x : U_2 \vdash M_2 : B$, $\Delta_1 \vdash N : U_1$, and $\Delta_2 \vdash N : U_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2, x : U_1 \cap U_2 \vdash M_1 M_2 : F$, $\Delta_1 \cap \Delta_2 \vdash N : U_1 \cap U_2$ and $(\Gamma'_1 \cap \Gamma'_2) \cap (\Delta_1 \cap \Delta_2) \approx (\Gamma'_1 \cap \Delta_1) \cap (\Gamma'_2 \cap \Delta_2) \approx \Gamma_1 \cap \Gamma_2 = \Gamma$.
- M is of the form $\lambda y.M_1$ and A is of the form F with $y \neq x$ and $y \notin \mathrm{fv}(N)$: Then, $M\{x := N\} = \lambda y.M_1\{x := N\}$. Therefore, there exist B, V and G such that $B \subseteq V$, $F = B \to G$ and $\Gamma, y : V \vdash M_1\{x := N\} : G$. By induction hypothesis, there exist Γ'_1, Δ and U such that $(\Gamma, y : V) \approx \Gamma'_1 \cap \Delta$, $\Gamma'_1, x : U \vdash M_1 : G$ and $\Delta \vdash N : U$. There exist Γ' and V' such that $\Gamma'_1 = (\Gamma', y : V')$. By Lemma 5.4 and by the fact that $y \notin \mathrm{fv}(M)$, we have $y \notin \mathsf{Dom}(\Delta)$. Hence, $\Gamma'_1 \cap \Delta = (\Gamma' \cap \Delta, y : V')$. Therefore, $\Gamma \approx \Gamma' \cap \Delta$ and $V \approx V'$. Hence, $B \subseteq V'$, $(\Gamma'_1, x : U) = (\Gamma', y : V', x : U) = (\Gamma', x : U, y : V')$ (because $x \neq y$). Therefore, $\Gamma', x : U \vdash \lambda y.M_1 : B \to G$.

□

Notice that, in Lemma 12, if $x \notin \mathrm{fv}(M)$, then $U = \omega$ and we do not have any typing information on N.

In $\Rightarrow_E$ and $\rightsquigarrow_E$, if a term is erased, then it is a normal form. Therefore, to prove Subject Expansion, we need to be able to type normal terms. This is also used in Theorem 6. To prove this, we need to know what is the type of an accumulator (when the context is an input).

Lemma 13 (Type of an accumulator)**.**
Assume $\Gamma \vdash M : F$ *and* $\mathrm{acc}_x(M)$*. Then for all* G*, there exists* Γ' *such that:*

- *For all* $y \neq x$, $\Gamma(x) = \Gamma'(x)$.
- *We have* $\Gamma' \vdash M : G$.

Proof. By induction on $\mathrm{acc}_x(M)$:

- For $\dfrac{}{\mathrm{acc}_x(x)}$ with $M = x$: Therefore, $\Gamma = (x : F)$. Hence, $x : G \vdash M : G$ and for all $y \neq x$, $(x : F)(y) = (x : G)(y) = \omega$.
- For $\dfrac{\mathrm{acc}_x(M_1)}{\mathrm{acc}_x(M_1 M_2)}$ with $M = M_1 M_2$: Then, there exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M_1 : A \to F$ and $\Gamma_2 \vdash M_2 : A$. By induction hypothesis, there exists Γ_1' such that $\Gamma_1' \vdash M_1 : A \to G$ and for all $y \neq x$, we have $\Gamma_1(y) = \Gamma_1'(y)$. Therefore, $\Gamma_1' \cap \Gamma_2 \vdash M_1 M_2 : G$.

  Assume $y \neq x$, then $(\Gamma_1' \cap \Gamma_2)(y) = \Gamma_1'(y) \cap \Gamma_2(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma(y)$.

□

Lemma 14 (Typing of a normal term)**.**
If M *cannot be reduced by* $\to_\beta$*, then there exist* Γ *and* F *such that* $\Gamma \vdash M : F$.

Proof. By induction on M. By Lemma 11.1, we are in one of the following cases:

- M is of the form x: Therefore, $x : \tau \vdash M : \tau$.
- M is of the form $M_1 M_2$ with $\mathrm{acc}_x(M)$: By induction hypothesis, there exist Γ_1, Γ_2, F_1 and F_2 such that $\Gamma_1 \vdash M_1 : F_1$ and $\Gamma_2 \vdash M_2 : F_2$. By Lemma 13, there exist Γ_1' such that $\Gamma_1' \vdash M_1 : F_2 \to \tau$. Therefore, $\Gamma_1' \cap \Gamma_2 \vdash M_1 M_2 : \tau$.
- M is of the form $\lambda x.M_1$: By induction hypothesis, there exist Γ_1 and F_1 such that $\Gamma_1 \vdash M_1 : F_1$. Then, there exist Γ and U such that $\Gamma_1 = (\Gamma, x : U)$. Let $A := U$ if U is of the form B and let $A := \tau$ if $U = \omega$. In both cases we have $A \subseteq U$. Therefore, $\Gamma \vdash \lambda x.M_1 : A \to F_1$.

□

Theorem 5 (Subject Expansion)**.**
Assume $\Gamma' \vdash M' : A$.

1. *If* $M \rightsquigarrow_E M'$*, then there exists* Γ *such that:*
   - *For all* $x \notin E$, $\Gamma(x) \approx \Gamma'(x)$.
   - *We have* $\Gamma \vdash M : A$.
2. *If* $M \Rightarrow_E M'$*, then there exist* Γ *and* B *such that:*
   - *For all* $x \notin E$, $\Gamma(x) \approx \Gamma'(x)$.
   - *We have* $\Gamma \vdash M : B$.
   - *If* A *is a* F*-type, then so is* B.

Proof. First by induction on $M \rightsquigarrow_E M'$ and $M \Rightarrow_E M'$ (mutual recursion), then by induction on A.

- If A is of the form $A_1 \cap A_2$ and $M \rightsquigarrow_E M'$: Then, there exist Γ_1' and Γ_2' such that $\Gamma' = \Gamma_1' \cap \Gamma_2'$, $\Gamma_1' \vdash M' : A_1$ and $\Gamma_2' \vdash M' : A_2$. By induction hypothesis on $(M \rightsquigarrow_E M', A_1)$ and $(M \rightsquigarrow_E M', A_2)$, there exist Γ_1 and Γ_2 such that:
  - For all $y \notin E$, $\Gamma_1(y) \approx \Gamma_1'(y)$ and $\Gamma_2(y) \approx \Gamma_2'(y)$.
  - $\Gamma_1 \vdash M : A_1$ and $\Gamma_2 \vdash M : A_2$.

  Therefore, $\Gamma_1 \cap \Gamma_2 \vdash M : A_1 \cap A_2$.

  Assume $y \notin E$. Then, $(\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) \approx \Gamma_1'(y) \cap \Gamma_2'(y) = (\Gamma_1' \cap \Gamma_2')(y) = \Gamma'(y)$.

- If A is of the form $A_1 \cap A_2$ and $M \Rightarrow_E M'$: Then, there exist Γ'_1 and Γ'_2 such that $\Gamma' = \Gamma'_1 \cap \Gamma'_2$, $\Gamma'_1 \vdash M' : A_1$ and $\Gamma'_2 \vdash M' : A_2$. By induction hypothesis on $(M \Rightarrow_E M', A_1)$ and $(M \Rightarrow_E M', A_2)$, there exist Γ_1, Γ_2, B_1 and B_2 such that:
  - For all $y \notin E$, $\Gamma_1(y) \approx \Gamma'_1(y)$ and $\Gamma_2(y) \approx \Gamma'_2(y)$.
  - $\Gamma_1 \vdash M : B_1$ and $\Gamma_2 \vdash M : B_2$.

  Therefore, $\Gamma_1 \cap \Gamma_2 \vdash M : B_1 \cap B_2$.

  Assume $y \notin E$. Then, $(\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) \approx \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y) = \Gamma'(y)$.
- For $\dfrac{x \in \mathrm{fv}(M_1)}{(\lambda x.M_1)M_2 \rightsquigarrow_\emptyset M_1\{x := M_2\}}$ with $M = (\lambda x.M_1)M_2$, $M' = M_1\{x := M_2\}$ and A is of the form F: By Lemma 12, there exist Γ_1, Γ_2 and U such that $\Gamma' \approx \Gamma_1 \cap \Gamma_2$, $\Gamma_1, x : U \vdash M_1 : F$ and $\Gamma_2 \vdash M_2 : U$. By Lemma 5.4, $x \in \mathsf{Dom}(\Gamma_1, x : U)$ and U is of the form B. Therefore, $\Gamma_1 \vdash \lambda x.M_1 : B \to F$ and $\Gamma_2 \vdash M_2 : B$. Hence, $\Gamma_1 \cap \Gamma_2 \vdash (\lambda x.M_1)M_2 : F$.

  Assume $y \notin \emptyset$. Then, $(\Gamma_1 \cap \Gamma_2)(y) \approx \Gamma'(y)$.
- For $\dfrac{x \notin \mathrm{fv}(M_1) \quad M_2 \Rightarrow_E M'_2}{(\lambda x.M_1)M_2 \rightsquigarrow_E (\lambda x.M_1)M'_2}$ with $M = (\lambda x.M_1)M_2$, $M' = (\lambda x.M_1)M'_2$ and A is of the form F: Then, there exist Γ'_1, Γ'_2 and B such that $\Gamma' = \Gamma'_1 \cap \Gamma'_2$, $\Gamma'_1 \vdash \lambda x.M_1 : B \to F$ and $\Gamma'_2 \vdash M_2 : B$.

  Therefore, there exists U such that $B \subseteq U$ and $\Gamma'_1, x : U \vdash M_1 : F$. By Lemma 5.4, $x \notin \mathsf{Dom}(\Gamma'_1, x : U)$ and $U = \omega$. Hence, $\Gamma'_1, x : \omega \vdash M_1 : F$.

  By induction hypothesis, there exist Γ_2 and B_1 such that:
  - $\Gamma_2 \vdash M_2 : B_1$.
  - For all $y \notin E$, $\Gamma_2(y) \approx \Gamma'_2(y)$.

  Therefore, $B_1 \subseteq \omega$, $\Gamma'_1 \vdash \lambda x.M_1 : B_1 \to F$ and $\Gamma'_1 \cap \Gamma_2 \vdash (\lambda x.M_1)M_2 : F$.

  Assume $y \notin E$. Then $(\Gamma'_1 \cap \Gamma_2)(y) = \Gamma'_1(y) \cap \Gamma_2(y) \approx \Gamma'_1(y) \cap \Gamma'_2(y) = \Gamma'(y)$.
- For $\dfrac{x \notin \mathrm{fv}(M_1) \quad M_2 \text{ cannot be reduced by } \to_\beta}{(\lambda x.M_1)M_2 \rightsquigarrow_{\mathrm{fv}(M_2)} M_1}$ with $M = (\lambda x.M_1)M_2$, $M' = M_1$ and A is of the form F:

  By Lemma 5.4 and by the fact that $x \notin \mathrm{fv}(M_1)$, we have $x \notin \mathsf{Dom}(\Gamma')$. Hence, $\Gamma' = (\Gamma', x : \omega)$ and $\Gamma', x : \omega \vdash M_1 : F$.

  By Lemma 14, there exist Δ and G such that $\Delta \vdash M_2 : G$. Therefore, $G \subseteq \omega$, $\Gamma' \vdash \lambda x.M_1 : G \to F$ and $\Gamma' \cap \Delta \vdash (\lambda x.M_1)M_2 : F$.

  Assume $y \notin \mathrm{fv}(M_2)$. By Lemma 5.4, $y \notin \mathsf{Dom}(\Delta)$ and $\Delta(y) = \omega$. Therefore, $(\Gamma' \cap \Delta)(y) = \Gamma'(y) \cap \Delta(y) = \Gamma'(y) \cap \omega = \Gamma'(y)$.
- For $\dfrac{M_1 \rightsquigarrow_E M'_1}{M_1M_2 \rightsquigarrow_E M'_1M_2}$ with $M = M_1M_2$, $M' = M'_1M_2$ and A is of the form F: Then, there exist Γ'_1, Γ'_2 and B such that $\Gamma' = \Gamma'_1 \cap \Gamma'_2$, $\Gamma'_1 \vdash M'_1 : B \to F$ and $\Gamma'_2 \vdash M_2 : B$. By induction hypothesis, there exist Γ_1 such that:
  - For all $x \notin E$, $\Gamma_1(x) \approx \Gamma'_1(x)$.
  - $\Gamma_1 \vdash M_1 : B \to F$.

  Therefore, $\Gamma_1 \cap \Gamma'_2 \vdash M_1M_2 : F$.

  Assume $x \notin E$. Then $(\Gamma_1 \cap \Gamma'_2)(x) = \Gamma_1(x) \cap \Gamma'_2(x) \approx \Gamma'_1(x) \cap \Gamma'_2(x) = (\Gamma'_1 \cap \Gamma'_2)(x) = \Gamma'(x)$.

- For $\dfrac{M_2 \rightsquigarrow_E M_2'}{M_1 M_2 \rightsquigarrow_E M_1 M_2'}$ with $M = M_1 M_2$, $M' = M_1 M_2'$ and A is of the form F: Then, there exist Γ_1', Γ_2' and B such that $\Gamma' = \Gamma_1' \cap \Gamma_2'$, $\Gamma_1' \vdash M_1 : B \to F$ and $\Gamma_2' \vdash M_2' : B$. By induction hypothesis, there exist Γ_2 such that:
  - For all $x \notin E$, $\Gamma_2(x) \approx \Gamma_2'(x)$.
  - $\Gamma_1 \vdash M_2 : B$.

  Therefore, $\Gamma_1' \cap \Gamma_2 \vdash M_1 M_2 : F$.

  Assume $x \notin E$. Then $(\Gamma_1' \cap \Gamma_2)(x) = \Gamma_1'(x) \cap \Gamma_2(x) \approx \Gamma_1'(x) \cap \Gamma_2'(x) = (\Gamma_1' \cap \Gamma_2')(x) = \Gamma'(x)$.
- For $\dfrac{M_1 \rightsquigarrow_E M_1' \quad x \notin E}{\lambda x.M_1 \rightsquigarrow_E \lambda x.M_1'}$ with $M = \lambda x.M_1$, $M' = \lambda x.M_1'$ and A is of the form F: Then, there exists B, U' and G such that $B \subseteq U'$, $F = B \to G$ and $\Gamma', x : U' \vdash M_1' : G$.

  By induction hypothesis, there exist Γ_1 such that:
  - For all $y \notin E$, $\Gamma_1(y) \approx (\Gamma', x : U')(y)$.
  - $\Gamma_1 \vdash M_1 : G$.

  There exist Γ and U such that $\Gamma_1 = (\Gamma, x : U)$. We have $x \notin E$. Therefore, $U = (\Gamma, x : U)(x) = \Gamma_1(x) \approx (\Gamma', x : U')(x) = U'$. Hence, $B \subseteq U$ and $\Gamma \vdash \lambda x.M_1 : B \to G$.

  Assume $y \notin E$.
  - If $y \neq x$: Then $\Gamma(y) = (\Gamma, x : U)(y) = \Gamma_1(y) \approx (\Gamma', x : U')(y) = \Gamma'(y)$.
  - If $y = x$: By Lemma 5.4 and by the fact that $x \notin \mathrm{fv}(\lambda x.M_1)$ and $x \notin \mathrm{fv}(\lambda x.M_1')$, we have $x \notin \mathsf{Dom}(\Gamma)$ and $x \notin \mathsf{Dom}(\Gamma')$. Therefore, $\Gamma(x) = \Gamma'(x) = \omega$.
- For $\dfrac{M \rightsquigarrow_E M'}{M \Rightarrow_E M'}$ with A is of the form F: Straightforward.
- For $\dfrac{M_1 \Rightarrow_E M_1'}{\lambda x.M_1 \Rightarrow_{E-\{x\}} \lambda x.M_1'}$ with $M = \lambda x.M_1$, $M' = \lambda x.M_1'$ and A is of the form F: Then, there exist B', U' and G' such that $B' \subseteq U'$, $F = B' \to G'$ and $\Gamma', x : U' \vdash M_1' : G'$. By induction hypothesis, there exist Γ_1 and G such that:
  - For all $y \notin E$, $\Gamma_1(y) \approx (\Gamma', x : U')(y)$.
  - $\Gamma_1 \vdash M_1 : G$.

  There exist Γ and U such that $\Gamma_1 = (\Gamma, x : U)$. Let $B := U$ if U is of the form C and let $B = \tau$ if $U = \omega$. In both cases, we have $B \subseteq U$ and $\Gamma \vdash \lambda x.M_1 : B \to G$.

  Assume $y \notin E - \{x\}$.
  - If $y \neq x$: Then $y \notin E$ and $\Gamma(y) = (\Gamma, x : U)(y) = \Gamma_1(y) \approx (\Gamma', x : U')(y) = \Gamma'(y)$.
  - If $y = x$: By Lemma 5.4 and by the fact that $x \notin \mathrm{fv}(\lambda x.M_1)$ and $x \notin \mathrm{fv}(\lambda x.M_1')$, we have $x \notin \mathsf{Dom}(\Gamma)$ and $x \notin \mathsf{Dom}(\Gamma')$. Therefore, $\Gamma(x) = \Gamma'(x) = \omega$.
- For $\dfrac{\mathrm{acc}_x(M_1) \quad M_2 \Rightarrow_E M_2'}{M_1 M_2 \rightsquigarrow_{E \cup \{x\}} M_1 M_2'}$ with $M = M_1 M_2$, $M' = M_1 M_2'$ and A is of the form F: Then, there exist Γ_1', Γ_2' and B such that $\Gamma' = \Gamma_1' \cap \Gamma_2'$, $\Gamma_1' \vdash M_1 : B \to F$ and $\Gamma_2' \vdash M_2' : B$.

  By induction hypothesis, there exist Γ_2 and B_1 such that:

– For all $y \notin E$, $\Gamma_2(y) \approx \Gamma'_2(y)$.
– $\Gamma_2 \vdash M_2 : B_1$.

By Lemma 13, there exist Γ_1 such that:
– For all $y \neq x$, $\Gamma_1(y) = \Gamma'_1(y)$.
– $\Gamma_1 \vdash M_1 : B_1 \to F$.

Therefore, $\Gamma_1 \cap \Gamma_2 \vdash M_1 M_2 : F$.

Assume $y \notin E \cup \{x\}$. Then, $y \notin E$ and $y \neq x$. Therefore, $(\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) \approx \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y) = \Gamma'(y)$.

□

Theorem 6 (Completeness)**.**
If M is strongly normalising, then there exist Γ and A such that $\Gamma \vdash M : A$.

Proof. By induction on the size of the longest β-reduction sequences of M:

- If M cannot be reduced by $\to_\beta$, then, by Lemma 14, there exist Γ and F such that $\Gamma \vdash M : F$.
- If M can be reduced by $\to_\beta$: By Lemma 11.2, there exist E and M' such that $M \Rightarrow_E M'$. By Lemma 11.4, $M \to_\beta M'$. By induction hypothesis, there exist Γ' and A' such that $\Gamma' \vdash M' : A'$. By Theorem 5, there exist Γ and A such that $\Gamma \vdash M : A$.

□

Theorem 7 ($\Rightarrow_E$ preserves strong normalisation)**.**
If $M \Rightarrow_E M'$ and M' is strongly normalising, then M is strongly normalising.

Proof. By Theorem 6, there exist Γ' and A' such that $\Gamma' \vdash M' : A'$. By Theorem 5, there exist Γ and A such that $\Gamma \vdash M : A$. By Theorem 1, M is strongly normalising.

□

Theorem , is a generalisation of the conservation theorem: If $M \to_\beta M'$ without erasure and M' is strongly normalising, then M is strongly normalising. And the proof mostly use a semantical argument.

2.7 Conclusion

In this chapter, we have defined a typing system of non-idempotent intersection types and we have proved that it characterizes strongly normalising terms. In particular, the proof of strong normalisation is a corollary of Subject Reduction. And we have also proved that this system can be used to build a denotational semantics based on I-filters which itself can be used to prove strong normalisation for other kinds of typing system such as System F

Theorem 2 (Soundness) gives us a bound on the size of longest β-reduction sequences. This is an inequality result. However, we would like a more precise result: an equality result. To have this kind of result, we have two different possible approaches:

- Not only erasure makes Completeness harder to prove, but it also makes Theorem 2 less precise. Therefore, instead of working in the pure λ-calculus, we can work in a calculus where we have a better control on the erasure of terms. The calculus we choose for this study is λS (λ-calculus with explicit substitutions). Of course we have to adapt the typing system of this chapter to this calculus. This is what we do in Chapter 3.

- We still work in the pure λ-calculus, but we choose to refine the types and the typing system such that the measure of a (certain) typing tree of every normal term is equal to zero. This is what we do in Chapter 4.

Chapter 3

Intersection types with explicit substitutions

3.1 Introduction

In the pure λ-calculus, we go from a β-redex $(\lambda x.M)N$ to the term $M\{x := N\}$ in one step. It is possible to work in a calculus where this reduction is refined: we go from the β-redex $(\lambda x.M)N$ to something called an explicit substitution $M[x := N]$ and then it can take several steps to go from the term $M[x := N]$ to the term $M\{x := N\}$ (this is called the propagation of the explicit substitution). With explicit substitutions we have a better control on duplication and erasure: only some propagation rules trigger duplications and erasures. In particular, on some calculi, the erasure can be postponed and done just before reaching the normal form. There are lots of variants for a λ-calculus with explicit substitutions (see for instance [Kes07, Ren11]). In this chapter, we will only study one and adapt the typing system of Chapter 2.

In Section 3.2, we define the syntax of the studied calculus and we give its operational semantics (reduction rules). In Section 3.3, we adapt the typing system of Chapter 2 to the calculus studied in this chapter. In Section 3.4, like in Section 2.4, we prove that if a term is typable then it is strongly normalising and the measure of the typing tree is a bound on the size of the longest reduction sequences (Theorem 9). In this chapter we measure the number of uses of the rule B that transforms an occurrence of $(\lambda x.M)N$ into the term $M[x := N]$. To have a more precise complexity result than just the one given in Theorem 9, we need to restrict ourselves to some particular kind of typing trees that we call *optimal*. In Section 3.5 we define what an optimal typing tree is and in Section 3.6 we prove that if a term is strongly normalising, then it is possible to typed it with an optimal typing tree. Finally, in Section 3.7, we prove a more precise result than Theorem 9 when the typing tree of a term is optimal.

3.2 Syntax

Definition 20 (Terms)**.** *We extends the syntax of λ-calculus as follows:*

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

The free variables $fv(M)$ of a term M are defined by the rules of figure 3.1.

Definition 21 (Reduction in λS)**.**

The reduction and equivalence rules of λS are presented in Fig. 3.2.

$$fv(x) = \{x\} \qquad fv(\lambda x.M) = fv(M) - \{x\} \qquad fv(MN) = fv(M) \cup fv(N)$$

$$fv(M[x := N]) = (fv(M) - \{x\}) \cup fv(N)$$

Figure 3.1: Free variables of a term

$B:$	$(\lambda x.M)N$	$\to$	$M[x := N]$	
$W:$	$y[x := N]$	$\to$	y	$x \neq y$
$S:$				
	$x[x := N]$	$\to$	N	(SR)
	$(M_1M_2)[x := N]$	$\to$	$(M_1[x := N])(M_2[x := N])$	$x \in fv(M_1), x \in fv(M_2)$
	$(M_1M_2)[x := N]$	$\to$	$M_1(M_2[x := N])$	$x \notin fv(M_1), x \in fv(M_2)$
	$(M_1M_2)[x := N]$	$\to$	$(M_1[x := N])M_2$	$x \notin fv(M_2)$
	$(\lambda y.M)[x := N]$	$\to$	$\lambda y.M[x := N]$	$x \neq y, y \notin fv(N)$
	$(M_1[y := M_2])[x := N]$	$\to$	$(M_1[x := N])[y := M_2[x := N]]$	$x \in fv(M_1), x \in fv(M_2), y \notin fv(N)$
	$(M_1[y := M_2])[x := N]$	$\to$	$M_1[y := M_2[x := N]]$	$x \notin fv(M_1), x \in fv(M_2)$
	$(M[x := N_1])[y := N_2]$	$\equiv$	$(M[y := N_2])[x := N_1]$	$x \neq y, x \notin fv(N_2), y \notin fv(N_1)$

Figure 3.2: Reduction and equivalence rules of λS

For a set of rules $E \subseteq \{B, S, W\}$ from Figure 3.2, $\to_E$ denotes the congruent closure of the rules in E modulo the $\equiv$ rule.

SN$_{\lambda S}$ *denotes the set of strongly normalising λS-terms for $\to_{B,S,W}$.*

We call this calculus λS because it is a variant of the calculi λ_s of [Kes07] and λ_{es} of [Ren11]. That of [Ren11] is more general than that of [Kes07] in the sense that it allows the reductions

(1) $(M_1M_2)[x := N] \to M_1[x := N]M_2$ when $x \notin fv(M_1), x \notin fv(M_2)$
(2) $(M_1M_2)[x := N] \to M_1M_2[x := N]$ when $x \notin fv(M_1), x \notin fv(M_2)$

Reduction (2) is problematic in our approach since, even though the Subject Reduction property would still hold, it would not hold with the quantitative information from which Strong Normalisation can be proved: In the typing tree, the type of M_1 is not an intersection (it is an F-type) but the type of M_2 can be one. So we cannot directly type $M_2[x := N]$. If $x \in fv(M_2)$ we can use Lemma 17, otherwise we have to duplicate the typing tree of N.

We therefore exclude (2) from the calculus, but keep (1) as one of our rules, since it is perfectly compatible with our approach. It is also needed to simulate (in several steps) the general *garbage collection* rule below

$$M[x := N] \to M \quad (x \notin fv(M))$$

which is present in both [Kes07] and [Ren11], and which we decide to restrict, for simplicity, to the case where M is a variable different from x.[1] All of our results would still hold with the general garbage collection rule.

Lemma 15. *$\to_{S,W}$ terminates.*

Proof. By a polynomial argument. More precisely, see appendix A. □

[1] [Kes07] needs the general version, if only for the lack of rule (1).

3.3 Typing judgments

In this chapter, we use the same intersection types and the same contexts used in Chapter 2. Also, we use the same definition for the equivalence and the inclusion (see Sections 2.3.1 and 2.3.2). For the typing judgment, we add an extra rule to type explicit substitutions.

Definition 22 (Typability)**.**
The judgment $\Gamma \vdash M : U$ *denotes the derivability of* $\Gamma \vdash M : U$ *with the rules of Fig. 3.3. We write* $\Gamma \vdash^n M : U$ *if there exists a derivation with* n *uses of the (App) rule.*

$$\frac{}{x : F \vdash x : F}\ (\text{Var}) \qquad \frac{\Gamma, x : U \vdash M : F \quad A \subseteq U}{\Gamma \vdash \lambda x.M : A \to F}\ (\text{Abs})$$

$$\frac{\Gamma \vdash M : A \to F \quad \Delta \vdash N : A}{\Gamma \cap \Delta \vdash MN : F}\ (\text{App})$$

$$\frac{\Gamma \vdash M : A \quad \Delta \vdash M : B}{\Gamma \cap \Delta \vdash M : A \cap B}\ (\text{Inter}) \qquad \frac{}{\vdash M : \omega}\ (\text{Omega})$$

$$\frac{\Gamma \vdash N : A \quad \Delta, x : U \vdash M : F \quad U = A \vee U = \omega}{\Gamma \cap \Delta \vdash M[x := N] : F}\ (\text{Subst})$$

Figure 3.3: Typing rules

We can notice that, unlike the (Abs) rule, we do not have any subsumption in the rule (Subst). But we still have two cases: when $x \in \text{fv}(M)$ (we have $U = A$) and when $x \notin \text{fv}(M)$ (we have $U = \omega$).

Like in Chapter 2, we have the following properties:

Lemma 16 (Basic properties)**.**

1. *If* $\Gamma \vdash^n M : U \cap V$*, then there exist* Γ_1*,* Γ_2*,* n_1*,* n_2 *such that* $n = n_1 + n_2$*,* $\Gamma = \Gamma_1 \cap \Gamma_2$*,* $\Gamma_1 \vdash^{n_1} M : U$ *and* $\Gamma_2 \vdash^{n_2} M : V$*.*
2. *If* $\Gamma \vdash_s M : A$*, then* $\mathsf{Dom}(\Gamma) = fv(M)$*.*
3. *If* $\Gamma \vdash^n M : U$ *and* $U \approx U'$*, then there exists* Γ' *such that* $\Gamma \approx \Gamma'$ *and* $\Gamma' \vdash^n M : U'$
4. *If* $\Gamma \vdash^n M : U$ *and* $U \subseteq V$*, then there exist* m *and* Δ *such that* $m \leq n$*,* $\Gamma \subseteq \Delta$ *and* $\Delta \vdash^m M : V$

Proof. Similar to the proof of Lemma 5. □

The following lemma is used to prove Subject Reduction in λS.

Lemma 17 (Typing of explicit substitution)**.**
Assume $\Gamma, x : A \vdash^n M : B$ *and* $\Delta \vdash^m N : A$*. Then, there exists* Γ' *such that* $\Gamma' \approx \Gamma \cap \Delta$ *and* $\Gamma' \vdash^{n+m} M[x := N] : B$*.*

Proof. By induction on B. See Appendix A. □

3.4 Soundness

Theorem 8 (Subject Reduction for λS)**.**
Assume $\Gamma \vdash^n M : A$. We have the following properties:

1. *If $M \to_B M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m < n$ and $\Gamma' \vdash^m M' : A$*
2. *If $M \to_S M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n M' : A$*
3. *If $M \to_W M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m \leq n$ and $\Gamma' \vdash^m M' : A$*
4. *If $M \equiv M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n M' : A$*

Proof. See Appendix A. □

Theorem 9 (Soundness for λS)**.**
If M is a λS-term and $\Gamma \vdash M : A$, then $M \in \boldsymbol{SN}_{\lambda S}$.

Proof. We have $\Gamma \vdash^n M : A$ for some n, and strong normalisation is provided by a lexicographic argument on the pair (n, m) where m the length of the longuest S, W reduction sequences (by Lemma 15, $\to_{W,S}$ terminated on its own). Indeed:

- $\to_B$ strictly decreases n
- $\to_S$ and $\to_W$ decrease n or do not change it
- $\to_{S,W}$ strictly decreases m.

□

3.5 Special property of typing trees: Optimality

In the next sub-section we will notice that the typing trees produced by the proof of completeness all satisfy a particular property. In this section we define this properties: *optimality*

This property involves the following notions:

Definition 23 (Subsumption and forgotten types)**.**

- *If π is a typing tree, we say that π* does not use subsumption *if every occurence of the abstraction rule is such that $A \subseteq U$ is either $A \approx U$ or $A \subseteq \omega$.*
- *We say that a type A is* forgotten *in an instance of rule (Abs) or rule (Subst) if in the side-condition of the rule we have $U = \omega$.*
- *If a typing tree π uses no subsumption, we collect the list of its forgotten types, written $\mathsf{forg}(\pi)$, by a standard prefix and depth-first search of the typing tree π.*

If $\Gamma \vdash^n M : A$ without subsumption, then we write $\Gamma \vdash^n_{ns} M : A$.

The optimal property also involves refining the grammar of types:

Definition 24 (Refined intersection types)**.**
A^+, A^-, A^{--} and U^{--} are defined by the following grammar:

$$\begin{array}{lll} A^+, B^+ & ::= & \tau \mid A^{--} \to B^+ \\ A^{--}, B^{--} & ::= & A^- \mid A^{--} \cap B^{--} \\ A^-, B^- & ::= & \tau \mid A^+ \to B^- \\ U^{--}, V^{--} & ::= & A^{--} \mid \omega \end{array}$$

We say that Γ is of the form Γ^{--} if for all x, $\Gamma(x)$ is of the form U^{--}.

We can finally define the optimal property:

Definition 25 (Optimal typing)**.**
A typing tree π concluding $\Gamma \vdash M : A$ is optimal if

- *There is no subsumption in π*
- *A is of the form A^+*
- *For every $(x : B) \in \Gamma$, B is of the form B^{--}*
- *For every forgotten type B in π, B is of the form B^+.*

We write $\Gamma \vdash_{opt} M : A^+$ if there exists such π.

In this definition, A^+ is an output type, A^- is a basic input type (i.e. for a variable to be used once), and A^{--} is the type of a variable that can be used several times. The intuition behind this asymmetric grammar can be found in linear logic:

Remark 1. *Intersection in a typing tree means duplication of resource. So intersections can be compared to exponentials in linear logic [Gir87]. Having an optimal typing tree means that duplications are not needed in certain parts of the optimal typing tree. In the same way, in linear logic, we do not need to have exponentials everywhere: A simple type T can be translated as a type T^* of linear logic as follows:*

$$\boxed{\begin{array}{ll} \tau^* & := \tau \\ (T \to S)^* & := !T^* \multimap S^* \end{array}}$$

We can find a more refined translation; it can also be translated as T^+ and T^- as follow :

$$\boxed{\begin{array}{ll} \tau^+ & := \tau \\ (T \to S)^+ & := !T^- \multimap S^+ \end{array}} \qquad \boxed{\begin{array}{ll} \tau^- & := \tau \\ (T \to S)^- & := T^+ \multimap S^- \end{array}}$$

And we have in linear logic : $T^- \vdash T^*$ and $T^* \vdash T^+$. So the translation T^+ is sound and uses less exponentials that the usual and naive translation. In some way, it is more "optimal". The main drawback is that we cannot compose proofs of optimal translations easily.

3.6 Completeness

In order to prove the completeness of the typing system with respect to $\mathbf{SN}_{\lambda S}$, we first show that terms in normal form (for some adequate notion of normal form) can be typed, and then we prove Subject Expansion for a notion of reduction that can reduce any term in $\mathbf{SN}_{\lambda S}$ to a normal form (which we know to be typed). In λS, Subject Expansion is true only for $\to_{B,S}$ (not for $\to_W$). We will prove that it is enough for completeness. The main reason is that $\to_W$ can be postponed w.r.t. $\to_{B,S}$:

We could also define $\rightsquigarrow_E$ and $\Rightarrow_E$ just like in Chapter 2, but, by the fact that we have a better control over erasure, it is not necessary here: $\to_{B,S}$ satisfies Subject Expansion, and a term that cannot by reduced by $\to_{B,S}$ then we can easily type it. Therefore, we do not need to go the the normal form.

Theorem 10 (Postponing)**.**

1. *If $x \notin fv(M)$, then $M[x := N] \to_S^* \to_W M$.*
2. *If $M \to_W \to_B M'$, then $M \to_B \to_W M'$.*
3. *If $M \to_W \to_S M'$, then $M \to_S^+ \to_W^+ M'$.*
4. *If $M \longrightarrow^*_{S,W} M'$, then $M \longrightarrow^*_S \longrightarrow^*_W M'$.*

Proof. 1. By induction on M.

2. By the fact that the reduction rule $\to_W$ can only be applied with a variable ($x[y := N] \to_W x$) and by the fact that the rule $\to_B$ does not do any duplication or erasure, the application of the rules $\to_B$ and $\to_W$ can be commuted.
3. There are two cases where this result is not trivial:
   - If the rule $\to_S$ duplicates a sub-term N, and the application of the rule $\to_W$ was inside N. Then, we have $M \to_S \to_W^2 M'$.
   - The following case:
     $$(x[y := N_1])[x := N_2] \to_W x[x := N_2] \to_S N_2$$
     with $x \neq y$, $y \notin \text{fv}(N_2)$ and $x \in \text{fv}(N_1)$.
     Then, we have:
     $(x[y := N_1])[x := N_2] \to_S (x[x := N_2])[y := N_1[x := N_2]] \to_S$
     $N_2[y := N_1[x := N_2]] \to_S^* \to_W N_2$
4. We prove this property by applying item 3., until the reduction is of the form $M \to_S^* \to_W^* M'$. However, we have to check that we cannot apply item 3., an infinite number of times on the same reduction.

   Let L a S, W non empty reduction sequence from M to M'. (if we have an empty sequence, then the result is trivial)

   If L is not a of the form $M \longrightarrow^*_S \longrightarrow^*_W M'$, then there exists a sub-sequence $M_1 \to_W \to_S M_2$ inside L and by using item 3, we can replace it by $M_1 \to_S^+ \to_W^+ M_2$ to obtain a non empty reduction sequence L' that also goes from M to M'. Therefore we have a non-deterministic rewriting on non empty S, W-reduction sequences L from M to M'.

   This rewriting increases or does not change the size of L. According to Lemma 15, M is strongly normalising for S, W. Therefore, after a certain number of steps, the size of L does not change. So, after a certain number of steps, the rewriting is just replacing a sub-seuqence $M_1 \to_W \to_S M_2$ inside L by $M_1 \to_S \to_W M_2$ and this terminates. Hence, this rewriting terminates.

   By taking a normal form of this rewriting we have $M \longrightarrow^*_S \longrightarrow^*_W M'$.

□

Therefore, the normal forms for $\to_{B,S}$ are "normal enough" to be easily typed:

Lemma 18 (Typability of B, S-normal term)**.**
If M cannot be reduced by $\to_{B,S}$, then there exist Γ and A such that $\Gamma \vdash_{opt} M : A$.

Proof. First, we prove the following intermediate results:

1. If M cannot be reduced by $\to_{B,S}$, then M is of one of the following form:
   - $\lambda x.M_1$
   - $x[y_1 := M_1] \cdots [y_n := M_n] N_1 \cdots N_m$ with $y_1, \ldots, y_n$ fresh distinct variables.
2. If $\Gamma \vdash_{\text{opt}} M : A$, then there exist Δ and B such that $\Delta \vdash_{\text{opt}} \lambda x.M : B$.
3. If $\Gamma_1 \vdash_{\text{opt}} M_1 : A_1, \ldots \Gamma_2 \vdash_{\text{opt}} M_n : A_n$, $\Delta_1 \vdash_{\text{opt}} N_1 : B_1, \ldots \Delta_m \vdash_{\text{opt}} N_m : B_m$,then there exists Δ and C such that $\Delta \vdash_{\text{opt}} x[y_1 := M_1] \cdots [y_n := M_n] N_1 \cdots N_m : C$.

Proofs:

1. The result is proved by induction on M. We are in one of the following cases:
   - M is of the form $\lambda x.M_1$. Then, we can conclude.
   - M is of the form x. Then, we can conclude.

- M is of the form M_1M_2. By induction hypothesis on M_1, we are in one of the following cases:
  - M_1 is of the form $\lambda x.M_3$. Then, $M \to_B M_3[x := M_2]$. Contradiction.
  - M_1 is of the form $x[y_1 := M_1'] \cdots [y_n := M_n']N_1 \cdots N_m$. Then, we can conclude.
- M is of the form $M_1[x := M_2]$. By induction hypothesis on M_1, we are in one of the following cases:
  - M_1 is of the form $\lambda y.M_3$, with $x \neq y$ and $y \notin \text{fv}(M_2)$. Then, $M \to_S \lambda y.M_3[x := M_2]$. Contradiction.
  - M_1 is of the form $z[y_1 := M_1'] \cdots [y_n := M_n']N_1 \cdots N_m$ with $y_1, \ldots y_n$ fresh variables and $m > 0$. Then, there exist N ($N = N_m$ is $x \notin \text{fv}(N_m)$), and $N = N_m[x := M_2]$ if $x \in \text{fv}(N_m)$) such that $M \to_S (z[y_1 := M_1'] \cdots [y_n := M_n']N_1 \cdots N_{m-1})[x := M_2]N$.
  - M_1 is of the form $z[y_1 := M_1'] \cdots [y_n := M_n']$ with $y_1, \ldots y_n$ fresh variables, and with $x = z$ or there exists i such that $x \in \text{fv}(M_i')$. Then, M can be reduced by $\to_S$.
  - M_1 is of the form $z[y_1 := M_1'] \cdots [y_n := M_n']$ with $y_1, \ldots y_n$ fresh variables, $x \neq z$ and for all i, $x \notin \text{fv}(M_i')$. Then, we can conclude.

2. By hypothesis, A is of the form A^+, and Γ is of the form Γ^{--}. We are in one of the following cases:
   - $x \in \mathsf{Dom}(\Gamma)$. Then, Γ^{--} is of the form $\Delta^{--}, x : B^{--}$. Therefore, we have $\Delta^{--} \vdash_{\text{opt}} \lambda x.M : B^{--} \to A^+$.
   - $x \notin \mathsf{Dom}(\Gamma)$. Then, Γ^{--} is of the form $\Gamma^{--}, x : \omega$. Therefore, we have $\Gamma^{--} \vdash_{\text{opt}} \lambda x.M : \tau \to A^+$.

3. The types $A_1, \ldots A_n$ are of the form $A_1^+, \ldots A_n^+$ and can be used as forgetten types in the final typing tree. The types $B_1, \ldots B_m$, are of the form $B_1^+, \ldots B_m^+$, Therefore, F defined by $F := B_1 \to \ldots \to B_m \to \tau$, is of the form A^-. In $\Gamma_1, \ldots, \Gamma_n$, $\Delta_1, \ldots \Delta_m$, the types of x are of the form U^{--}. Hence, if we choose the intersection of F, of all the $\Gamma_i(x)$ and of all the $\Delta_i(x)$ for the type of x, we can conclude.

By induction on M and by using the items 1, 2 and 3, we can prove the lemma. □

Theorem 11 (Subject Expansion)**.**

If $M \to_{B,S} M'$ and $\Gamma' \vdash M' : A$, then there exist $\Gamma \approx \Gamma'$ such that $\Gamma \vdash M : A$. Moreover, the optimality property is preserved.

Proof. First by induction on $\to_{B,S}$ and $\equiv$, then by induction on A.

We adapt the proof of Subject Reduction. The optimality property is preserved : indeed, since we are considering $\to_{B,S}$ and not $\to_W$, the interface (typing context, type of the term and forgotten types) is not changed and we do not add any subsumption.

□

Theorem 12 (Completeness)**.**

If $M \in \boldsymbol{SN}_{\lambda S}$, then there exist Γ and A such that $\Gamma \vdash_{opt} M : A$.

Proof. By induction on the size of the longest reduction sequence of M. If M can be reduced by $\to_{B,S}$ we can use the induction hypothesis and Theorem 11. Otherwise, M is typable by Lemma 18.

□

Corollary 1. *If $M \to_{B,S} M'$ and $M' \in \boldsymbol{SN}_{\lambda S}$, then $M \in \boldsymbol{SN}_{\lambda S}$.*

Proof. By Theorem 12, M' is typable with an optimal typing tree. By Theorem 11, M is also typable with an optimal typing tree. Therefore, by Theorem 9, $M \in \mathbf{SN}_{\lambda S}$.

□

3.7 Complexity

In λS, we take advantage of the fact that $\to_W$ can be postponed w.r.t. to $\to_{B,S}$ steps. This allows us to concentrate on $\to_{B,S}$ and the normal forms for it.

Lemma 19 (Refined Subject Reduction)**.**
If $\Gamma \vdash^n_{ns} M : A$ then:

- *If $M \to_B M'$, then there exist Γ' and m such that $\Gamma \approx \Gamma'$, $m < n$ and $\Gamma' \vdash^m_{ns} M' : A$.*
- *If $M \to_S M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n_{ns} M' : A$*

Proof. We simply check that, in the proof of of Subject Reduction (Theorem 8), the absence of subsumptions and the context (modulo equivalence) is preserved. This comes from the fact that only the rule $\to_W$ can create subsumptions and without any subsumptions the rule $\to_B$ does not reduce the context.

□

Lemma 20 (Most inefficient reduction)**.**
Assume $\Gamma \vdash^n_{opt} M : A$. If M can be reduced by $\to_B$ and not by $\to_S$, then there exist M' and Γ' such that $\Gamma \approx \Gamma'$, $M \to_B M'$ and $\Gamma' \vdash^{n-1}_{opt} M' : A$.

Proof. We follow the proof by induction given in the proof of Subject Reduction (Theorem 8). In this induction, n can be decreased by more than 1 by a $\to_B$ in two cases:

- In the case where the type is an intersection, then n will be decreased by at least 2.
- When we build a typing of $M[x := N]$ from a typing of $(\lambda x.M)N$: if there were subsumption in the typing the λ-abstraction, then the proof calls Lemma 16.4 which might decrease n by more than 1.

Those two cases are never encountered when optimality is assumed, as we prove the result by induction on M. Since M cannot be reduced by $\to_S$, it is of one of the following forms:

- $\lambda x.M_1$. It is clear that M_1 satisfies the necessary conditions to apply the induction hypothesis.
- $(\lambda x.M_1)N_1 \dots N_p$ (with $p \geq 1$). We reduce to $M_1[x := N_1]\ N_2 \dots N_p$. By the optimality property, A is not an intersection, and none of the types of $((\lambda x.M_1)N_1 \dots N_i)_{1 \leq i \leq p-1}$ are intersections either (since they are applied to an argument). Also by the optimality property, there is no subsumption in the typing of the λ-abstraction, and therefore the call to Lemma 16.4 is replaced by a call to Lemma 17 and therefore n is decreased by exactly 1.
- $x[y_1 := N_1]...[y_p := N_p]N_{p+1}...N_m$. Therefore there exists i such that N_i can be reduced by $\to_B$. Moreover, optimality requires the type of x to be of the form $A_1^+ \to ... \to A_p^+ \to B-$, and therefore the sub-derivation typing N_i is also optimal: we can apply the induction hypothesis on it.

□

Lemma 21 (Resources of a normal term)**.**
If $\Gamma \vdash^{n}_{opt} M : A$*, and* M *cannot be reduced by* $\to_{B,S}$*, then* n *is the number of applications in* M*.*

Proof. Straightforward.

□

Theorem 13 (Complexity result)**.**
If $\Gamma \vdash^{n}_{opt} M : A$*, then* $n = n_1 + n_2$ *where*

- n_1 *is the maximum number of* $\to_B$ *in a* B, S*-reduction sequence from* M
- n_2 *is the number of applications in the* B, S *normal form of* M*.*

Proof. The second item of Lemma 19 and Lemma 20, gives n_1, M' and n_2 such that:

- $M \to^{*}_{B,S} M'$ with n_1 B-steps and M' is the B, S normal form. (λS is confluent)
- $n = n_1 + n_2$.
- $\Gamma \vdash^{n_2}_{\text{opt}} M' : A$.

To go from M to M':

- If we can reduce by $\to_S$, then we apply Lemma 19 and the measure on the typing tree does not change.
- If we cannot reduce by $\to_S$ and we can reduce by $\to_B$, then we apply Lemma 20 and the measure on the typing tree strictly decreases by 1.
- We repeat the process until we reach the B, S-normal form.

By Lemma 21, n_2 is the number of applications in M'.

Assume we have a B, S-reduction sequence, from M to M', with m B-steps. By applying successively Subject Reduction (Theorem 8), we can give a typing tree of M' of measure n' such that $n - n' \geq m$. To type M', we need at least one application rule by application in M'. Therefore, we also have $n_2 \leq n'$. Hence, by the fact that $n - m \geq n'$, we have $n - m \geq n_2$ and $n - n_2 \geq m$. Therefore $n_1 \geq m$.

Assume we have a B, S-reduction sequence, from M to any term M_1. It can be completed into a B, S-reduction sequence with more B-steps, from M to the B, S-normal form of M_1.

□

3.8 Conclusion

By adapting the typing system of intersection types given in Chapter 2 in a calculus with explicit substitutions and by measuring the number of B-step instead of β-steps, we have:

- Completeness is easier to prove: By construction of the calculus, we directly have the reductions that satisfy Subject Expansion.
- We have a more precise result about complexity (Theorem 13): We have more than just a bound.

A pure λ-term M is also a term of λS and the typing judgments of M with the typing system of Chapter 2 are exactly the ones with the typing system of this chapter (because there are no explicit substitutions inside M). We may try to use the more precise complexity result of this chapter to have a more precise result on the pure λ-calculus. Unfortunately, the obtained result depends on the B, S-normal

form. Therefore, it is not independent of λS. The main reason is that we have to keep track of the erased terms one way or another.

Actually, there is a way to go around this problem [BL13]: we can define a notion of *degree* of a typing tree and consider only optimal typing tree of minimal degree (such a typing tree is called *principal*). Then, a variation of Theorem 13 gives a result independent from the B, S-normal form and that can be used to have a result on the pure λ-calculus.

To have a precise complexity result for the pure λ-calculus we are going to use a different approach: In the next chapter, instead of refining the calculus like we did in this chapter, we are going to refine the intersection types.

Chapter 4

A big-step operational semantics via non-idempotent intersection types

We present a typing system of non-idempotent intersection types that characterises strongly normalising λ-terms and can been seen as a big-step operational semantics: we prove that a strongly normalising λ-term accepts, as its type, the structure of its normal form. As a by-product of identifying such semantical components in typing trees, we are able to define a trivial measure (the number of times a typing rule is applied) that exactly captures the length of the longest β-reduction sequences starting from a given typable term.

4.1 Introduction

Operational semantics describes the behaviour of programs using syntax, in other words without (explicitly) constructing a denotational model (for example, a Scott Domain [Sco82b]). While small-step operational semantics is based on the step-by-step evaluation of a program (usually by means of a rewrite system or an abstract machine), a big-step operational semantics directly relates a program to its final value / normal form (if it exists): judgements of the form $M \downarrow v$ (where M is a program and v the value of M after computation) can be derived with inference rules such as

$$\frac{M_1 \downarrow \lambda x.M_3 \quad M_2 \downarrow M_4 \quad M_3\{x := M_4\} \downarrow M_5}{M_1 M_2 \downarrow M_5}$$

in the case of λ-calculus.

In this chapter we provide a typing system for the λ-calculus that can serve as an inference system deriving a big-step semantics for all strongly normalising λ-terms.

Not only is this system following the traditional concepts and notations of typing, it more fundamentally differs from standard big-step semantics in that at no point of derivation trees is a substitution ever computed (unlike the rule above).

Moreover, while most big-step semantics correspond to an (often strict) evaluation strategy that is neither the least nor the most efficient, our typing system is related to longest β-reduction sequences. It therefore provides a semantics to strongly normalising terms only, but on the other hand it provides quantitative

information about reduction: a trivial measure on the typing derivation relating a term M to its semantics provides the exact length of the longest β-reduction sequences starting from M.

To build this semantics for all strongly normalising terms, we naturally use intersection types.

In Chapter 2 and in one paper [BL11a], we gave a typing system that (unlike de Carvalho's) characterises strongly normalising terms, and a trivial measure on typing trees (the number of uses of the rule typing applications) strictly decreases with each β-reduction. Therefore, strong normalisation is a corollary of Subject Reduction and the measure provides a bound on the length of the longest β-reduction sequences.

Refining the bound into an exact measurement was the challenge of the latter part of [BL11a], which provided a convoluted solution: First, the exact length of the longest reduction sequences could only be obtained by considering the typing trees satisfying a particular condition called *optimality* (which is the same notion of optimality given in Chapter 3). Then, another quantitative information about typing trees, the *degree*, had to be read, and the exact length was finally computed as *the measure of an optimal typing tree of smallest degree minus that degree*. Moreover, the result was obtained indirectly via the study of the above notions in Church-Klop's λI-calculus [KvOvR93] and then transposed into the pure λ-calculus by a sophisticated simulation.

However, it can be noticed that

- the degree is a quantitative data related to the normal form of a term;
- going via the λI-calculus is required because longest β-reduction sequences in the pure λ-calculus sometimes have to erase sub-terms in normal form.

Working with λS instead of the λI-calculus has been discussed at the end of Chapter 3 and done in [BL13].

The present chapter, we develops the idea of incorporating information about normal forms into typing trees: for every normal form M, its structure v yields a special type $[v]$.[1]

Pushing this idea further leads to a non-idempotent intersection typing system where typing trees can be viewed as the derivations of a big-step semantics.

As a by-product, the system improves the results of [BL11a] and simplifies their proofs: the length of the longest β-reduction sequences starting from a pure and strongly normalising λ-term M is simply the number of times a typing rule occurs in an optimal typing tree for M. The notion of optimality itself is actually much simpler, the notion of degree becomes useless, and no detour via λI is required.

The trick is that the typing tree for a given term directly indicates which λ-abstractions and applications will be consumed by β-reduction and which will remain in the normal form. These additional typing rules ensure that the measure of an optimal typing tree for a term in normal form is zero.

This enhancement of the typing system does not make the proofs of Soundness (typable implies strongly normalising) and Completeness (strongly normalising implies typable) more complicated.

In contrast to other semantics built with intersection types (for example the denotational semantics given in Chapter 2 or in [BCDC83, BL11b]), this semantics does not use filters of intersection types.

In Section 4.2, we give the basic definitions and properties. In Section 4.3, we prove Soundness (typable implies strongly normalising) and Completeness (strongly normalising implies typable). In Section 4.4, we give more refined properties in the case where the typing tree is optimal. In particular, we prove the Complexity

[1] The structure of a normal form is the normal form where all variable names are forgotten.

$$\frac{x \in \mathrm{fv}(M)}{(\lambda x.M)N \rightsquigarrow_h M\{x := N\}} \qquad \frac{x \notin \mathrm{fv}(M) \quad N \Rightarrow_h N'}{(\lambda x.M)N \rightsquigarrow_h (\lambda x.M)N'}$$

$$\frac{x \notin \mathrm{fv}(M) \quad N \text{ cannot be reduced by } \rightarrow_\beta}{(\lambda x.M)N \rightsquigarrow_h M}$$

$$\frac{M \rightsquigarrow_h M'}{MN \rightsquigarrow_h M'N} \qquad \frac{\mathrm{acc}(M) \quad N \Rightarrow_h N'}{MN \rightsquigarrow_h MN'}$$

$$\frac{M \rightsquigarrow_h M'}{M \Rightarrow_h M'} \qquad \frac{M \Rightarrow_h M'}{\lambda x.M \Rightarrow_h \lambda x.M'}$$

Figure 4.1: Perpetual reduction

Result (Theorem 19) and the relation between the type of a term and its normal form (Theorem 20). In Section 4.5, we give a brief study of alternative definitions of the typing system.

4.2 Basic definitions and properties

In Section 4.2.1, we review basic concepts of λ-calculus. In Section 4.2.2, we define the intersection types used in this chapter, the contexts of the typing judgements and we prove some basic properties. In Section 4.2.3, we present the typing system and its basics properties.

4.2.1 Syntax

In this chapter we work with the pure λ-calculus which is defined and used in Chapter 2 (see Section 2.2).

We proceed towards the notion of the perpetual reduction which is used:

- to express the longest reduction sequences in Theorem 19
- as a reduction strategy to prove Theorem 17

In order to define it formally, we also define $\rightsquigarrow_h$. $\mathrm{acc}(M)$ is already defined in Section 2.2.

Definition 26.
$M \rightsquigarrow_h M'$ and $M \Rightarrow_h M'$ are mutually defined by the rules of Figure 4.1.

Notice that this is the same perpetual reduction strategy used in [BL11a, vRSSX99]. However, the presentation is different (we did not used $\rightsquigarrow_h$ in the definition). The presentation given in this chapter avoids informal notations such as $xM_1 \dots M_n$, which is here very cumbersome for case analyses: since there are two ways to type an application, there are 2^n ways to type the above.

Also notice that the perpetual reduction is a particular case of the reductions of Figure 2.5:

Remark 2.
If $M \Rightarrow_h M'$ (resp. $M \rightsquigarrow_h M'$), then there exists E such that $M \Rightarrow_E M'$ (resp. $M \rightsquigarrow_E M'$).

We could prove Subject Expansion for $\rightsquigarrow_E$ and $\Rightarrow_E$, like we did in Chapter 2, but working with $\rightsquigarrow_h$ and $\Rightarrow_h$ is enough for what we do in this chapter.

Examples for $\Rightarrow_h$:

- $(\lambda x.M)NN_1 \dots N_n \Rightarrow_h M\{x := N\}N_1 \dots N_n$ if $x \in \text{fv}(M)$.
- If $M \Rightarrow_h M'$, then $xM \Rightarrow_h xM'$.
- It is possible to have $M \Rightarrow_h M'$ without having $(\lambda x.M)N \Rightarrow_h (\lambda x.M')N$. For example, we have $(\lambda y.a)(xx) \Rightarrow_h a$ but we do not have $(\lambda x.(\lambda y.a)(xx))(\lambda z.zz) \Rightarrow_h (\lambda x.a)(\lambda z.zz)$. However, we do have $(\lambda x.(\lambda y.a)(xx))(\lambda z.zz) \Rightarrow_h (\lambda y.a)((\lambda z.zz)(\lambda z.zz))$.

Every lemma and theorem based on a syntactical analysis of a term in normal term uses the following lemma:

Lemma 22 (Shape of a normal form)**.**
If M cannot be reduced by $\rightarrow_\beta$, then

- *either we have $acc(M)$,*
- *or M is of the form $\lambda x.M_1$.*

Proof. By induction on M. See Appendix A. □

One of the reasons why $\Rightarrow_h$ is used in Theorems 17 and 19, is that it is possible to reach the normal form by using $\Rightarrow_h$ only. More formally:

Lemma 23 (Applicability of $\Rightarrow_h$)**.** *If M can be reduced by $\rightarrow_\beta$, then M can be reduced by $\Rightarrow_h$.*

Proof. We prove by induction on M that if M can be reduced by $\rightarrow_\beta$ then:

- If M is of the form $\lambda x.M_1$, then there exists M' such that $M \Rightarrow_h M'$.
- If not, then there exists M' such $M \rightsquigarrow_h M'$.

Therefore, in both cases there exists M' such that $M \Rightarrow_h M'$. See Appendix A. □

An optimal typing tree for a term M does not give the exact normal form M' of M but the *structure* of M' (Theorem 20), which is M' where the names of variables are forgotten. More formally:

Definition 27 (Structures)**.** *Structures v are defined by the following grammar:*

$$\begin{array}{rcl} v & ::= & \lambda v \mid k \\ k & ::= & \nabla \mid kv \end{array}$$

For every term M, $struct(M)$ is partially defined as follows:

$$\begin{array}{lcll} struct(x) & := & \nabla & \\ struct(\lambda x.M) & := & \lambda struct(M) & \\ struct(MN) & := & struct(M)struct(N) & (struct(M) \text{ is of the form } k) \end{array}$$

$struct(M)$ is coherent with α-equivalence because: If $struct(M)$ is well-defined, then $struct(M\{x := y\})$ is well-defined and $struct(M\{x := y\}) = struct(M)$.

Lemma 24 (Structure of a normal term)**.** *M cannot be reduced by $\rightarrow_\beta$ if and only if $struct(M)$ is well-defined. Moreover, if we have $acc(M)$, then $struct(M)$ is of the form k.*

Proof. By induction on M. See Appendix A. □

Remark 3.
- *$struct(\lambda f \lambda x.f^n x) = \lambda\lambda\nabla^n\nabla$.*

Hence, Church's integers all have different structures.

- *Structures cannot distinguish Church's booleans:*
  $struct(\lambda x.\lambda y.x) = struct(\lambda x.\lambda y.y) = \lambda\lambda\nabla$. *However, this problem disappears if booleans are encoded as Church's integers (*0 *for "false" and* 1 *for "true"). More generally, the inhabitants of any enumerated type can be encoded in a way such that their structures remain distinct.*
- *Similarly, Church's encoding of pairs is preserved in the structures:*
  Assume M *and* N *cannot be reduced. We have* $struct(\lambda x.xMN) = \lambda\nabla struct(M)struct(N)$. *Therefore, if* $struct(\lambda x.xMN) = struct(\lambda x.xM'N')$, *then* $struct(M) = struct(M')$ *and* $struct(N) = struct(N')$.
- *Consequently, the inhabitants of any algebraic data type (for example lists, trees, etc ...) can be encoded in a way such that their structures remain distinct.*

Therefore structures can be considered a reasonable semantics.

4.2.2 Intersection types and contexts

First, we define the intersection types used in this chapter and we show basic definitions and properties about them. Second, we define the contexts used in the typing derivations used in this chapter and we show basic definitions and properties about them.

4.2.2.1 Intersection types

Definition 28 (Intersection Types)**.**
In this chapter, intersection types are defined as follows:

$$\begin{array}{lcl} F, G & ::= & [v] \mid A \to F \\ A, B, C & ::= & F \mid A \cap B \\ U, V & ::= & A \mid \omega \end{array}$$

F-types, A-types and U-types are defined by grammar on the right, where v ranges over structures (Definition 27). With this grammar, $U \cap V$ is defined if and only if U and V are A-types. Therefore, by defining $A \cap \omega := A$, $\omega \cap A := A$ and $\omega \cap \omega := \omega$, we have $U \cap V$ defined for all U and V.

Some remarks:

- This grammar is an extension of the grammar for the intersection types given in Chapter 2.
- $[v]$ is a type used to give information about the normal form of a term and is used to type applications and abstractions that are not meant to be used in a β-reduction.

For example, $([\nabla(\lambda\nabla)] \to [\lambda\nabla]) \cap [\nabla]$ is an A-type.

To define the notion of *optimality*, which is used in Theorems 19 and 20, we need to define the notions of *input* types and *output* types:

- An input is an intersection of $[\nabla]$.
- An output is of the form $[v]$.

More formally:

Definition 29 (Inputs and outputs)**.**
The judgement $input(U)$ is defined with the following rules:

$$\frac{}{input([\nabla])} \qquad \frac{input(A) \quad input(B)}{input(A \cap B)} \qquad \frac{}{input(\omega)}$$

We write $output(F)$ if and only if F is of the form $[v]$.

To prove Subject Reduction and Subject Expansion (Theorems 14, 18 and 16) by using Lemmas 29 and 30, we have to define equivalence $\approx$ and inclusion $\subseteq$ between types. Here is the formal definitions and basic properties (notice that we do not have $A \approx A \cap A$):

We define $U \approx V$ with the same definition given in Chapter 2 (Definition 7).

Lemma 25 (Properties of $\approx$)**.**
We have all the properties of Lemma 2 plus items 10 and 11.

1. *Neutrality of ω: $U \cap \omega = \omega \cap U = U$.*
2. *Strictness of F-types: If $U \approx F$, then $U = F$.*
3. *Strictness of ω: If $U \approx \omega$, then $U = \omega$.*
4. *$\approx$ is an equivalence relation.*
5. *Commutativity of $\cap$: $U \cap V \approx V \cap U$*
6. *Associativity of $\cap$: $U_1 \cap (U_2 \cap U_3) \approx (U_1 \cap U_2) \cap U_3$*
7. *Stability of $\cap$: If $U \approx U'$ and $V \approx V'$, then $U \cap V \approx U' \cap V'$.*
8. *If $U \cap V = \omega$, then $U = V = \omega$.*
9. *If $U \cap V \approx U$, then $V = \omega$.*
10. *If $U \approx V$ and $input(U)$, then $input(V)$.*
11. *$input(U \cap V)$ if and only if $input(U)$ and $input(V)$.*

Proof.
- 10: By induction on $U \approx V$.
- 11: Straightforward.

The other items have been already proved in Lemma 2 □

We define $U \subseteq V$ the same definition given in Chapter 2 (Definition 8).

Lemma 26 (Properties of $\subseteq$)**.**
We have the properties as in Lemma 3 plus items 10 and 11.

1. *$\subseteq$ is a partial pre-order and $\approx$ is the equivalence relation associated to it: $U \subseteq V$ and $V \subseteq U$ if and only if $U \approx V$.*
2. *Projections: $U \cap V \subseteq U$ and $U \cap V \subseteq V$.*
3. *Stability of $\cap$: If $U \subseteq U'$ and $V \subseteq V'$, then $U \cap V \subseteq U' \cap V'$.*
4. *Greatest element: $U \subseteq \omega$.*
5. *If $U \subseteq V$ and $input(U)$, then $input(V)$.*

Proof. Straightforward. □

4.2.2.2 Contexts

We define the contexts (and the equivalence and inclusion between contexts) the same way we have defined them in Chapter 2 (Definition 9)

Definition 30 (Input contexts)**.**
We write $input(\Gamma)$ if and only if for all x, we have $input(\Gamma(x))$.

Lemma 27 (Properties of contexts)**.**
We have the same properties as in Lemma 4 plus items 10 and 11.

1. *$\approx$ for contexts is an equivalence relation.*

$$\frac{}{x : F \vdash^0 x : F}\,(Var) \qquad \frac{\Gamma \vdash^n M : A \quad \Delta \vdash^m M : B}{\Gamma \cap \Delta \vdash^{n+m} M : A \cap B}\,(\cap) \qquad \frac{}{\vdash^0 M : \omega}\,(\omega)$$

$$\frac{\Gamma, x : U \vdash^n M : F \quad A \subseteq U}{\Gamma \vdash^n \lambda x.M : A \to F}\,(Fun_1) \qquad \frac{\Gamma, x : U \vdash^n M : [v] \quad \mathrm{input}(U)}{\Gamma \vdash^n \lambda x.M : [\lambda v]}\,(Fun_2)$$

$$\frac{\Gamma \vdash^n M : A \to F \quad \Delta \vdash^m N : A}{\Gamma \cap \Delta \vdash^{n+m+1} MN : F}\,(App_1) \qquad \frac{\Gamma \vdash^n M : [k] \quad \Delta \vdash^m N : [v]}{\Gamma \cap \Delta \vdash^{n+m} MN : [kv]}\,(App_2)$$

Figure 4.2: Typing rules

2. *$\subseteq$ for contexts is a partial pre-order and $\approx$ is its associated equivalence relation: $\Gamma \subseteq \Delta$ and $\Delta \subseteq \Gamma$ if and only if $\Gamma \approx \Delta$.*
3. *Projections: $\Gamma \cap \Delta \subseteq \Gamma$ and $\Gamma \cap \Delta \subseteq \Delta$.*
4. *Alternative definition: $\Gamma \subseteq \Delta$ if and only if there exists a context Γ' such that $\Gamma \approx \Delta \cap \Gamma'$.*
5. *Commutativity of $\cap$: $\Gamma \cap \Delta \approx \Delta \cap \Gamma$.*
6. *Associativity of $\cap$: $(\Gamma_1 \cap \Gamma_2) \cap \Gamma_3 \approx \Gamma_1 \cap (\Gamma_2 \cap \Gamma_3)$.*
7. *Stability of $\cap$: If R is either $\approx$ or $\subseteq$, $\Gamma\ R\ \Gamma'$ and $\Delta\ R\ \Delta'$, then $\Gamma \cap \Delta\ R\ \Gamma' \cap \Delta'$.*
8. *Greatest context: $\Gamma \subseteq ()$.*
9. *$(\Gamma, x : U) \subseteq \Gamma$.*
10. *If $\Gamma \subseteq \Delta$ and input(Γ), then input(Δ).*
11. *If input(Γ) and input(Δ), then input($\Gamma \cap \Delta$)*

Proof. Straightforward. □

4.2.3 Typing system

We now have all the elements to present the typing system:

Definition 31 (Typing system)**.**
Assume Γ is a context, M is a term, n is an integer, and U is a U-type. The judgement $\Gamma \vdash^n M : U$ is inductively defined by the rules given in Figure 4.2.
We write $\Gamma \vdash M : U$ if there exists n such that $\Gamma \vdash^n M : U$.

Some remarks:

- (App_1) (resp. (Fun_1)) are the rules (App) (resp. (Fun)) given in Chapter 2(Figure 2.3).
- In $\Gamma \vdash^n M : U$, n is the number of uses of the rule (App_1) and it is the trivial measure on typing trees that we use.
- (App_2) (resp. (Fun_2)) is used to type applications (resp. abstractions) that are not meant to be used in a β-reduction.
- Another advantage in having the type ω for the presentation of the typing system: Without the notation U or V, we would have to duplicate each abstraction rules that types $\lambda x.M$ (one case where $x \in \mathrm{fv}(M)$ and one case where $x \notin \mathrm{fv}(M)$). That would make four rules instead of two.

To define *optimality*, we first need to formally define what the *absence of subsumptions* means:

Definition 32 (No subsumptions)**.**

In a derivation of $\Gamma \vdash^n M : U$*, we say that there are no subsumptions if and only if every occurrence of rule* (Fun_1)

$$\frac{\Delta, x : V \vdash^n M_1 : F \quad A \subseteq V}{\Delta \vdash^n \lambda x.M_1 : A \to F}$$

is such that:

- *either* $A = V$
- *or both* $V = \omega$ *and output*(A)*.*

If $\Gamma \vdash^n M : U$ *with no subsumptions, then we write* $\Gamma \vdash^n_{ns} M : U$ *and we write* $\Gamma \vdash_{ns} M : U$ *if there exists* n *such that* $\Gamma \vdash^n_{ns} M : U$*.*

Another way of expressing the absence of subsumptions is that (Fun_1) is only used in one of the two following ways:

$$\frac{\Gamma, x : A \vdash M : F}{\Gamma \vdash \lambda x.M : A \to F} \qquad \frac{\Gamma \vdash M : F \quad x \notin \mathsf{Dom}(\Gamma)}{\Gamma \vdash \lambda x.M : [v] \to F}$$

Typing satisfies the following basic properties:

Lemma 28 (Basic properties of typing)**.**

1. $\Gamma \vdash^n M : U \cap V$ *(resp.* $\Gamma \vdash^n_{ns} M : U \cap V$*) if and only if there exist* Γ_1*,* Γ_2*,* n_1 *and* n_2 *such that* $\Gamma = \Gamma_1 \cap \Gamma_2$*,* $n = n_1 + n_2$*,* $\Gamma_1 \vdash^{n_1} M : U$ *(resp.* $\Gamma_1 \vdash^{n_1}_{ns} M : U$*) and* $\Gamma_2 \vdash^{n_2} M : V$ *(resp.* $\Gamma_2 \vdash^{n_2}_{ns} M : V$*).*
2. *If* $\Gamma \vdash^n M : U$ *(resp.* $\Gamma \vdash^n_{ns} M : U$*) and* $U \approx V$*, then there exists* Δ *such that* $\Gamma \approx \Delta$ *and* $\Delta \vdash^n M : V$ *(resp.* $\Delta \vdash^n_{ns} M : V$*).*
3. *If* $\Gamma \vdash^n M : U$ *(resp.* $\Gamma \vdash^n_{ns} M : U$*) and* $U \subseteq V$*, then there exist* Δ *and* m *such that* $\Gamma \subseteq \Delta$*,* $m \leq n$ *and* $\Delta \vdash^m M : V$ *(resp.* $\Delta \vdash^m_{ns} M : V$*).*
4. *If* $\Gamma \vdash M : A$*, then* $\mathsf{Dom}(\Gamma) = fv(M)$*.*
5. *If* $\Gamma \vdash M : U$*, then* $\mathsf{Dom}(\Gamma) \subseteq fv(M)$*.*

Proof. 1. Straightforward.

2. By induction on $U \approx V$.
3. Corollary of 1 and 2.
4. By induction on $\Gamma \vdash M : A$.
5. Corollary of 4.

□

We now have all the elements to define *optimality*: the property that a typing tree should satisfy if it should be viewed as the derivation of a big-step semantics or if we want to read from it the length of the longest β-reduction sequences (Theorems 20 and 19).

Definition 33 (Optimal typing tree)**.**

Assume Γ *is a context,* n *is an integer,* M *is a term and* F *is a* F*-type.*
We write $\Gamma \vdash^n_{opt} M : F$ *if and only if:*

- *We have* $\Gamma \vdash^n_{ns} M : F$*.*
- *We have input*(Γ) *and output*(F)*.*

We write $\Gamma \vdash_{opt} M : F$ *if and only if there exists* n *such that* $\Gamma \vdash^n_{opt} M : F$*.*

Example:

$$\dfrac{\dfrac{\dfrac{}{x : [\lambda\nabla] \vdash^0 x : [\lambda\nabla]}\,(Var)}{\vdash^0 \lambda x.x : [\lambda\nabla] \to [\lambda\nabla]}\,(Fun_1) \qquad \dfrac{\dfrac{}{y : [\nabla] \vdash^0 y : [\nabla]}\,(Var)}{\vdash^0 \lambda y.y : [\lambda\nabla]}\,(Fun_2)}{\vdash^1_{\text{opt}} (\lambda x.x)(\lambda y.y) : [\lambda\nabla]}\,(App_1)$$

4.3 Characterisation of the typing system

In Section 4.3.1, we prove Subject Reduction and Soundness (typable implies strongly normalising). In Section 4.3.2, we prove Subject Expansion and Completeness (strongly normalising implies typable).

4.3.1 Soundness

As usual for the proof of Subject Reduction, we first prove a substitution lemma:

Lemma 29 (Substitution lemma)**.**
If $\Gamma, x : U \vdash^n M : A$ *and* $\Delta \vdash^m N : U$*, then there exists* Γ' *such that* $\Gamma' \approx \Gamma \cap \Delta$ *and* $\Gamma' \vdash^{n+m} M\{x := N\} : A$*.*

Proof. By induction on $\Gamma, x : U \vdash M : A$. The measure of the final typing tree is $n + m$ because, by the fact that the intersection types are non-idempotent, this proof does not do any duplications. See Appendix A. □

Theorem 14 (Subject Reduction)**.**
If $\Gamma \vdash^n M : A$ *and* $M \to_\beta M'$*, then there exist* Γ' *and* n' *such that* $\Gamma \subseteq \Gamma'$*,* $n > n'$ *and* $\Gamma' \vdash^{n'} M' : A$*.*

Proof. First by induction on $M \to_\beta M'$, then by induction on A. See Appendix A. The rules (Fun_2) and (App_2) do not create any problem because we have to use the rules (Fun_1) and (App_1) to type a β-redex. □

In Theorem 14, we have $n > n'$ because, by the fact that types are non-idempotent, we do not do any duplications in the proof of Subject Reduction. Therefore, by Subject Reduction, for each β-reduction, the measure of the typing tree strictly decreases and then, we have Soundness as a corollary.

Theorem 15 (Soundness)**.**
If $\Gamma \vdash^n M : A$*, then* $M \in SN_{\leq n}$*.*

Proof. Corollary of Theorem 14: We prove by induction on n that if $\Gamma \vdash^n M : A$ then $M \in \mathrm{SN}_{\leq n}$.

Let M' be a term such that $M \to_\beta M'$. By Theorem 14, there exist Γ' and n' such that $n' < n$ and $\Gamma' \vdash^{n'} M' : A$. By induction hypothesis, $M' \in \mathrm{SN}_{\leq n'}$. Hence, $M' \in \mathrm{SN}_{\leq n-1}$ because $n' \leq n - 1$.

Therefore, $M \in \mathrm{SN}_{\leq n}$. □

Theorem 15 gives us a bound on the length of the longest β-reduction sequences. For a more precise result, see Theorem 19.

4.3.2 Completeness

We prove Subject Expansion for $\Rightarrow_h$. We could prove Subject Expansion for a larger subset of $\rightarrow_\beta$. However, the main purpose of this chapter is to emphasize the new complexity result and Theorem 20. Therefore, we prove Subject Expansion for $\Rightarrow_h$, which is enough to prove Completeness (Theorem 17).

As usual for the proof of Subject Expansion, we first prove an anti-substitution lemma:

Lemma 30 (Anti-substitution lemma)**.**
If $\Gamma \vdash_{ns} M\{x := N\} : A$*, then there exist* Γ'*,* Δ *and* U *such that:*

- $\Gamma \approx \Gamma' \cap \Delta$.
- $\Gamma', x : U \vdash_{ns} M : A$ *and* $\Delta \vdash_{ns} N : U$.
- *For all* $y \notin \mathsf{Dom}(\Delta)$, $\Gamma(y) = \Gamma'(y)$.

Proof. First by induction on M, then by induction on A. See Appendix A. The last property is necessary to preserve the absence of subsumptions in the induction. □

Notice that, in Lemma 30, if $x \notin \mathrm{fv}(M)$, then $U = \omega$ and we do not have any typing information on N.

In $\Rightarrow_h$, if a term is erased, then it is a normal form. Therefore, to prove Subject Expansion, we need to be able to type normal terms. This is also used in Theorem 17. To prove this, we need to know what is the type of an accumulator (when the context is an input).

Lemma 31 (Typing accumulators)**.**
If $\Gamma \vdash M : F$*,* $\mathit{input}(\Gamma)$ *and* $\mathit{acc}(M)$*, then* F *is of the form* $[k]$.

Proof. By induction on $\mathrm{acc}(M)$. See Appendix A. □

Lemma 32 (Typing normal forms)**.**
If M *cannot be reduced by* $\rightarrow_\beta$*, then there exist* Γ *and* v *such that* $\Gamma \vdash_{opt} M : [v]$.

Proof. By induction on M. In particular, we use Lemma 22 and Lemma 31. See Appendix A. □

Theorem 16 (Subject Expansion)**.**
If $\Gamma' \vdash_{opt} M' : F$ *and* $M \Rightarrow_h M'$*, then there exists* Γ *such that* $\Gamma \subseteq \Gamma'$ *and* $\Gamma \vdash_{opt} M : F$.

Proof. $\Rightarrow_h$ is mutually defined with $\rightsquigarrow_h$. We therefore prove, by simultaneous induction on $M \Rightarrow_h M'$ and $M \rightsquigarrow_h M'$, a stronger statement that forms an appropriate induction hypothesis:

If $\Gamma' \vdash_{\mathrm{ns}} M : F$ and $\mathrm{input}(\Gamma')$, and if we are in one of the following cases:

- We have $M \rightsquigarrow_h M'$
- We have $M \Rightarrow_h M'$ and $\mathrm{output}(F)$.

Then, there exists Γ such that $\Gamma \subseteq \Gamma'$, $\mathrm{input}(\Gamma)$, and $\Gamma \vdash_{\mathrm{ns}} M : F$.

See the details in Appendix A.

□

Finally, we can prove Completeness:

Theorem 17 (Completeness)**.**
If $M \in \boldsymbol{SN}$*, then there exist* Γ *and* F *such that* $\Gamma \vdash_{opt} M : F$.

Proof. By induction on the size of the longest β-reduction sequences:

- If M cannot be reduced by $\to_\beta$: By Lemma 32, there exist Γ and F such that $\Gamma \vdash_{\text{opt}} M : F$.
- If M can be reduced by $\to_\beta$: By Lemma 23, there exists M' such that $M \Rightarrow_h M'$. By induction hypothesis, there exist Γ' and F such that $\Gamma' \vdash_{\text{opt}} M' : F$. By Theorem 16, there exists Γ such that $\Gamma \vdash_{\text{opt}} M : F$.

□

Remark 4. *The proof of Theorem 17 gives us an algorithm that builds an optimal typing tree from a strongly normalising term. Hence, if we consider an optimal typing tree as the derivation in a semantics, then this semantics is indeed an operational semantics, and β-reduction is only a tool to build the derivation.*

This algorithm is based on the worst reduction (see Section 4.4.1), i.e. not very efficient. But even if there may be more efficient ways of building a typing tree for M, they have to construct, at the end of the day, something whose size is at least the length of the longest β-reduction sequences. We are therefore only interested in this as a purely theoretical construction.

Notice that we could have proved completeness (a strongly normalising term is typable) without the notion of optimality. But proving it with optimality gives more value to Theorems 19 (Complexity Result) and 20 (Structure of the normal form of a typed term): Indeed, if a term is typable, then it is strongly normalising, so it is typable with an optimal typing tree, and therefore we can apply Theorems 19 and 20 to it.

4.4 Refined soundness

The purpose of this section is to prove results when we have an optimal typing tree. In Section 4.4.1, we prove a refined Subject Reduction property and the Complexity Result. In Section 4.4.2, we prove that the type of a term in an optimal typing gives the structure of its normal form, which allows us to see typing as a derivation of semantics.

4.4.1 Complexity

Proving the complexity result is shorter than that of [BL11a], as we benefit from the semantical elements that we have here added to typing trees.

To prove Theorem 18 (a refined version of Theorem 14), we need a refined Substitution Lemma to preserve the absence of subsumptions:

Lemma 33 (Refined Substitution Lemma)**.**
If $\Gamma, x : U \vdash^n_{ns} M : A$ and $\Delta \vdash^m_{ns} N : U$, then there exists Γ' such that:

- $\Gamma' \approx \Gamma \cap \Delta$.
- $\Gamma' \vdash^{n+m}_{ns} M\{x := N\} : A$.
- *For all $y \notin \mathsf{Dom}(\Delta)$, $\Gamma(y) = \Gamma'(y)$.*

Proof. By induction on $\Gamma, x : U \vdash_{\text{ns}} M : A$. We adapt the proofs of Lemmas 29 and 30. See Appendix A. □

The last property of Lemma 33 has the same purpose as in Lemma 30.

One of the main advantage of this typing system compared to the one in [BL11a] is that the measure of an optimal typing tree for a normal term is equal to zero:

Lemma 34 (Measure of normal forms)**.**
If $\Gamma \vdash^n M : F$*,* M *cannot be reduced by* $\to_\beta$*,* $\mathit{input}(\Gamma)$ *and* $\mathit{output}(F)$*, then* $n = 0$*.*

Proof. By induction on M with the use of Lemmas 22 and 31. Only rule (App_2) (resp. (Fun_2)) can be used to type an application (resp. an abstraction). Hence, the measure of the typing tree is indeed 0. See the details in Appendix A. □

In Theorem 14, the measure can decrease by more than one. However, under the assumption of optimality and using Lemma 34, we can prove a refined version of Subject Reduction:

Theorem 18 (Refined Subject Reduction)**.**
If $\Gamma \vdash^n_{opt} M : F$ *and* $M \Rightarrow_h M'$*, then there exists* Γ' *such that* $\Gamma \subseteq \Gamma'$ *and* $\Gamma' \vdash^{n-1}_{opt} M' : F$*.*

Proof. As in Theorem 16, we have a stronger result with $\rightsquigarrow_h$. We prove, by simultaneous induction on $M \Rightarrow_h M'$ and $M \rightsquigarrow_h M'$, the following statement that forms an appropriate induction hypothesis:

If $\Gamma \vdash^n_{\mathrm{ns}} M : F$, $\mathrm{input}(\Gamma)$, and if we are in one the following cases:

- We have $M \Rightarrow_h M'$ and $\mathrm{output}(F)$.
- We have $M \rightsquigarrow_h M'$.

Then, there exists Γ' such that $\Gamma \subseteq \Gamma'$, $\Gamma' \vdash^{n-1}_{\mathrm{ns}} M' : F$ and then, by Lemma 27.10, we have $\mathrm{input}(\Gamma')$.

We adapt the proof of Theorem 14.

In particular, if $\Rightarrow_h$ erases a term, then this term is a normal term and, by Lemma 34, the measure of its typing tree is equal to zero. Also, by the fact that there are no subsumptions, we never have to discard a part of a typing tree. Moreover, by the fact that in the induction, the type of a term is not an intersection, the induction does not have to deal with the intersection rule. All of these reasons make the measure strictly decrease by one and only one. See Appendix A for details. □

Therefore, we can have a refined version of Theorem 15:

Theorem 19 (Complexity Result)**.**
If $\Gamma \vdash^n_{opt} M : F$*, then* $M \in \boldsymbol{SN}_{=n}$*.*

Proof. By induction on n and by Theorem 18, and Lemmas 23 and 34, there exists a β-reduction sequence from M of length n. By Theorem 15, $M \in \mathrm{SN}_{\leq n}$. Therefore, $M \in \mathbf{SN}_{=n}$. □

Theorems 14 and 15 are still useful because they require a weaker hypothesis than Theorems 18 and 19. In fact, Theorems 15 and 18 are both used to prove Theorem 19.[2]

4.4.2 Viewing optimal typing as a big-step semantics

Section 4.4.1 gives us the length of the longest β-reduction sequences from an optimal typing tree. Similarly, optimality (using the refined Subject Reduction property) allows us to derive our final result: viewing the typing tree as the derivation of a big-step semantics.

We first need a result about normal forms:

[2]We can notice that, although the results of this section are a refined version of what we prove in Section 4.3.1, the structure of the lemmas and proofs is closer to the ones in Section 4.3.2.

Lemma 35 (Structure of a typed normal term)**.**
If $\Gamma \vdash_{opt} M : [v]$ *and* M *cannot be reduced by* $\to_\beta$*, then* $struct(M) = v$*.*

Proof. By induction on M: By Lemma 22, we have $\text{acc}(M)$ or M is of the form $\lambda x.M_1$. By Definition 33, we have $\Gamma \vdash_{\text{ns}} M : [v]$ and $\text{input}(\Gamma)$.

- If M is of the form $\lambda x.M_1$: Then, because $[v]$ is not an arrow $A \to F$, there exist U and v_1 such that $\text{input}(U)$, $v = \lambda v_1$ and $\Gamma, x : U \vdash_{\text{ns}} M_1 : [v_1]$. Hence, $\text{input}(\Gamma, x : U)$. Therefore, $\Gamma, x : U \vdash_{\text{opt}} M_1 : [v_1]$. By induction hypothesis, $\text{struct}(M_1) = v_1$. Therefore, $\text{struct}(M) = \text{struct}(\lambda x.M_1) = \lambda\text{struct}(M_1) = \lambda v_1 = v$.
- If M is of the form x: Then, we have $\Gamma = (x : [v])$. Hence, we have $\text{input}(x : [v])$ and $\text{input}([v])$. Therefore, $v = \nabla$ and we have $\text{struct}(x) = \nabla = v$.
- If M is of the form M_1M_2 with $\text{acc}(M_1)$: Then, we are in one of the two following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} M_1 : A \to [v]$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : A$. By Lemma 27.10, we have $\text{input}(\Gamma_1)$. By Lemma 31, $A \to [v]$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, k and v_1 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $v = kv_1$, $\Gamma_1 \vdash_{\text{ns}} M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v_1]$. By Lemma 27.10, we have $\text{input}(\Gamma_1)$ and $\text{input}(\Gamma_2)$. Therefore, $\Gamma_1 \vdash_{\text{opt}} M_1 : [k]$ and $\Gamma_2 \vdash_{\text{opt}} M_2 : [v_1]$. By induction hypothesis, $\text{struct}(M_1) = k$ and $\text{struct}(M_2) = v_1$. Therefore, $\text{struct}(M_1M_2) = \text{struct}(M_1)\text{struct}(M_2) = kv_1 = v$.

□

Theorem 20 (Structure of the normal form of a typed term)**.**
If $\Gamma \vdash_{opt} M : [v]$ *and* M' *is the normal form of* M*, then* $struct(M') = v$*.*

Proof. By Theorem 15, M is strongly normalising. We prove the result by induction on the longest β-reduction sequences.

- If M cannot be reduced by $\to_\beta$, then $M = M'$ and, by Lemma 35, we can conclude.
- If M can be reduced by $\to_\beta$, then, by Lemma 23, there exists M'' such that $M \Rightarrow_h M''$. By Theorem 18, there exists Γ'' such that $\Gamma \subseteq \Gamma''$ and $\Gamma'' \vdash_{\text{opt}} M'' : [v]$. M' is also the normal form of M''. By induction hypothesis, $\text{struct}(M') = v$.

□

Theorem 20 gives us the structure of the normal form of a term. A more precise result is discussed in Section 4.5.2.

We can also notice that non-idempotency and the absence of subsumptions is not used to prove Theorem 20.

Remark 5. *Assume* M *is a closed strongly normalising term such that its normal form is a Church integer. Then, there exists a type* A *such that, for all closed strongly normalising terms* N *whose normal form is a Church integer,* $\vdash_{opt} N : A$ *if and only if* M *and* N *are* β*-equivalent.*[3]

We can notice that our results are concerning β-reduction/expansion/ equivalence. It does not give any results on η-reduction/expansion/equivalence. Indeed, our typing system is not "η-friendly": we do not have subject reduction nor expansion for η.

[3]This is similar to a result by Salvati [Sal10] who provides a type system $\vdash_{\mathsf{Sal}}$ satisfying: given a simply-typed λ-term M, there exist a context Γ and a type A such that

$$\Gamma \vdash_{\mathsf{Sal}} N : A \qquad \text{if and only if} \qquad M =_{\beta\eta} N$$

4.5 Alternative systems

In Section 4.5.1, we give a simpler variant that provides the Complexity Result (still much simpler than in [BL11a]) without giving information about the normal form. In Section 4.5.2, we study the issue of how to obtain a more precise information about the normal form.

4.5.1 Variant with no information about the normal form

This chapter improves the Complexity Result of [BL11a] and brings a new result about the normal form of a term. But if the point is only improving [BL11a], we do not need to go as far; consider the grammar of F-types is defined as follows:

$$F, G ::= \ \nabla \ \mid \ \delta \ \mid \ A \to F$$

All the types $[kv]$ in this chapter are simply collapsed into ∇, while all the types $[\lambda v]$ are collapsed into δ.

We therefore defined $\mathrm{input}(U)$ if and only if U is an intersection of ∇ and $\mathrm{output}(U)$ if and only if $U = \nabla$ or $U = \delta$.

Moreover, the rules (Fun_2) and (App_2) are defined as follows:

$$\frac{\Gamma, x : U \vdash^n M : F \quad \mathrm{input}(U) \quad \mathrm{output}(F)}{\Gamma \vdash^n \lambda x.M : \delta} \qquad \frac{\Gamma_1 \vdash^{n_1} M_1 : \nabla \quad \Gamma_2 \vdash^{n_2} M_2 : F \quad \mathrm{output}(F)}{\Gamma_1 \cap \Gamma_2 \vdash^{n_1+n_2} M_1 M_2 : \nabla}$$

The proofs of Soundness, Completeness and the Complexity Result are similar.

4.5.2 Obtaining the exact normal form

It would be interesting to enrich the typing system of this chapter to improve Theorem 20, so that the type of a term gives the exact normal form instead of just its structure. To highlight the separation between terms and types, even when these denote terms in normal form, we use a different syntax in the enriched grammar for v and k:

$$\begin{array}{rcl} v & ::= & \lambda\mathfrak{x}.M \ \mid \ k \\ k & ::= & \mathfrak{x} \ \mid \ kv \end{array}$$

where $\mathfrak{x}, \mathfrak{y}, \mathfrak{z}$ are *labels*, which differ from λ-term variables in that we choose them to not be α-convertible in v and k (see below why). A naive way of modifying (Fun_2) is:

$$\frac{\Gamma, x : U \vdash^n M : [v] \quad \mathrm{input}_{\mathfrak{y}}(U)}{\Gamma \vdash^n \lambda x.M : [\lambda\mathfrak{y}.v]}$$

where $\mathrm{input}_{\mathfrak{x}}(U)$ means that U is an intersection of $[\mathfrak{x}]$. Now in order to give an optimal typing tree to $(\lambda x.\lambda y.xy)(\lambda z.z)$, indicating its normal form with the type $[\lambda\mathfrak{z}.\mathfrak{z}]$, we would like this instance of (Fun_2):

$$\frac{x : [\mathfrak{z}] \to [\mathfrak{z}], y : [\mathfrak{z}] \vdash xy : [\mathfrak{z}]}{x : [\mathfrak{z}] \to [\mathfrak{z}] \vdash \lambda y.xy : [\lambda\mathfrak{z}.\mathfrak{z}]}$$

which shows the need to not consider $\lambda\mathfrak{z}.\mathfrak{z}$ to be α-convertible (to $\lambda\mathfrak{y}.\mathfrak{y}$, for instance). Yet, avoiding name clashes and degeneration of types prompts for a side-condition for (Fun_2) of the form $\mathfrak{y} \notin \mathrm{fv}(\Gamma)$ (freshness of $\mathfrak{y}$), which unfortunately forbids the above example (so we would not have Completeness, for lack of Subject Expansion).

To solve this problem we need a subtler way of defining the freshness of $\mathfrak{y}$ in (Fun_2). We add a condition in the definition of an optimal typing tree: In the typing tree, a label $\mathfrak{y}$ must be used at most once in an occurrence of (Fun_2). This is a notion of *global freshness*, that is somewhat similar to Barendregt's convention for labels, and related to the idea that principal types can specify normal forms [NM04]. This condition can blend in the lemmas and theorems of the chapter:

- Completeness: We have enough fresh labels to type a normal term when needed.
- Soundness: Subject Reduction does not do any duplications (because of non-idempotency), and therefore, global freshness is preserved by (refined or not) Subject Reduction.

More formally we need to collect the labels $\mathfrak{y}$ used in (Fun_2). Therefore, we have typing judgements of the form $\Gamma \vdash^{n}_{\mathfrak{A}} MU$ with $\mathfrak{A}$ a finite multi-set of labels. This makes the lemmas and proofs harder to read and this is the reason we have chosen to keep the version with structures as the main theory.

Alternatively, we could work in a calculus that does not have λ-abstractions. For example, the calculus with only combinators (S, K, etc ...) and applications would be a good candidate.

4.6 Conclusion

We have presented a typing system of non-idempotent intersection types that characterises strongly normalising λ-terms and in which an (optimal) typing derivation provides

1. the exact length of the longest β-reduction sequences
2. the structure of the normal form

The typing system is an enhanced version of that introduced Chapter 2 (and in [BL11a] and [BL11b]), which improves and simplifies the result similar to one that was proved in [BL11a] and [BL13]. Indeed, we used fewer definitions and lemmas, and the measure of an optimal typing tree is exactly the length of the longest β-reduction sequences from the typed term.

Perhaps more importantly, introducing information about normal forms, within types, has turned the typing system into a system deriving a big-step semantics for strongly normalising terms (provided we accept the structure of normal forms as an acceptable semantics). This style of derivations (based on typing) is unusual in that it never computes a substitution, and it relates to worst-case reduction strategies (generating longest β-reduction sequences).

The next section is not directly related to this chapter but is another enhanced version of the typing system introduced in Chapter 2.

Chapter 5

Strong normalisation in a calculus with constructors and fixpoints via non-idempotent intersection types

We introduce a calculus with a fixpoint operator that allows to naturally write strongly normalising functions. In the study of this calculus, we introduce a typing system with non-idempotent intersection types that characterises strong normalisation.

5.1 Introduction

Pure λ-calculus is Turing Complete: for every computable function we can find a corresponding λ-term. In particular, we can construct a fixpoint operator to write recursive functions. For example, we can write:

$$\text{fix}(M) := (\lambda x.M(xx))(\lambda x.M(xx)) \quad x \notin \text{fv}(M)$$

We can also extend the λ-calculus with a fixpoint operator and a rewrite rule that has the same behavior as the previous construction:

$$\text{fix}_1(M) \to M(\text{fix}_1(M))$$

Of course, in the λ-calculus or in this extension, some terms are not strongly normalising. However, such a fixpoint is worse: Every term that contains $\text{fix}_1(M)$ is definitively not strongly normalising. It is weakly normalising at best. For example, it is the case for fact_1 defined by:

$$\text{fact}_1 := \text{fix}_1(\lambda f.\lambda n.\text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n-1))$$

Strong normalisation is a useful and desirable property: for example, with strict evaluation, we do no have to worry about the evaluation order (see Appendix 5.6). Therefore, we would like the ability to write recursive functions in a style as natural as possible, but with better strong normalisation properties.

First, we can choose to only reduce the cases where the fixpoint reduction can be applied. For example, we can have the operator $\text{fix}_2()$ and the rule:

$\text{fix}_2(M)N \to M\text{fix}_2(M)N$ (An argument is needed to trigger the reduction)

Then, for example, fact_2 defined by:

$$\text{fact}_2 := \text{fix}_2(\lambda f.\lambda n.\text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n-1))$$

is indeed strongly normalising. However, $\text{fact}_2 3$, for example, is not:

$$\text{fact}_2 3 \longrightarrow^+ \text{if } 3 = 0 \text{ then } 1 \text{ else } 3 \times \text{fact}_2(3-1) \rightarrow_{\dots}$$

So, this restriction is not enough. An other possible restriction is to have the operator $\text{fix}_3()$ with the following rule:

$$\text{fix}_3(M)N \rightarrow M\text{fix}_3(M)N \quad (N \text{ is either a constant or a constructor})$$

With fact_3 defined with $\text{fix}_3()$, $\text{fact}_3 n$ is strongly normalising for all integers n. This is the approach given, for example, in Coq and in [GL02]. However, this approach has some disadvantages: First, it is bound to the notion of constants and constructors for the calculus: We cannot consider pure λ-calculus with this fixpoint operator. Second, this does not work for some recursive functions that are supposed to terminate (see examples in Section 5.2.1.3). Therefore, in this chapter we choose another approach and use the operator $\text{fix}_4()$ defined with the rule:

$$\text{fix}_4(M)N \rightarrow MN\text{fix}_4(M)$$

Apparently, this does not change anything compared to $\text{fix}_2()$. For example, with fact_4 defined by:

$$\text{fact}_4 := \text{fix}_4(\lambda n.\lambda f.\text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n-1))$$

we have a problem similar to fact_2. However, if we construct fact_5 as follows:

$$\text{fact}_5 := \text{fix}_4(\lambda n.\text{if } n = 0 \text{ then } \lambda f.1 \text{ else } \lambda f.f(n-1))$$

then, for all integers n, $\text{fact}_5 n$ is strongly normalising. The trick is that f is replaced by fact_5 only after the condition $n = 0$ has been calculated. The only difference compared to the usual style is to put the λf inside the branches (see more examples in Section 5.2.1.3). Of course, we can still write non strongly normalising terms with $\text{fix}_4()$ but by knowing the trick, for every recursive function that terminates for an entry, we can easily write a term such that, applied to this entry, it is strongly normalising. We could do the same trick with $\text{fix}_2()$. However, as shown with fact_6 defined by:

$$\text{fact}_6 := \text{fix}_2(\lambda f.\lambda n.(\text{if } n = 0 \text{ then} \lambda g.1 \text{ else } \lambda g.n \times g(n-1))f)$$

we can notice that it is more natural to work with $\text{fix}_4()$.

Fixpoint operator $\text{fix}_4()$ has been introduced in [Ber09]. Even if this fixpoint operator is useful when it is the only thing added to pure λ-calculus, the calculus we present here also has constructors and matching. In this calculus, strong normalisation might not be the only thing we expect from a program: we do not want any crash. For example, we can have a normal form while having a λ-abstraction as the argument of a branch (see Definition 36). One way to deal with this is that every term M, which makes the program crash, reduces to itself: Therefore, it is not strongly normalising. Such a of reduction is interesting and it is not a naive definition. We would also like to study the naive version. Fortunately, most of the work between these two versions of reductions can be factorised. Therefore, we will study the two calculi.

We can prove strong normalisation of some small examples. However, we would like more systematic tools to prove terms to be strongly normalising: Unlike weak normalisation, exhibiting a reduction sequence to a normal form is not a proof of strong normalisation. Therefore, we introduce a refined version of the reduction that preserves strong normalisation both ways (Theorem 26). Using this tool to prove strong normalisation is more powerful than using I-filters (see Section 5.2.2.2) which are used for this purpose in Chapter 2 (and in[CS06, Ber09, BL11b]). In order to prove Theorem 26, we introduce a typing system with intersection types that characterises strong normalisation.

In Section 5.2, we present the calculus with its reductions and various examples. In Section 5.3, we introduce the typing system that characterize strong normalisation and we prove Theorem 26. In Section 5.5, we prove the confluence of the calculus. In Section 5.6, we give an example of implementation of this fixpoint operator in a real strict functional language.

$$
\begin{array}{lcl}
\mathrm{fv}(x) & := & \{x\} \\
\mathrm{fv}(\lambda x.M) & := & \mathrm{fv}(M) - \{x\} \\
\mathrm{fv}(MN) & := & \mathrm{fv}(M) \cup \mathrm{fv}(N) \\
\mathrm{fv}(\mathrm{fix}(M)) & := & \mathrm{fv}(M) \\
\mathrm{fv}(c) & := & \emptyset \\
\mathrm{fv}((C_1.M_1, \ldots, C_n M_n)) & := & (\mathrm{fv}(M_1) - \mathrm{fv}(C_1)) \cup \ldots \cup (\mathrm{fv}(M_n) - \mathrm{fv}(C_n))
\end{array}
$$

Figure 5.1: Free variables

5.2 Calculus

This section is purely syntax-based (no typing involved). In Section 5.2.1 we define the calculus (the two versions of it). In Section 5.2.2 we introduce a refined version of the reduction as a tool to prove results of strong normalisation.

5.2.1 Definition of the calculus

In Section 5.2.1.1 we give the syntax of the terms of the calculus. In Section 5.2.1.2 we define the two versions of reductions. In Section 5.2.1.3 we give examples of terms that have a correct use of the fixpoint operator.

5.2.1.1 Syntax

In this section we define the terms of the calculus and the crash forms.

The calculus we present in this chapter is just pure λ-calculus enriched with constructors, matching and a fixpoint operator described in Section 5.1.

Definition 34 (Grammar of the calculus)**.**

Assume we have a set of constants c, an infinite set of variables x and a mapping from the constants to the integers that we call arity*.*

The terms *M, N of the calculus are defined with the following grammar:*

$$
\begin{array}{llll}
M, N & ::= & x \mid \lambda x.M \mid MN \mid \mathit{fix}(M) \mid c \mid \mathfrak{B} & c \text{ constant} \\
\mathfrak{B} & ::= & (C_1.M_1, \ldots, C_n.M_n) & \\
C & ::= & c x_1 \ldots x_n & c \text{ constant and } n \text{ arity of } c
\end{array}
$$

We naturally define the free variables of a term.

Definition 35 (Free variables)**.**

Assume M is a term.

The free variables *of M is a finite set of variables $fv(M)$ defined by induction on M as described in Figure 5.1.*

We consider terms up to α-equivalence.

To define one of the reductions, we need to identify ill-formed terms that we call crash forms.

Definition 36 (Crash forms)**.**

A crash form *is a term of one of the following forms:*

- *Wrong arity: $cM_1 \ldots M_{n+1}$ with n the arity of c*
- *Matching a function: $\mathfrak{B}(\lambda x.M)$, $\mathfrak{B}\mathfrak{B}$, $\mathfrak{B}(\mathit{fix}(M))$, $\mathfrak{B}(cN_1 \ldots N_n)$ with m arity of c and $n < m$*
- *Unknown pattern: $(c_1 \vec{x_1}.M_1, \ldots, c_n \vec{x_n}.M_n)(cN_1 \ldots N_m)$ such that for all i, $c_i \neq c$ and m arity of c.*

The set of crash forms is denoted CF.

$$\frac{}{(\lambda x.M)N \to_\alpha M\{x := N\}}\ (\beta) \qquad \frac{}{\mathrm{fix}(M)N \to_\alpha MN\mathrm{fix}(M)}\ (\mathrm{Fix})$$

$$\frac{\vec{N} = N_1 \dots N_m \quad m \text{ arity of } c_i}{(c_1\vec{x_1}.M_1, \dots, c_n\vec{x_n}.M_n)(c_i\vec{N}) \to_\alpha M_i\{\vec{x_i} := \vec{N}\}}\ (\mathrm{Match}) \qquad \frac{M \in \mathrm{CF}}{M \to_s M}\ (\mathrm{Crash\ Form})$$

$$\frac{M \to_\alpha M'}{MN \to_\alpha M'N} \qquad \frac{N \to_\alpha N'}{MN \to_\alpha MN'} \qquad \frac{M \to_\alpha M'}{\lambda x.M \to_\alpha \lambda x.M'} \qquad \frac{M \to_\alpha M'}{\mathrm{fix}(M) \to_\alpha \mathrm{fix}(M')}$$

$$\frac{M_i \to_\alpha M_i'}{(c_1\vec{x_1}.M_1, \dots, c_n\vec{x_n}M_n) \to_\alpha (c_1\vec{x_1}.M_1 \dots, c_i\vec{x_i}.M_i', \dots, c_n\vec{x_n}.M_n)}$$

Figure 5.2: Reductions

5.2.1.2 Reductions

In this section, we define the two reductions of the calculus in a modular way: Reduction are parametrised by a α which can be either u or s:

- If $\alpha = u$ (u for "unsafe"): Then the reduction is the naive one. Therefore, if M is strongly normalising for $\to_u$ then the execution of M terminates, but can also crash.
- If $\alpha = s$ (s for "safe"): Then the reduction $\to_s$ is $\to_u$ with the additional rule that if M is ill-formed, then $M \to_s M$ and therefore, M is not strongly normalising for $\to_s$. Hence, if M is strongly normalising for $\to_s$, then the execution of M terminates and does not crash.

More formally:

Definition 37 (Reductions)**.**
Assume α is either u or s and M and N are terms.
We define $M \to_\alpha N$ with the rules of Figure 5.2.

Theorem 21 (Relation between the two reductions)**.**

1. *If $M \to_u M'$ then $M \to_s M'$.*
2. *If $M \to_s M'$ then either $M \to_u M'$ or $M = M'$.*

Proof. 1. By induction on $M \to_u M'$.

2. By induction on $M \to_s M'$.

□

Definition 38 (Strong normalisation)**.**
Assume α is either s or u.
The set of strongly normalising *terms for $\to_\alpha$ is denoted $\boldsymbol{SN}\alpha$.*

Theorem 22 (Relation between the two strong normalisation)**.**
If $M \in \boldsymbol{SN}s$ then $M \in \boldsymbol{SN}u$.

Proof. Corollary of Theorem 21.1. □

Theorem 23 (Confluence)**.**
$\to_\alpha$ is confluent: If $M \longrightarrow^*_\alpha M_1$ and $M \longrightarrow^*_\alpha M_2$, then there exists M_3 such that $M_1 \longrightarrow^*_\alpha M_3$ and $M_2 \longrightarrow^*_\alpha M_3$.
Therefore, the normal form of M, if it exists, is unique.

Proof. With parallel reductions: similar to pure λ-calculus. See Appendix 5.5. □

5.2.1.3 Examples

Assume O, $True$ and $False$ are constructors of arity 0 and S is a constructor of arity 1.

$$\begin{array}{lcl} \text{add} & := & \lambda x.\text{fix}((O.\lambda f.x, Sy.\lambda f.S(fy))) \\ \text{mult} & := & \lambda x.\text{fix}((O.\lambda f.O, Sy.\lambda f.\text{add}(fy)x)) \\ \text{findmin} & := & \lambda g.\text{fix}(\lambda x.(\text{True}.\lambda f.x, \text{False}.\lambda f.f(Sx))(gx)) \end{array}$$

An integer n can be coded by the term S^nO. We use the following definition to express the fact that a term codes a function:

Definition 39 (Coding of a μ-recursive function)**.**
We say that M codes *with weak (resp. strong) normalisation a μ-recursive function f (of arity k) if and only if, for all integers $n_1, \ldots n_k$, if f is defined in $(n_1, \ldots, n_n)$ then $M(S^{n_1}O)\ldots(S^{n_k}O)$ is weakly (resp. strongly) normalising and its normal form is $S^{f(n_1,\ldots,n_k)}O$. Alternatively, we can also say that M is* correct *for f with weak (resp. strong normalisation).*

We can code the addition add and the multiplication mult. More generally we can code any primitive recursive function.

Intuitively, findmingx return the smallest y such that gy is true and $x \leq y$. More generally, with findmin, we can code any μ-recursive function f.

For any term M we cannot code fix(M) without fix() and keep strong normalisation properties. However, we can do it for specific M (generally the ones we are interested in). For example, we could use add_1 and findmin_1 defined by:

$$\begin{array}{lcl} M_1 & := & (O.\lambda f.x, Sy.\lambda f.S(fyf)) \\ \text{add}_1 & := & \lambda x.\lambda y.M_1 y M_1 \\ M_2 & := & \lambda x.(True.\lambda f.x, False.\lambda f.f(Sx)f)(gx) \\ \text{findmin}_1 & := & \lambda g.\lambda x.M_2 x M_2 \end{array}$$

But we can notice that it is much more simple to use fix().

In a program that use these examples, exhibiting a reduction sequence of $\to_\alpha$ to the normal form is as simple as if we had used a more usual fixpoint operator. Therefore, if we are not interested in strong normalisation, proving that these examples, written with this style, are correct with weak normalisation is as simple as proving that they are correct with the usual style.

The proof that they are correct with strong normalisation is given in Section 5.2.2.2.

Notice that with findmin defined with a fixpoint such as the one in Coq (named $\text{fix}_3()$ in Section 5.1) we would not have strong normalisation.

5.2.2 Refined notion of reduction

In Section 5.2.2.1, we define the refined notion of reduction. In Section 5.2.2.2, we illustrate how this reduction can be used to prove strong normalisation results.

5.2.2.1 Definition

Accumulators are terms which shape cannot be deconstructed. For example, a λ-abstraction is not an accumulator. Usually accumulators are used to study the shape of normal forms. In this chapter, they are also used to define the refined notion of reduction.

Definition 40 (Accumulators)**.**

Assume α is either s or u, M a term and E a set of variables with one element or less.

We define $accu_\alpha(M, E)$ as follows:

- *$accu_\alpha(M, \{x\})$ is denoted $accu_\alpha(M, x)$.*
- *$accu_\alpha(M, \emptyset)$ is denoted $accu_\alpha(M, \epsilon)$*
- *We write $accu_\alpha(M)$ if there exists E such that $accu_\alpha(M, E)$.*
- *We use the following rules:*

$$\frac{}{accu_\alpha(x, x)} \qquad \frac{accu_\alpha(M, E)}{accu_\alpha(MN, E)} \qquad \frac{accu_\alpha(M, E)}{accu_\alpha(\mathfrak{B}M, E)} \qquad \frac{M \in CF}{accu_u(M, \epsilon)}$$

A good way of understanding these rules is that $accu_\alpha()$ must satisfy Lemma 42.

We can have $M \to_\alpha M'$ with $M' \in \mathbf{SN}\alpha$ and $M \notin \mathbf{SN}\alpha$. Therefore, we define the refined reductions $\rightsquigarrow_\alpha$ and $\Rightarrow_\alpha$ (mutually defined) that preserve strong normalisation both ways.

- If $M \to_\alpha M'$ without any erasure then $M \rightsquigarrow_\alpha M'$.
- If $M \to_\alpha M'$ with the erasure of terms $N_1, \ldots, N_n$ then we have $M \rightsquigarrow_\alpha M'$ as long as $N_1, \ldots, N_n$ are normal forms and there is not conflict between the free variables of $N_1, \ldots, N_n$ and the bound variables of M. To determine if there is a conflict, we collect the free variables in a set E.
- If one of the N_i is strongly normalising, then we can reduce N_i by $\Rightarrow_\alpha$ until it reaches its normal form. Therefore, we have $M \rightsquigarrow_\alpha^* M'$.
- If there is a conflict, we have to use $\Rightarrow_\alpha$ which have stricter propagation rules.
- We also have some additional rules to be able to reduce any strongly normalising terms to its normal form.

More formally:

Definition 41 (Refined reductions)**.**

Assume α is either u or s, M and M' are terms and E a finite set of variables.

We define $M \rightsquigarrow_{E,\alpha} M'$ and $M \Rightarrow_{E,\alpha} M'$ with the rules given in Figure 5.3.

We write $M \rightsquigarrow_\alpha M'$ (resp. $M \Rightarrow_\alpha M'$) if and only if there exists E such that $M \rightsquigarrow_{E,\alpha} M'$ (resp. $M \Rightarrow_{E,\alpha} M'$).

We write $M \rightsquigarrow_{E,\alpha}^n M'$ if and only if there exists $E_1, \ldots E_n$, $M_1, \ldots, M_{n-1}$ such that $M \rightsquigarrow_{E_1,\alpha} M_1 \rightsquigarrow_{E_2,\alpha} \ldots \rightsquigarrow_{E_n,\alpha} M'$ and $E = E_1 \cup \ldots \cup E_n$. We give a similar definition for $M \Rightarrow_{E,\alpha}^n M'$.

A good way of understanding these rules is that $\rightsquigarrow_\alpha$ and $\Rightarrow_\alpha$ must satisfy Theorem 29.

Theorem 24 (Basic relations between the reductions)**.**

1. *If $M \rightsquigarrow_{E,\alpha} M'$ then $M \Rightarrow_{E,\alpha} M'$.*
2. *If $M \rightsquigarrow_\alpha M'$ then $M \Rightarrow_\alpha M'$.*
3. *If $M \Rightarrow_\alpha M'$ then $M \to_\alpha M'$.*
4. *If $M \rightsquigarrow_{E,\alpha} M'$ (resp. $M \Rightarrow_{E,\alpha} M'$) then $fv(M) = E \cup fv(M')$.*

Proof.

1. By definition of $M \rightsquigarrow_{E,\alpha} M'$ and $M \Rightarrow_{E,\alpha} M'$.
2. Corollary of 1.
3. We prove by induction on $M \Rightarrow_{E,\alpha} M'$ and $M \rightsquigarrow_{E,\alpha} M'$ that if $M \Rightarrow_{E,\alpha} M'$ or $M \rightsquigarrow_{E,\alpha} M'$ then $M \to_\alpha M'$.

$$\frac{x \in \mathrm{fv}(M)}{(\lambda x.M)N \rightsquigarrow_{\emptyset,\alpha} M\{x := N\}} \qquad \frac{x \notin \mathrm{fv}(M) \quad \mathrm{N}\not\longrightarrow_\alpha}{(\lambda x.M)N \rightsquigarrow_{\mathrm{fv}(N),\alpha} M} \qquad \frac{x \notin \mathrm{fv}(M) \quad N \Rightarrow_{E,\alpha} N'}{(\lambda x.M)N \rightsquigarrow_{E,\alpha} (\lambda x.M)N'}$$

$$\frac{\vec{x_i} = y_1 \dots y_m \quad \vec{N} = N_1 \dots N_m \quad X = \{\lambda \vec{x_j}.M_j \mid j \neq i\} \cup \{N_j \mid y_j \notin \mathrm{fv}(M_i)\} \quad \forall N' \in X, N' \not\longrightarrow_\alpha}{(c_1\vec{x_1}.M_1, \dots c_n\vec{x_n}.M_n)(c_i\vec{N}) \rightsquigarrow_{\bigcup_{N' \in X} \mathrm{fv}(N'),\alpha} M_i\{\vec{x_i} := \vec{N}\}}$$

$$\frac{i \neq j \quad M_j \Rightarrow_{E,\alpha} M'_j \quad \vec{x_j} = y_1 \dots y_m \quad E' = E - \{y_k \mid 1 \leq k \leq m\}}{(c_1\vec{x_1}.M_1, \dots c_n\vec{x_n}.M_n)(c_i\vec{N}) \rightsquigarrow_{E',\alpha} (c_1\vec{x_1}.M_1, \dots c_j\vec{x_j}.M'_j, \dots c_n\vec{x_n}.M_n)(c_i\vec{N})}$$

$$\frac{\mathfrak{B} = (c_1\vec{x_1}.M_1, \dots c_n\vec{x_n}.M_n) \quad x_i = y_1 \dots y_m \quad y_j \notin \mathrm{fv}(M_i) \quad N_j \Rightarrow_{E,\alpha} N'_j}{\mathfrak{B}(c_iN_1 \dots N_m) \rightsquigarrow_{E,\alpha} \mathfrak{B}(c_iN_1 \dots N'_j \dots N_m)}$$

$$\frac{}{\mathrm{fix}(M)N \rightsquigarrow_{\emptyset,\alpha} MN(\mathrm{fix}(M))} \qquad \frac{M \rightsquigarrow_{E,\alpha} M'}{M \Rightarrow_{E,\alpha} M'}$$

$$\frac{M \rightsquigarrow_{E,\alpha} M'}{MN \rightsquigarrow_{E,\alpha} M'N} \qquad \frac{N \rightsquigarrow_{E,\alpha} N'}{MN \rightsquigarrow_{E,\alpha} MN'} \qquad \frac{M \rightsquigarrow_{E,\alpha} M' \quad x \notin E}{\lambda x.M \rightsquigarrow_{E,\alpha} \lambda x.M'}$$

$$\frac{M \rightsquigarrow_{E,\alpha} M'}{\mathrm{fix}(M) \rightsquigarrow_{E,\alpha} \mathrm{fix}(M')} \qquad \frac{\vec{x_i} = y_1 \dots y_m \quad M_i \rightsquigarrow_{E,\alpha} M'_i \quad \forall j, y_j \notin E}{(c_1\vec{x_1}.M_1, \dots c_n\vec{x_n}.M_n) \rightsquigarrow_{E,\alpha} (c_1\vec{x_1}.M_1, \dots c_i\vec{x_i}.M'_i, \dots c_n\vec{x_n}.M_n)}$$

$$\frac{M \Rightarrow_{E,\alpha} M'}{\lambda x.M \Rightarrow_{E-\{x\},\alpha} \lambda x.M'} \qquad \frac{M \Rightarrow_{E,\alpha} M'}{\mathrm{fix}(M) \Rightarrow_{E,\alpha} \mathrm{fix}(M')} \qquad \frac{N_i \Rightarrow_{E,\alpha} N'_i \quad m \text{ arity of } c \quad n \leq m}{cN_1 \dots N_n \Rightarrow_{E,\alpha} cN_1 \dots N'_i \dots N_n}$$

$$\frac{\vec{x_i} = y_1 \dots y_m \quad M_i \Rightarrow_{E,\alpha} M'_i \quad E' = E - \{y_k \mid 1 \leq k \leq m\}}{(c_1\vec{x_1}.M_1, \dots c_n\vec{x_n}.M_n) \Rightarrow_{E',\alpha} (c_1\vec{x_1}.M_1, \dots c_i\vec{x_i}.M'_i, \dots c_n\vec{x_n}.M_n)}$$

$$\frac{\mathrm{accu}_\alpha(M, E_1) \quad N \Rightarrow_{E_2,\alpha} N'}{MN \rightsquigarrow_{E_1 \cup E_2,\alpha} MN'} \qquad \frac{\mathrm{accu}_\alpha(M, E_1) \quad \mathfrak{B} \Rightarrow_{E_2,\alpha} \mathfrak{B}'}{\mathfrak{B}M \rightsquigarrow_{E_1 \cup E_2,\alpha} \mathfrak{B}'M}$$

$$\frac{M \Rightarrow_{E,u} M' \quad MN \in \mathrm{CF}}{MN \rightsquigarrow_{E,u} M'N} \qquad \frac{N \Rightarrow_{E,u} N' \quad MN \in \mathrm{CF}}{MN \rightsquigarrow_{E,u} MN'} \qquad \frac{M \in \mathrm{CF}}{M \rightsquigarrow_{\emptyset,s} M}$$

Figure 5.3: Refined reductions

4. By induction on $M \rightsquigarrow_{E,\alpha} M'$ and $M \Rightarrow_{E,\alpha} M'$.

□

Lemma 36 (Shape of a normal form)**.**
If M cannot be reduced by $\rightarrow_\alpha$ then we are in one of the following cases:

- *M is of the form $\lambda x.M_1$, $\mathfrak{B}$, $fix(M_1)$, or $cM_1 \dots M_n$ with m arity of c and $n \leq m$.*
- *We have $accu_\alpha(M)$.*

Proof. By induction on M.

Assume M cannot be reduced by $\rightarrow_\alpha$.

If M is an application M_1M_2: Then M_1 and M_2 cannot be reduced by $\rightarrow_\alpha$. Hence we can use the induction hypothesis on M_1 and M_2.

Then we are in one of the following cases:

- M is a variable x: Therefore $\text{accu}_\alpha(x)$.
- M is a constant c.
- M is of the form $\lambda x.M_1$, $\mathfrak{B}$ or $\text{fix}(M_1)$.
- M is of the form $(\lambda x.M_1)M_2$. Therefore, $M \rightarrow_\alpha M_1\{x := M_2\}$. Contradiction.
- We have $M \in \text{CF}$:
  - If $\alpha = s$ then $M \rightarrow_\alpha M$. Contradiction.
  - If $\alpha = u$ then we have $\text{accu}_u(M)$.
- M is of the form $(c_1\vec{x_1}.M'_1, \dots, c_n\vec{x_n}.M'_n)(c_i\vec{N})$ with $\vec{N} = N_1 \dots N_m$ and m arity of c_i. Therefore, $M \rightarrow_\alpha M'_i\{\vec{x_i} := \vec{N}\}$. Contradiction.
- M is of the form $\mathfrak{B}M_2$ with $\text{accu}_\alpha(M_2)$. Therefore $\text{accu}_\alpha(\mathfrak{B}M_2)$.
- M is of the form $\text{fix}(M_3)M_2$. Therefore $M \rightarrow_\alpha M_3M_2\text{fix}(M_3)$. Contradiction.
- M is of the form $cM_1 \dots M_n$ with m arity of c and $n \leq m$.
- M is of the form M_1M_2 and we have $\text{accu}_\alpha(M_1)$. Therefore, we have $\text{accu}_\alpha(M_1M_2)$.

□

Lemma 37 ($\Rightarrow_\alpha$ can be used on a non-normal form)**.**
If M can be reduced by $\rightarrow_\alpha$ then there exists M' such that $M \Rightarrow_\alpha M'$.

Proof. By induction on M. See Appendix A. □

Lemma 38 (Reaching the normal form)**.**
If $M \in \boldsymbol{SN}\alpha$ then $M \Rightarrow_\alpha^* M'$ with M' the normal form of M.

Proof. By induction on the longest reduction from M of $\rightarrow_\alpha$ and as a corollary of Lemma 37.

□

Theorem 25 (Common uses of $\rightsquigarrow_\alpha$)**.**

1. *If $x \notin fv(M)$ and $N \in \boldsymbol{SN}\alpha$ then $(\lambda x.M)N \rightsquigarrow^+_{fv(N),\alpha} M$.*
2. *If $N \in \boldsymbol{SN}\alpha$, then there exist E such that $(\lambda x.M)N \rightsquigarrow^+_{E,\alpha} M\{x := N\}$ and $E \subseteq fv(N)$.*
3. *Assume $\vec{N} = N_1 \dots N_m$, $\vec{x_i} = y_1 \dots y_m$, m arity of c_i,*
   $X = \{\lambda\vec{x_j}.M_j \mid i \neq j\} \cup \{N_j \mid y_j \notin fv(M_i)\}$ and for all $N' \in X$, $N' \in \boldsymbol{SN}\alpha$. Then $(c_1\vec{x_1}.M_1, \dots, c_n\vec{x_n}.M_n)(c_i\vec{N}) \rightsquigarrow^+_{E,\alpha} M_i\{\vec{x_i} := \vec{N}\}$ with $E = \bigcup_{N' \in X} fv(N')$.

Proof. 1. By Lemma 38, there exists n such that $N \Rightarrow_\alpha^n N'$ with N' normal form of N. By Theorem 24.4, and by induction on n, there exists E such that $N \Rightarrow_{E,\alpha}^n N'$ and $\mathrm{fv}(N) = E \cup \mathrm{fv}(N')$. Hence, $(\lambda x.M)N \rightsquigarrow_{E,\alpha}^n (\lambda x.M)N'$. We also have, $(\lambda x.M)N' \rightsquigarrow_{\mathrm{fv}(N'),\alpha}^n M$. Therefore $(\lambda x.M)N \rightsquigarrow_{E\cup\mathrm{fv}(N'),\alpha}^{n+1} M$.

2. If $x \in \mathrm{fv}(M)$, then $(\lambda x.M)N \rightsquigarrow_{\emptyset,\alpha} M\{x := N\}$. If $x \notin \mathrm{fv}(M)$, then, by 1, $(\lambda x.M)N \rightsquigarrow_{\mathrm{fv}(N),\alpha}^+ M$.
3. We adapt the proof of 1.

□

Theorem 26 (Preservation of strong normalisation)**.**
If $M \Rightarrow_\alpha M'$ and $M' \in \boldsymbol{SN}\alpha$ then $M \in \boldsymbol{SN}\alpha$.

Proof. Proof given in Section 5.3.

□

We can notice, for example, that $(\lambda x.(\lambda y.a)(xx))(\lambda z.zz) \to_\alpha (\lambda x.a)(\lambda z.zz)$. However, we do not have $(\lambda x.(\lambda y.a)(xx))(\lambda z.zz) \Rightarrow_\alpha (\lambda x.a)(\lambda z.zz)$. Indeed, we have $(\lambda x.a)(\lambda z.zz) \in \mathbf{SN}\alpha$ but $(\lambda x.(\lambda y.a)(xx))(\lambda z.zz) \notin \mathbf{SN}\alpha$.

5.2.2.2 Discussion

To prove that a term is strongly normalising we only have to exhibit a sequence of $\Rightarrow_\alpha$ (or $\rightsquigarrow_\alpha$) to its normal form. To do this, Theorem 25 can be useful.

In Section 5.2.1.3, the reduction sequences of $\to_\alpha$ that we need in the proofs, are reduction sequences of $\Rightarrow_\alpha$. Therefore, proving that the examples of Section 5.2.1.3 are correct with strong normalisation is almost as easy as proving that they are correct with weak normalisation.

Let us compare this technique to prove strong normalisation to the technique that uses I-filters (used in Chapter 2 and in [CS06, Ber09, BL11b]) . We have the following tools:

- A denotational semantics: Each term can be interpreted as an I-filter.
- We have equations on I-filters.
- If the semantics of a term is different from $\bot$ (the empty I-filter) then M is strongly normalising.

A proof of strong normalisation that uses these tools is as follow: Assume we have a specific term M. By using equations on I-filters, we can prove that the semantics of M is equal to an I-filters which is trivially different from $\bot$. We can then conclude that M is strongly normalising.

If we can do such a proof, then we can use $\rightsquigarrow_\alpha$ to prove strong normalisation: each equation on the I-filters correspond to a $\rightsquigarrow_\alpha^+$. Therefore, using $\rightsquigarrow_\alpha$ is as powerful as using equations on I-filters in order to prove strong normalisation. It is even more powerful because, under certain conditions, we can reduce by $\rightsquigarrow_\alpha$ under λ abstractions.

5.3 Strong normalisation

The main goal of this section is to prove Theorem 26 via non-idempotent intersection types. In Section 5.3.1, we define the typing system. In Section 5.3.2, we prove Soundness via subject reduction: if a term is typable then it is strongly normalising. In Section 5.3.3, we prove Completeness via subject expansion: if a term is strongly normalising, then it is typable. Then the proof of Theorem 26 is straightforward.

5.3.1 Intersection types

In Section 5.3.1.1, we define the types, the contexts and give their basic properties. In Section 5.3.1.2, we introduce the typing rules and the basic properties of the system.

5.3.1.1 Types

Definition 42 (Intersection types)**.**
U-types, A-types and F-types are defined with the following grammar:

$$\begin{array}{lll} F,G,H & ::= & \delta \mid \nabla \mid cA_1 \dots A_n \mid A \to F \quad n \text{ arity of } c \\ A,B,C & ::= & F \mid A \cap B \\ U,V,W & ::= & A \mid \omega \end{array}$$

With this grammar, $U \cap V$ is defined if and only if U and V are A-types.

Therefore, if we write $A \cap \omega := A$, then $\omega \cap A := A$ and $\omega \cap \omega := \omega$, for all U and V, $U \cap V$ is defined.

Some remarks:

- We have extended the intersection types of Chapter 2.
- $cA_1 \dots A_n$ is used to type $cM_1 \dots M_n$.
- ∇ can only be the type of something that is or reduces itself to an accumulator (like $xM_1 \dots M_n$). In particular, ∇ cannot be the type of $\lambda x.M$.
- δ is used to type the base case of $\mathrm{fix}(M)$.

Equivalence and inclusion between types is defined with the same definition given in Chapter 2 and we have the same properties.

The same thing is done for contexts.

For factorization reasons, we define the following notion:

Definition 43 (Application of types)**.**
Assume α is either u or s, F and G are F-types, and A is a A-type.
$F@_\alpha A : G$ is defined with the following rules:

$$\frac{}{(A \to F)@_\alpha A : F} \qquad \frac{}{(c\vec{A})@_u B : F}$$

5.3.1.2 Typing rules

We want a typing judgment $\Gamma \vdash_u M : A$ that characterizes terms in $\mathbf{SN}u$ and another one $\Gamma \vdash_s M : A$ that characterizes terms in $\mathbf{SN}s$. By the fact that $\mathbf{SN}s \subseteq \mathbf{SN}u$, it is natural that the rules of $\Gamma \vdash_s M : A$ are a subset of the rules of $\Gamma \vdash_u M : A$. We can define the two typing systems by defining $\Gamma \vdash_\alpha M : A$ as follows:

Definition 44 (Typing judgements)**.**
Assume α is either u or s, Γ is a context, M is a term, n is an integer and U is a U-type.
We define $\Gamma \vdash^n_\alpha M : U$ with the rules of Figure 5.4.
We write $\Gamma \vdash_\alpha M : U$ if there exists n such that $\Gamma \vdash^n_\alpha M : U$.
We can notice that $\Gamma \vdash_\alpha M : \omega$ does not give any information on M and cannot be found inside a typing tree. It is only used to have more concise lemmas and proofs.

Some remarks:

- We have extended the typing system given in Chapter 2.
- The typing rule of c and the first typing rule of $\mathfrak{B}$ (rule (B_1)) is quite intuitive.

$$\frac{}{x : F \vdash^0_\alpha x : F} \qquad \frac{\Gamma, x : U \vdash^n_\alpha M : F \quad A \subseteq U}{\Gamma \vdash^n_\alpha \lambda x.M : A \to F} \qquad \frac{\Gamma \vdash^n_\alpha M : F \quad \Delta \vdash^m_\alpha N : A \quad F@_\alpha A : G}{\Gamma \cap \Delta \vdash^{n+m+1}_\alpha MN : G}\ (App)$$

$$\frac{\Gamma \vdash^n_\alpha M : A \quad \Delta \vdash^m_\alpha M : B}{\Gamma \cap \Delta \vdash^{n+m}_\alpha M : A \cap B} \qquad \frac{}{\vdash^0_\alpha M : \omega} \qquad \frac{n \text{ arity of } c}{\vdash^0_\alpha c : A_1 \to \ldots \to A_n \to cA_1 \ldots A_n}$$

$$\frac{\forall j \quad \Gamma_j \vdash^{m_j}_\alpha \lambda \vec{x_j}.M_j : A_{j,1} \to \ldots \to A_{j,n_j} \to F_j}{(\ldots(\Gamma_1 \cap \Gamma_2)\ldots) \cap \Gamma_n \vdash^{m_1+\ldots+m_n}_\alpha (c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n) : c_i A_{i,1} \ldots A_{i,n_i} \to F_i}\ (B_1)$$

$$\frac{\Gamma \vdash^n_\alpha \mathfrak{B} : A}{\Gamma \vdash^n_\alpha \mathfrak{B} : \triangledown \to F} \qquad \frac{\Gamma \vdash^n_u \mathfrak{B} : A_1}{\Gamma \vdash^n_u \mathfrak{B} : (A \to F) \to G} \qquad \frac{\Gamma \vdash^m_u (c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n) : B \quad \forall j, c_j \neq c}{\Gamma \vdash^m_u (c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n) : c\vec{A} \to F}$$

$$\frac{\Gamma \vdash^n_\alpha M : A}{\Gamma \vdash^n_\alpha \text{fix}(M) : \delta}\ (Fix_1) \qquad \frac{\Gamma \vdash^n_u \mathfrak{B} : A}{\Gamma \vdash^n_u \mathfrak{B} : \delta \to F}$$

$$\frac{\Gamma \vdash^n_\alpha M : F \quad F@_\alpha A : G \quad \Gamma \vdash^m_\alpha \text{fix}(M) : B \quad G@_\alpha B : H}{\Gamma \cap \Delta \vdash^{n+m+2}_\alpha \text{fix}(M) : A \to H}\ (Fix_2)$$

Figure 5.4: Typing rules

- The accumulators can be typed by any type F, in particular they can by typed by $\triangledown$.
- When $\alpha = u$, there are more terms that are strongly normalising than in the case where $\alpha = s$. Therefore, when $\alpha = u$, more terms are typed and it is natural to have more typing rules. Theses extra rules are needed to type crash forms (like $\mathfrak{B}(\lambda x.M)$).
- When $\alpha = s$, the application rule (App) is the same one as the one presented in Chapter 2.
- The rules to type $\text{fix}(M)$ are chosen as follow: From a typing of $\text{fix}(M)N$, we can create a typing of $MN\text{fix}(M)$. There is an intuitive typing rule of $\text{fix}(M)$ that satisfy this (the rule (Fix_2)). This rule use a typing of $\text{fix}(M)$ to give an other typing of $\text{fix}(M)$. For Completeness, we need an other rule that type the base of recursion (Fix_2): If M is typable, then $\text{fix}(M)$ is of type δ.

  When $\alpha = s$, δ is not used in any other rules. When $\alpha = u$, the only other use of δ is to type crash forms (like $\mathfrak{B}(\text{fix}(M))$).

Lemma 39 (Basic properties of typing)**.**

1. *If* $\Gamma \vdash^n_\alpha M : U \cap V$ *then there exist* Γ_1, Γ_2, n_1 *and* n_2 *such that* $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1}_\alpha M : U$ *and* $\Gamma_2 \vdash^{n_2}_\alpha M : V$.
2. *If* $\Gamma \vdash^n_\alpha M : U$ *and* $U \approx V$ *then there exists* Δ *such that* $\Gamma \approx \Delta$ *and* $\Delta \vdash^n_\alpha M : V$.
3. *If* $\Gamma \vdash^n_\alpha M : U$ *and* $U \subseteq V$ *then there exist* Δ *and* m *such that* $\Gamma \subseteq \Delta$, $m \leq n$ *and* $\Delta \vdash^m_\alpha M : V$.
4. *If* $\Gamma \vdash_\alpha M : A$ *then* $\mathsf{Dom}(\Gamma) = fv(M)$.
5. *If* $\Gamma \vdash^n_\alpha cM_1 \ldots M_k : F$ *with* m *arity of* c *and* $k \leq m$, *then there exist* $\Gamma_1, \ldots, \Gamma_k$, $n_1, \ldots, n_k$, $A_1, \ldots, A_m$ *and* G *such that* $\Gamma = (\ldots(\Gamma_1 \cap \Gamma_2)\ldots) \cap \Gamma_k$, $n = n_1 + \ldots + n_k$, *for all* i, $\Gamma_i \vdash^{n_i}_\alpha M_i : A_i$ *and* $F = A_{i+1} \to A_m \to G$.

6. *If* $\Gamma_1 \vdash_\alpha^{n_1} M_1 : A_1, \ldots, \Gamma_k \vdash_\alpha^{n_k} M_k : A_k$, *m arity of c and* $k \leq m$, *then* $(\ldots(\Gamma_1 \cap \Gamma_2)\ldots) \cap \Gamma_k \vdash_\alpha^{n_1+\ldots+n_k} cM_1 \ldots M_k : A_{k+1} \to \ldots A_m \to F$ *for every* $A_{k+1}, \ldots, A_m, G$.

Proof.

1. Straightforward.
2. By induction on $U \approx V$.
3. Corollary of 1 and 2.
4. By induction on $\Gamma \vdash_\alpha M : A$.
5. By induction on k.
6. By induction on k.

□

5.3.2 Soundness

Lemma 40 (Substitution lemma)**.**
If $\Gamma, x : U \vdash_\alpha^n M : A$ *and* $\Delta \vdash_\alpha^m N : U$, *then there exists* Γ' *such that* $\Gamma' \approx \Gamma \cap \Delta$ *and* $\Gamma' \vdash_\alpha^{n+m} M\{x := N\} : A$.

Proof. By induction on $\Gamma, x : U \vdash_\alpha M : A$. □

Theorem 27 (Subject reduction)**.**
If $\Gamma \vdash_\alpha^n M : A$ *and* $M \to_\alpha M'$, *then there exists* Γ' *and* n' *such that* $\Gamma \subseteq \Gamma'$, $n > n'$ *and* $\Gamma' \vdash_\alpha^{n'} M' : A$.

Proof. First by induction on $M \to_\alpha M'$ then by induction on A. See Appendix A.
□

Theorem 28 (Soundness)**.**
If $\Gamma \vdash_\alpha M : A$, *then* $M \in \boldsymbol{SN}\alpha$.

Proof. Corollary of Theorem 27: We prove by induction on n that if $\Gamma \vdash_\alpha^n M : A$ then $M \in \mathbf{SN}\alpha$.

Let M' be a term such that $M \to_\alpha M'$. By Theorem 27, there exist Γ' and n' such that $n' < n$ and $\Gamma' \vdash_\alpha^{n'} M' : A$. By induction hypothesis, $M' \in \mathbf{SN}\alpha$.

Therefore $M \in \mathbf{SN}\alpha$.

□

5.3.3 Completeness

Lemma 41 (Anti-substitution lemma)**.**
If $\Gamma \vdash_\alpha^n M\{x := N\} : A$, *then there exists* Γ_1, Γ_2 *and* U *such that* $\Gamma \approx \Gamma_1 \cap \Gamma_2$, $\Gamma_1, x : U \vdash_\alpha M : A$ *and* $\Gamma_2 \vdash_\alpha N : U$.

Proof. First induction on M, then by induction on A.

□

Lemma 42 (Typing accumulators)**.**
If $\Gamma, x_1 : U_1, \ldots, x_n : U_n \vdash_\alpha M : F$ *and* $accu_\alpha(M, \{x_1, \ldots, x_n\})$ *(we have* $n = 0$ *or* $n = 1$*), then for all* G *there exists* $U_1', \ldots, U_n'$ *such that* $\Gamma, x_1 : U_1', \ldots, x_n : U_n' \vdash_\alpha M : G$.

Proof. By induction on $\mathrm{accu}_\alpha(M, \{x_1, \ldots, x_n\})$. See Appendix A. □

Lemma 43 (Typing normal forms)**.**
If M cannot be reduced by $\to_\alpha$, then there exist Γ and F such that $\Gamma \vdash_\alpha M : F$.

Proof. By induction on M, using Lemma 36. See Appendix A. □

Theorem 29 (Subject expansion)**.**
Assume $\Gamma, x_1 : U_1, \ldots, x_n : U_n \vdash_\alpha M' : A$ and $E = \{x_1, \ldots, x_n\}$:

- *If $M \rightsquigarrow_{E,\alpha} M'$ then there exists Γ', U'_1, …, U'_n such that $\Gamma \approx \Gamma'$ and $\Gamma', x_1 : U'_1, \ldots, x_n : U'_n \vdash_\alpha M : A$.*
- *If $M \Rightarrow_{E,\alpha} M'$ then there exists Γ', U'_1, …, U'_n and B such that $\Gamma \approx \Gamma'$ and $\Gamma', x_1 : U'_1, \ldots, x_n : U'_n \vdash_\alpha M : B$.*

Proof. First by induction on $M \rightsquigarrow_{E,\alpha} M'$ and on $M \Rightarrow_{E,\alpha} M'$, then by induction on A. See Appendix A. □

Theorem 30 (Completeness)**.**
If $M \in \boldsymbol{SN}\alpha$ then there exist Γ and A such that $\Gamma \vdash_\alpha M : A$.

Proof. Corollary of Theorem 27 and Lemma 43: By induction on the size of longest reduction sequences of $\to_\alpha$ from M.

- If M cannot be reduced by $\to_\alpha$ then by Lemma 43 we can conclude.
- If M can be reduced by $\to_\alpha$, then by Lemma 37, there exists M' such that $M \Rightarrow_\alpha M'$. By Theorem 24.3, we have $M \to_\alpha M'$. Hence we can use the induction hypothesis: There exist Γ' and A' such that $\Gamma' \vdash_\alpha M' : A'$. By Theorem 29, there exist Γ and A such that $\Gamma \vdash_\alpha M : A$.

□

Corollary 2. *Theorem 26 is true.*

Proof. Corollary of Theorems 28, 30, and 29: Assume $M \Rightarrow_\alpha M'$. By Theorem 30, there exist Γ' and A' such that $\Gamma' \vdash_\alpha M' : A'$. By Theorem 29, there exist Γ and A such that $\Gamma \vdash_\alpha M : A$. Therefore, by Theorem 28, $M \in \mathbf{SN}\alpha$.

□

5.4 Conclusion

We have introduced a calculus with constructors, matching, and a fixpoint operator that allows to naturally write recursive functions that can be used in strongly normalising terms. To facilitate proofs of strong normalisation (by $\to_\alpha$), we have introduced a refined reduction $\Rightarrow_\alpha$ that preserves strong normalisation both ways. This was proved via a typing system with non-idempotent intersection types.

Also, $\mathrm{accu}_\alpha()$ and $\rightsquigarrow_\alpha$ could be used to define our own notion of reducibility candidates to prove strong normalisation of a polymorphic type system over the λ-calculus (à la System F). Moreover, for some specific M we could have a typing rule for $\mathrm{fix}(M)$. And then the soundness lemma in the proof of strong normalisation of the system for this rule would be just a use of $\Rightarrow^*_\alpha$.

5.5 Confluence

In this section, we prove confluence of $\to_\alpha$.

Definition 45 (Parallel reductions)**.**
Assume M and M' are terms, we define $M \mid\!\to M'$ with the rules of Figure 5.5.

$$\frac{}{x \mid\!\!\rightarrow x} \qquad \frac{}{c \mid\!\!\rightarrow c} \qquad \frac{M \mid\!\!\rightarrow M'}{\lambda x.M \mid\!\!\rightarrow \lambda x.M'} \qquad \frac{M \mid\!\!\rightarrow M' \quad N \mid\!\!\rightarrow N'}{MN \mid\!\!\rightarrow M'N'}$$

$$\frac{M \mid\!\!\rightarrow M'}{\mathrm{fix}(M) \mid\!\!\rightarrow \mathrm{fix}(M')} \qquad \frac{\forall i \quad M_i \mid\!\!\rightarrow M_i'}{(c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n) \mid\!\!\rightarrow (c_1\vec{x_1}.M_1', \ldots, c_n\vec{x_n}.M_n')}$$

$$\frac{M \mid\!\!\rightarrow M' \quad N \mid\!\!\rightarrow N'}{(\lambda x.M)N \mid\!\!\rightarrow M'\{x := N'\}} \qquad \frac{M \mid\!\!\rightarrow M' \quad N \mid\!\!\rightarrow N'}{\mathrm{fix}(M)N \mid\!\!\rightarrow M'N'\mathrm{fix}(M')}$$

$$\frac{\vec{N} = N_1 \ldots N_m \quad \vec{N'} = N_1' \ldots N_m' \quad m \text{ arity of } c_i \quad M_i \mid\!\!\rightarrow M_i' \quad \forall j, N_j \mid\!\!\rightarrow N_j'}{(c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n)(c_i\vec{N}) \mid\!\!\rightarrow M_i'\{\vec{x_i} := \vec{N'}\}}$$

Figure 5.5: Parallel reductions

Lemma 44 (Properties of $\mid\!\!\rightarrow$)**.**

1. *For every M, $M \mid\!\!\rightarrow M$.*
2. *If $M \rightarrow_\alpha M'$ then $M \mid\!\!\rightarrow M'$.*
3. *If $M \mid\!\!\rightarrow M'$ then $M \longrightarrow^*_\alpha M'$.*
4. *If $M \mid\!\!\rightarrow M'$ and $N \mid\!\!\rightarrow N'$, then $M\{x := N\} \mid\!\!\rightarrow M'\{x := N'\}$*

Proof. 1. By induction on M.

2. By induction on $M \rightarrow_\alpha M$.
3. By induction on $M \mid\!\!\rightarrow M$.
4. By induction on M.

□

Definition 46 (Definition of M^*)**.**

Assume M is a term. We define the term M^* by induction on M as follows:

x^*	:=	x	
c^*	:=	x	
$(\lambda x.M)^*$	:=	$\lambda x.M^*$	
$fix(M)^*$	:=	$fix(M^*)$	
$(c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n)^*$	:=	$(c_1\vec{x_1}.{M_1}^*, \ldots, c_n\vec{x_n}.{M_2}^*)$	
$((\lambda x.M)N)^*$	:=	$M^*\{x := N^*\}$	
$(fix(M)N)^*$	:=	$M^*N^*fix(M^*)$	
$(c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n)(c_i\vec{N})$	:=	${M_i}^*\{\vec{x_i} := \vec{N^*}\}$	*with good arity*
$(MN)^*$	:=	M^*N^*	*in all other cases*

Lemma 45 (Propery of M^*)**.**

If $M \mid\!\!\rightarrow M'$ then $M' \mid\!\!\rightarrow M^*$.

Proof. By induction on M.

□

Corollary 3. *Theorem 23 is true.*

Proof. By Lemma 45, $\mid\!\!\rightarrow$ has the diamond property: For every M, M_1 and M_2, if $M \mid\!\!\rightarrow M_1$ and $M \mid\!\!\rightarrow M_2$, then there exists M_3 such that $M_1 \mid\!\!\rightarrow M_3$ and $M_2 \mid\!\!\rightarrow M_3$ (we choose $M_3 = M^*$). Hence, $\mid\!\!\rightarrow$ is confluent. By Lemmas 44.2 and 44.3, $\rightarrow_\alpha$ is confluent.

□

5.6 Example in Caml

To illustrate the fixpoint operator presented in this chapter in a real programming languages, we have to choose a functional language with strict evaluation (which Haskell, for example, is not). Here we choose OCaml.

Assume we have the following code:

```
let rec fix1 f x = f (fix1 f) x

let rec fix2 f x = f x (fix2 f)

let eif b x y = if b then x else y

let rec fact1 x = if x = 0 then 1 else x * fact1 (x - 1)

let fact2 = fix1 (fun f x -> if x = 0 then 1 else x * f (x - 1))

let fact3 =
  fix2 (fun x -> if x = 0 then fun f -> 1 else fun f -> x * f (x - 1))

let rec fact4 x = eif (x = 0) 1 (x * fact4 (x - 1))

let fact5 = fix1 (fun f x -> eif (x = 0) 1 (x * f (x - 1)))

let fact6 = fix2 (fun x -> eif (x = 0) (fun f -> 1) (fun f -> x * f (x - 1)))
```

This gives various implementation of the factorial function. Each implementation has been made with the following choices:

- How the recursion is done: Either with the usual way of coding recursive functions, or with the usual fixpoint operator or with the fixpoint operator presented in this chapter (and with the coding style that go within).
- How branching is done: Either with the usual way of branching (if ... then ... else ...) which is lazy or with a branching *eif* that is strict (it always calculates the two possibilities).

With the usual branching it always terminates on a positive argument. However, with a strict branching, only the implementation that uses the fixpoint operator of this chapter terminates on a positive argument. This is a way to illustrate strong normalisation.

Chapter 6

Conclusion

In the previous chapters, we have studied systems of non-idempotent intersection types and their applications:

- In Chapter 2, we have introduced the simplest version of the typing system of intersection types that we use. We have proved that a simpler measure on the typing trees gives a bound on the longest β-reduction sequence and that it characterizes strong normalisation. Moreover we have used it to define a denotational semantics and we have showed that we can use it to prove strong normalisation in other typing systems.
- In Chapter 3, we have adapted the typing system of Chapter 2 to a calculus with explicit substitutions. By the fact that we have a better control on the erasure of terms, we have a more precise result on the bound of size of the longest reduction sequence.
- In Chapter 4, we have refined the types of Chapter 2, this gives us a more refined result on the bound of the size of the longest β-reduction sequence while still remaining in the pure λ-calculus. Moreover, the typing of a term can give information about the normal form.
- In Chapter 5, we have adapted the typing system of Chapter 2 to a richer calculus with constructors, matching and fixpoints.

We have proved that, by removing idempotency of the intersection in the intersection types, a typing tree gives more quantitative information about the typed λ-term. We could wonder, what happens when we remove other properties. For example, by removing the commutativity of the intersection (the type $A \cap B$ is no longer equivalent to $B \cap A$), we could describes the execution of side effects in a λ-calculus with references:

- $M : A \cap B$ means that M will do an action described by the type A, then it will do the action described by the type B.
- $M : B \cap A$ means that M will do the action described by B and then, it will do the action described by B.

This could be a lead for a further work.

Bibliography

[Abr93] S. Abramsky. Computational interpretations of linear logic. *Theoret. Comput. Sci.*, 111:3–57, 1993. 12

[BBdH93] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In J. F. G. Groote and M. Bezem, editors, *Proc. of the 1st Int. Conf. on Typed Lambda Calculus and Applications*, volume 664 of *LNCS*, pages 75–90. Springer-Verlag, 1993. 12

[BCDC83] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. of Symbolic Logic*, 48(4):931–940, 1983. 46

[BEM10] A. Bucciarelli, T. Ehrhard, and G. Manzonetto. Categorical models for simply typed resource calculi. *ENTCS*, 265:213–230, 2010. 12

[Ber09] A. Bernadet. Fixpoints and strong normalisation. Internship report, ENS Cachan, Chalmers University, 2009. Available at http://lix.polytechnique.fr/ bernadet/Publications.htm 61, 68

[BL11a] A. Bernadet and S. Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In M. Hofmann, editor, *Proc. of the 14th Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS'11)*, volume 6604 of *LNCS*. Springer-Verlag, 2011. 46, 47, 55, 58, 59

[BL11b] A. Bernadet and S. Lengrand. Filter models: non-idempotent intersection types, orthogonality and polymorphism. In M. Bezem, editor, *Proc. of the 20th Annual Conf. of the European Association for Computer Science Logic (CSL'11)*, LIPIcs. Schloss Dagstuhl LCI, 2011. 46, 59, 61, 68

[BL13] A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013. 44, 46, 59

[BM03] P. Baillot and V. Mogbil. Soft lambda-calculus: a language for polynomial time computation. *CoRR*, cs.LO/0312015, 2003. 12

[Böh68] C. Böhm. Alcune proprietà delle forme β-η-normali nel λK-calcolo. Technical report, IAC, Roma, 1968. 28

[CD78] M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archiv für mathematische Logik und Grundlagenforschung*, 19:139–156, 1978. 6, 12

[Chu85] A. Church. *The Calculi of Lambda Conversion. (AM-6) (Annals of Mathematics Studies)*. Princeton University Press, 1985. 7

[CR36] A. Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, 1936. http://www.jstor.org/stable/2268573Electronic Edition. 8

[CS06] T. Coquand and A. Spiwack. A proof of strong normalisation using domain theory. In R. Alur, editor, *21st Annual IEEE Symp. on Logic in Computer Science*, pages 307–316. IEEE Computer Society Press, 2006. 61, 68

[CS07] T. Coquand and A. Spiwack. A proof of strong normalisation using domain theory. *Logic. Methods Comput. Science*, 3(4), 2007.12, 20, 21

[csl07] *Proc. of the 16th Annual Conf. of the European Association for Computer Science Logic (CSL'07)*, volume 4646 of *LNCS*. Springer-Verlag, 2007. 77, 78

[dC05] D. de Carvalho. Intersection types for light affine lambda calculus. *ENTCS*, 136:133–152, 2005. 12

[dC09] D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009. 12, 13

[DCHM00] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama. Compositional characterizations of lambda-terms using intersection types (extended abstract). In *MFCS: Symp. on Mathematical Foundations of Computer Science*, 2000. 12

[ER03] T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theoret. Comput. Sci.*, 309(1-3):1–41, 2003. 12

[Ghi96] S. Ghilezan. Strong normalization and typability with intersection types. *Notre Dame J. Formal Loigc*, 37(1):44–52, 1996. 12

[Gir72] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse d'état, Université Paris 7, 1972. 20

[Gir87] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–101, 1987. 12, 39

[GL02] B. Grégoire and X. Leroy. A compiled implementation of strong reduction. In M. Wand and S. L. P. Jones, editors, *Proc. of the 7th ACM Intern. Conf. on Functional Programming*, pages 235–246. ACM Press, 2002. 61

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989. This is textbook on proof theory and type systems, based on lectures by Girard. It contains an appendix by Lafont on linear logic, and also treats Girard's polymorphic lambda calculus. 6, 7

[GR07] M. Gaboardi and S. R. D. Rocca. A soft type assignment system for *lambda* -calculus. In *Proc. of the 16th Annual Conf. of the European Association for Computer Science Logic (CSL'07)* [csl07], pages 253–267. 12

[Hey71] A. Heyting. *Intuitionism: an introduction.* Studies in logic and the foundations of mathematics. North-Holland Pub. Co., 1971. 6

[How80] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980. Reprint of a manuscript written 1969. 12

[Kes07] D. Kesner. The theory of calculi with explicit substitutions revisited. In *Proc. of the 16th Annual Conf. of the European Association for Computer Science Logic (CSL'07)* [csl07], pages 238–252. 35, 36

[KR11] D. Kesner and F. Renaud. A prismoid framework for languages with resources. *Theoret. Comput. Sci.*, 412(37):4867–4892, 2011. 80

[KvOvR93] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *THEORETICAL COMPUTER SCIENCE*, 121:279–308, 1993. 46

[KW99] A. J. Kfoury and J. B. Wells. Principality and decidable type inference for finite-rank intersection types. In *Proc. of the 26th Annual ACM Symp. on Principles of Programming Languages (POPL'99)*, pages 161–174. ACM Press, 1999. 13

[Laf04] Y. Lafont. Soft linear logic and polynomial time. *Theoret. Comput. Sci.*, 318(1-2):163–180, 2004. 12

[Lei86] D. Leivant. Typing and computational properties of lambda expressions. *Theoret. Comput. Sci.*, 44(1):51–68, 1986. 12

[LS09] F. W. Lawvere and S. H. Schanuel. *Conceptual Mathematics: A First Introduction to Categories.* Cambridge University Press, 2nd edition, 2009. 6

[NM04] P. M. Neergaard and H. G. Mairson. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In C. Okasaki and K. Fisher, editors, *Proc. of the 9th ACM Intern. Conf. on Functional Programming*, pages 138–149. ACM Press, 2004. 13, 58

[Pit03] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Inf. Comput.*, 186(2):165–193, 2003. 9

[Ren11] F. Renaud. *Les ressources explicites vues par la théorie de la réécriture.* PhD thesis, Université Paris 7, 2011. 35, 36

[Sal10] S. Salvati. On the Membership Problem for Non-Linear Abstract Categorial Grammars. *J. of Logic, Language, and Information*, 19(2):163–183, 2010. 57

[Sco82a] D. S. Scott. Domains for denotational semantics. In M. Nielsen and E. M. Schmidt, editors, *ICALP*, volume 140 of *Lecture Notes in Computer Science*, pages 577–613. Springer, 1982. 8

[Sco82b] D. S. Scott. Domains for denotational semantics. In M. Nielsen and E. M. Schmidt, editors, *Proc. of the 9th Intern. Col. on Automata, Languages and Programming (ICALP)*, volume 140 of *LNCS*, pages 577–613. Springer-Verlag, 1982. 45

[Tai75] W. W. Tait. A realizability interpretation of the theory of species. In *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 240–251. Springer-Verlag, 1975. 20

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936. 6, 8

[vB95] S. van Bakel. Intersection Type Assignment Systems. *Theoret. Comput. Sci.*, 151(2):385–435, 1995. 14

[vH02] J. van Heijenoort. *From Frege to Gödel : A Source Book in Mathematical Logic, 1879-1931 (Source Books in the History of the Sciences)*. Harvard University Press, 2002. 6

[vRSSX99] F. van Raamsdonk, P. Severi, M. H. B. Sørensen, and H. Xi. Perpetual reductions in λ-calculus. *Inform. and Comput.*, 149(2):173–225, 1999. 47

[Wel96] J. B. Wells. Typability and type checking in the second-order lambda-calculus are equivalent and undecidable. In *In Proceedings of the Ninth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 176–185. Society Press, 1996. 6

Appendix A

Full proofs

Lemma 15. *$\to_{S,W}$ terminates.*

Proof. By a polynomial argument.

We define $m_x(M)$ as follow: if $x \notin fv(M)$, then $m_x(M) = 1$. Otherwise we have:

$$\begin{array}{lcll}
m_x(x) & = & 1 & \\
m_x(\lambda y.M) & = & m_x(M) & \\
m_x(M_1M_2) & = & m_x(M_1) + m_x(M_2) & x \in fv(M_1), x \in fv(M_2) \\
m_x(M_1M_2) & = & m_x(M_1) & x \notin fv(M_2) \\
m_x(M_1M_2) & = & m_x(M_2) & x \notin fv(M_1) \\
m_x(M[y := N]) & = & m_x(M) + m_y(M) \times (m_x(N) + 1) & x \in fv(M), x \in fv(N) \\
m_x(M[y := N]) & = & m_y(M) \times (m_x(N) + 1) & x \notin fv(M), x \in fv(N) \\
m_x(M[y := N]) & = & m_x(M) & x \notin fv(N)
\end{array}$$

We also define $S(M)$ as follow:

$$\begin{array}{lcl}
S(x) & = & 1 \\
S(M_1M_2) & = & S(M_1) + S(M_2) \\
S(\lambda x.M) & = & S(M) \\
S(M[x := N]) & = & S(M) + m_x(M) \times S(N)
\end{array}$$

Finally, we define $I(M)$ as follow:

$$\begin{array}{lcl}
I(x) & = & 2 \\
I(\lambda x.M) & = & 2I(M) + 2 \\
I(M_1M_2) & = & 2I(M_1) + 2I(M_2) + 2 \\
I(M[x := N]) & = & I(M) \times (I(N) + 1)
\end{array}$$

If we consider $n = (S(M), I(M))$ in lexical order, then $\to_{S,W}$ strictly decreases n and $\equiv$ does not change it.

Hence $\to_{S,W}$ terminates. This lemma and proof are a special case of [KR11]. □

Lemma 17 (Typing of explicit substitution)**.**
Assume $\Gamma, x : A \vdash^n M : B$ and $\Delta \vdash^m N : A$. Then, there exists Γ' such that $\Gamma' \approx \Gamma \cap \Delta$ and $\Gamma' \vdash^{n+m} M[x := N] : B$.

Proof. By induction on B:

- If $B = F$, then the result is trivial : we use the (Subst) rule.
- If $B = B_1 \cap B_2$, then, by Lemma 16.1, there exist Γ_1, Γ_2, A_1, A_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $A = A_1 \cap A_2$, $n = n_1 + n_2$, $\Gamma_1, x : A_1 \vdash^{n_1} M : B_1$ and $\Gamma_2, x : A_2 \vdash^{n_2} M : B_2$. By hypothesis, $\Delta \vdash^m N : A$. Hence, by Lemma 16.1,

there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : A_1$ and $\Delta_2 \vdash^{m_2} N : A_2$.

By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M[x := N] : B_1$ and $\Gamma'_2 \vdash^{n_2+m_2} M[x := N] : B_2$. So we have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2} M[x := N] : B_1 \cap B_2$ with $n_1+m_1+n_2+m_2 = n+m$, $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) \approx \Gamma \cap \Delta$ and $B = B_1 \cap B_2$.

□

Theorem 8 (Subject Reduction for λS)**.**
Assume $\Gamma \vdash^n M : A$. We have the following properties:

1. *If $M \to_B M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m < n$ and $\Gamma' \vdash^m M' : A$*
2. *If $M \to_S M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n M' : A$*
3. *If $M \to_W M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m \leq n$ and $\Gamma' \vdash^m M' : A$*
4. *If $M \equiv M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n M' : A$*

Proof. First by induction on $M \to_E M'$ and $M \equiv M'$, then by induction on A.

For modularity, the triplet $(\to_E, R, r)$ can be one of the following triplets: $(\to_B, \approx, <)$, $(\to_S, \approx, =)$, $(\to_W, \subseteq, \leq)$, $(\equiv, \approx, =)$.

- If $A = A_1 \cap A_2$, then there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} M : A_1$ and $\Gamma_2 \vdash^{n_2} M : A_2$.

  By induction hypothesis (on $(M \to_E M', A_1)$ and $(M \to_E M', A_2)$), there exist Γ'_1, Γ'_2, m_1 and m_2 such that $\Gamma_1 \ R \ \Gamma'_1$, $\Gamma_2 \ R \ \Gamma'_2$, $m_1 \ r \ n_1$, $m_2 \ r \ n_2$, $\Gamma'_1 \vdash^{m_1} M' : A_1$ and $\Gamma'_2 \vdash^{m_2} M' : A_2$.

  Hence, $\Gamma'_1 \cap \Gamma'_2 \vdash^{m_1+m_2} M' : A_1 \cap A_2$ with $A = A_1 \cap A_2$, $m_1 + m_2 \ r \ n$, $\Gamma \ R \ \Gamma'_1 \cap \Gamma'_2$.

- $(\lambda x.M_1)M_2 \to_B M_1[x := M_2]$ and $A = F$: There exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} \lambda x.M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$. Hence, there exists U such that $B \subseteq U$ and $\Gamma_1, x : U \vdash^{n_1} M_1 : F$.
  - If $U = C$, then by using Lemma 16.4 there exist Δ and m such that $m \leq n_2$, $\Gamma_2 \subseteq \Delta$ and $\Delta \vdash^m M_2 : C$. Hence $\Gamma_1 \cap \Delta \vdash^{n_1+m} M_1[x := M_2] : F$ with $n_1 + m < n$, $F = A$ and $\Gamma \subseteq \Gamma_1 \cap \Delta$.
  - If $U = \omega$, then $\Gamma_1 \cap \Gamma_2 \vdash^{n_1+n_2} M_1[x := M_2] : F$ with $n_1 + n_2 < n$, $A = F$ and $\Gamma \subseteq \Gamma_1 \cap \Gamma_2$.

- $y[x := N] \to_W y$, $x \neq y$ and $A = F$: there exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} N : B$ and $\Gamma_2, x : \omega \vdash^{n_2} y : F$. So we have $\Gamma_2 \vdash^{n_2} y : A$ with $\Gamma \subseteq (\Gamma_2, x : \omega)$ and $n_2 \leq n$.

- $x[x := N] \to_S N$ and $A = F$: there exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} N : B$ and $\Gamma_2, x : B \vdash^{n_2} x : F$. Hence $\Gamma_2 = ()$ and $B = F$ and $n_2 = 0$. Therefore $\Gamma = \Gamma_1$ and $n_1 = n$. So we have $\Gamma \vdash^n N : A$ with $\Gamma \approx \Gamma$.

- $(M_1M_2)[x := N] \to_S M_1[x := N]M_2[x := N]$ with $x \in fv(M_1)$, $x \in fv(M_2)$ and $A = F$: Then there exist Γ_1, Γ_2, Γ_3, Γ_4, A_1, A_2, B, n_1, n_2, n_3 and n_4 such that: $\Gamma_1, x : A_1 \vdash^{n_1} M_1 : B \to F$ $\Gamma_2, x : A_2 \vdash^{n_2} M_2 : B$, $\Gamma_3 \vdash^{n_3} N : A_1$, $\Gamma_4 \vdash^{n_4} N : A_2$, $n = n_1 + n_2 + n_3 + n_4$ and $\Gamma = (\Gamma_1 \cap \Gamma_2) \cap (\Gamma_3 \cap \Gamma_4)$. Hence $(\Gamma_1 \cap \Gamma_3) \cap (\Gamma_2 \cap \Gamma_4) \vdash^{n_1+n_3+n_2+n_4} M_1[x := N]M_2[x := N] : F$.

- $(M_1M_2)[x := N] \to_S M_1[x := N]M_2$ with $x \notin fv(M_2)$ and $A = F$: Then there exist Γ_1, Γ_2, Γ_3, U, A_1, B, n_1, n_2, and n_3 such that: $\Gamma_1, x : U \vdash^{n_1} M_1 : B \to F$ $\Gamma_2, x : \omega \vdash^{n_2} M_2 : B$, $\Gamma_3 \vdash^{n_3} N : A_1$, $n = n_1 + n_2 + n_3$, $\Gamma = (\Gamma_1 \cap \Gamma_2) \cap \Gamma_3$, $U = A_1$ or $U = \omega$. Hence $(\Gamma_1 \cap \Gamma_3) \cap \Gamma_2 \vdash^{n_1+n_3+n_2} M_1[x := N]M_2 : F$.
- $(M_1M_2)[x := N] \to_S M_1M_2[x := N]$ with $x \notin fv(M_1)$, $x \in fv(M_2)$ and $A = F$: Then there exist Γ_1, Γ_2, Γ_3, A_1, B, n_1, n_2 and n_3 such that $\Gamma_1, x : \omega \vdash^{n_1} M_1 : B \to F$, $\Gamma_2, x : A_1 \vdash^{n_2} M_2 : F$, $\Gamma_3 \vdash^{n_3} N : A_1$, $n = n_1 + n_2 + n_3$, $\Gamma = (\Gamma_1 \cap \Gamma_2) \cap \Gamma_3$, Hence $\Gamma_1 \cap (\Gamma_2 \cap \Gamma_3) \vdash^{n_1+n_2+n_3} M_1M_2[x := N] : F$.
- For $M[x := N_1][y := N_2] \equiv M[y := N_2][x := N_1]$ with $x \neq y$, $x \notin fv(N_2)$, $y \notin fv(N_1)$ and $A = F$: There exist Γ_1, Γ_2, n_1, n_2, U and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = B$ or $U = \omega$, $\Gamma_1, y : U \vdash^{n_1} M[x := N_1] : F$ and $\Gamma_2 \vdash^{n_2} N_2 : B$. Therefore, there exist Γ_3, Γ_4, n_3, n_4, V and C such that $\Gamma_1, y : U = \Gamma_3 \cap \Gamma_4$, $n_1 = n_3 + n_4$, $V = C$ or $V = \omega$, $\Gamma_3, x : V \vdash^{n_3} M : F$ and $\Gamma_4 \vdash^{n_4} N_1 : C$. By the fact that $y \notin fv(N_1)$ and by Lemma 16.2, we have $\Gamma_4 = \Gamma_4, x : \omega$. Hence, there exists Γ_5 such that $\Gamma_3 = \Gamma_5, x : U$ and $\Gamma_1 = \Gamma_5 \cap \Gamma_4$. So, $\Gamma_5, y : U, x : V \vdash^{n_3} M : F$. Hence, $(\Gamma_5, x : V) \cap \Gamma_2 \vdash^{n_3+n_2} M[y := N_2] : F$. By the fact that $x \notin fv(N_2)$ and by Lemma 16, $\Gamma_2 = \Gamma_2, x : \omega$. Hence, $(\Gamma_5, x : V) \cap \Gamma_2 = \Gamma_5 \cap \Gamma_2, x : V$. Therefore, $(\Gamma_5 \cap \Gamma_2)\Gamma_4 \vdash^{n_3+n_2+n_4} M[y := N_2][x := N_1] : F$ with $n_3 + n_2 + n_4 = n$ and $(\Gamma_5 \cap \Gamma_2) \cap \Gamma_4 \approx \Gamma$.
- The other rules follow the same patterns, especially for the propagation of an explicit substitution over another explicit substitution. Now concerning the congruent closure of the rules, all cases are straightforward but for the following one:
- $M[x := N] \to_W M'[x := N]$ with $M \to_W M'$, $x \in fv(M)$, $x \in fv(M')$ and $A = F$: Then there exist Γ_1, Γ_2, B, n_1, n_2 such that: $\Gamma_1, x : B \vdash^{n_1} M : F$ and $\Gamma_2 \vdash^{n_2} N : B$, $n = n_1 + n_2$ and $\Gamma = \Gamma_1 \cap \Gamma_2$. By induction hypothesis, there exist Γ_1', C and n_1' such that $\Gamma_1 \subseteq \Gamma_1'$, $B \subseteq C$, $n_1' \leq n_1$ and $\Gamma_1', x : C \vdash^{n_1'} M' : A$. Then there exist Γ_2' and n_2' such that $\Gamma_2 \subseteq \Gamma_2'$, $n_2' \leq n_2$ and $\Gamma_2' \vdash^{n_2'} N : C$. Hence $\Gamma_1' \cap \Gamma_2' \vdash^{n_1'+n_2'} M'[x := N] : F$ with $n_1' + n_2' \leq n$ and $\Gamma \subseteq \Gamma_1' \cap \Gamma_2'$.

□

Lemma 20 (Most inefficient reduction)**.**
Assume $\Gamma \vdash^n_{opt} M : A$. If M can be reduced by $\to_B$ and not by $\to_S$, then there exist M' and Γ' such that $\Gamma \approx \Gamma'$, $M \to_B M'$ and $\Gamma' \vdash^{n-1}_{opt} M' : A$.

Lemma 22 (Shape of a normal form)**.**
If M cannot be reduced by $\to_\beta$, then

- *either we have $acc(M)$,*
- *or M is of the form $\lambda x.M_1$.*

Proof. By induction on M.

- If M is of the form $\lambda x.M_1$, then we can conclude.
- If M is a variable x, then we have $\mathrm{acc}(M)$.
- If M is of the form M_1M_2: Then, M_1 cannot be reduced by $\to_\beta$. By induction hypothesis on M_1, either we have $\mathrm{acc}(M_1)$ or M_1 is of the form $\lambda x.M_3$.
  - If $\mathrm{acc}(M_1)$, then $\mathrm{acc}(M_1M_2)$.
  - If M_1 is of the form $\lambda x.M_3$, then $M \to_\beta M_3\{x := M_2\}$. Contradiction.

□

Lemma 23 (Applicability of $\Rightarrow_h$)**.** *If M can be reduced by $\to_\beta$, then M can be reduced by $\Rightarrow_h$.*

Proof. We prove by induction on M that if M can be reduced by $\to_\beta$ then:

- If M is of the form $\lambda x.M_1$, then there exists M' such that $M \Rightarrow_h M'$.
- If not, then there exists M' such $M \rightsquigarrow_h M'$.

Therefore, in both cases there exists M' such that $M \Rightarrow_h M'$.

- If M is a variable, then M cannot be reduced by $\to_\beta$. Contradiction.
- If M is of the form $\lambda x.M_1$: Then, M_1 can be reduced by $\to_\beta$. By induction hypothesis, there exists M_1' such that $M_1 \Rightarrow_h M_1'$. Therefore, $M \Rightarrow_h \lambda x.M_1'$.
- If M is of the form M_1M_2: Therefore, we are in one of the following cases:
  - M_1 is of the form $\lambda x.M_3$ and $x \in \mathrm{fv}(M_3)$: Therefore, $M \rightsquigarrow_h M_3\{x := M_2\}$.
  - M_1 is of the form $\lambda x.M_3$, $x \notin \mathrm{fv}(M_3)$ and M_2 can be reduced by $\to_\beta$: By induction hypothesis, there exist M_2' such that $M_2 \Rightarrow_h M_2'$. Therefore $M \rightsquigarrow_h (\lambda x.M_3)M_2'$.
  - M_1 is of the form $\lambda x.M_3$, $x \notin \mathrm{fv}(M_3)$ and M_2 cannot be reduced by $\to_\beta$: Then, $M \rightsquigarrow_h M_3$.
  - M_1 is not of the form $\lambda x.M_3$ and M_1 can be reduced by $\to_\beta$: By induction hypothesis, there exist M_1' such that $M_1 \rightsquigarrow_h M_1'$. Therefore, $M \rightsquigarrow_h M_1'M_2$.
  - M_1 is not of the form $\lambda x.M_3$ and M_1 cannot be reduced by $\to_\beta$: By the fact that M_1 is not of the form $\lambda x.M_3$, M_2 can be reduced by $\to_\beta$. By induction hypothesis, there exist M_2' such that $M_2 \Rightarrow_h M_2'$. By Lemma 22, we have $\mathrm{acc}(M_1)$. Therefore, $M \rightsquigarrow_h M_1M_2'$.

□

Lemma 24 (Structure of a normal term)**.** *M cannot be reduced by $\to_\beta$ if and only if $\mathrm{struct}(M)$ is well-defined. Moreover, if we have $\mathrm{acc}(M)$, then $\mathrm{struct}(M)$ is of the form k.*

Proof. By induction on M.

- If M is of the form x: x cannot be reduced by $\to_\beta$, $\mathrm{struct}(x)$ is well-defined and $\mathrm{struct}(x) = \triangledown$ which is of the form k.
- If M is of the form $\lambda x.M_1$:
  - If $\lambda x.M_1$ cannot be reduced by $\to_\beta$: Then, M_1 cannot be reduced by $\to_\beta$. By induction hypothesis, $\mathrm{struct}(M_1)$ is well-defined. Therefore, $\mathrm{struct}(\lambda x.M_1)$ is well-defined and we do not have $\mathrm{acc}(\lambda x.M_1)$.
  - If $\mathrm{struct}(\lambda x.M_1)$ is well-defined: Then, $\mathrm{struct}(M_1)$ is well-defined. By induction hypothesis, M_1 cannot be reduced by $\to_\beta$. Therefore, $\lambda x.M_1$ cannot be reduced by $\to_\beta$.
- If M is of the form M_1M_2:
  - If M_1M_2 cannot be reduced by $\to_\beta$: Then, M_1 and M_2 cannot be reduced by $\to_\beta$. By induction hypothesis, $\mathrm{struct}(M_1)$ and $\mathrm{struct}(M_2)$ are well-defined. By Lemma 22, we have $\mathrm{acc}(M_1)$ or M_1 is of the form $\lambda x.M_3$. If M_1 is of the form $\lambda x.M_3$, then $M_1M_2 \to_\beta M_3\{x := M_2\}$. Contradiction. Hence, we have $\mathrm{acc}(M_1)$. By induction hypothesis, $\mathrm{struct}(M_1)$ is of the form k. Therefore, $\mathrm{struct}(M_1M_2)$ is well-defined and $\mathrm{struct}(M_1M_2) = \mathrm{struct}(M_1)\mathrm{struct}(M_2)$ which is of the form k'.

– If $\mathrm{struct}(M_1M_2)$ is well-defined: Then, $\mathrm{struct}(M_1)$ and $\mathrm{struct}(M_2)$ are well-defined, and $\mathrm{struct}(M_1)$ is of the form k. By induction hypothesis, M_1 and M_2 cannot be reduced by $\to_\beta$. If M_1 is of the form $\lambda x.M_3$, then $\mathrm{struct}(M_1) = \lambda\mathrm{struct}(M_3)$ which is not of the form k. Contradiction. Hence, M_1 is not of the form $\lambda x.M_3$. Therefore, M_1M_2 cannot be reduced by $\to_\beta$.

□

Lemma 29 (Substitution lemma)**.**
If $\Gamma, x : U \vdash^n M : A$ *and* $\Delta \vdash^m N : U$*, then there exists* Γ' *such that* $\Gamma' \approx \Gamma \cap \Delta$ *and* $\Gamma' \vdash^{n+m} M\{x := N\} : A$*.*

Proof. By induction on $\Gamma, x : U \vdash^n M : A$.

- For $\dfrac{}{x : F \vdash^0 x : F}$ with $\Gamma = ()$, $n = 0$, $M = x$, $U = F$ and $A = F$: We have $x\{x := N\} = N$. By hypothesis, $\Delta \vdash^m N : U$. Therefore, $\Delta \vdash^m M\{x := N\} : F$ with $n + m = m$ and $\Gamma \cap \Delta = () \cap \Delta = \Delta$.
- For $\dfrac{}{y : F \vdash^0 y : F}$ with $y \neq x$, $\Gamma = (y : F)$, $n = 0$, $M = y$, $U = \omega$, and $A = F$: By hypothesis, $\Delta \vdash^m N : \omega$. Hence, $\Delta = ()$ and $m = 0$. We have $y\{x := N\} = y$. Therefore, $y : F \vdash^0 M\{x := N\} : F$ with $n + m = 0$ and $\Gamma \cap \Delta = (y : F) \cap () = (y : F)$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M : A_1 \quad \Gamma_2, x : U_2 \vdash^{n_2} M : A_2}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2} M : A_1 \cap A_2}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$ and $A = A_1 \cap A_2$: By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$. By Lemma 28.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ_1' and Γ_2' such that $\Gamma_1' \approx \Gamma_1 \cap \Delta_1$, $\Gamma_2' \approx \Gamma_2 \cap \Delta_2$, $\Gamma_1' \vdash^{n_1+m_1} M\{x := N\} : A_1$ and $\Gamma_2' \vdash^{n_2+m_2} M\{x := N\} : A_2$. Therefore, $\Gamma_1' \cap \Gamma_2' \vdash^{n_1+m_1+n_2+m_2} M\{x := N\} : A_1 \cap A_2$ with $\Gamma_1' \cap \Gamma_2' \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 = n + m$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n M_1 : F \quad B \subseteq V}{\Gamma, x : U \vdash^n \lambda y.M_1 : B \to F}$ with $M = \lambda y.M_1$, $x \neq y$, $y \notin \mathrm{fv}(N)$ and $A = B \to F$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exists Γ' such that $\Gamma' \approx (\Gamma, y : V) \cap \Delta$ and $\Gamma' \vdash^{n+m} M_1\{y := N\} : F$. By Lemma 28.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V \approx V'$. Hence, $B \subseteq V'$. Therefore, $\Gamma'' \vdash^{n+m} \lambda y.M_1\{x := N\} : B \to F$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n M_1 : [v] \quad \mathrm{input}(V)}{\Gamma, x : U \vdash^n \lambda y.M_1 : [\lambda v]}$ with $M = \lambda y.M_1$, $y \notin \mathrm{fv}(N)$, $A = [\lambda v]$ and $y \neq x$: We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exists Γ' such that $\Gamma' \approx (\Gamma, y : V) \cap \Delta$ and $\Gamma' \vdash^{n+m} M_1\{x := N\} : [v]$. By Lemma 28.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V \approx V'$. By Lemma 25.10, we have $\mathrm{input}(V')$. Therefore, $\Gamma'' \vdash^{n+m} \lambda y.M_1\{x := N\} : [\lambda v]$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M_1 : B \to F \quad \Gamma_2, x : U_2 \vdash^{n_2} M_2 : B}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2+1} M_1M_2 : F}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $U = U_1 \cap U_2$, $M = M_1M_2$ and $A = F$: We have $(M_1M_2)\{x := N\} = M_1\{x := N\}M_2\{x := N\}$. By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$.

By Lemma 28.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M_1\{x := N\} : B \to F$ and $\Gamma'_2 \vdash^{n_2+m_2} M_2\{x := N\} : B$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2+1} M_1M_2 : F$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 + 1 = n + m$.

- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M_1 : [k] \quad \Gamma_2, x : U_2 \vdash^{n_2} M_2 : [v]}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2} M_1M_2 : [kv]}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$, $M = M_1M_2$ and $A = [kv]$: We have $(M_1M_2)\{x := N\} = M_1\{x := N\}M_2\{x := N\}$. By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$. By Lemma 28.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M_1\{x := N\} : [k]$ and $\Gamma'_2 \vdash^{n_2+m_2} M_2\{x := N\} : [v]$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2} M_1M_2 : [kv]$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 = n + m$.

□

Theorem 14 (Subject Reduction)**.**
If $\Gamma \vdash^n M : A$ and $M \to_\beta M'$, then there exist Γ' and n' such that $\Gamma \subseteq \Gamma'$, $n > n'$ and $\Gamma' \vdash^{n'} M' : A$.

Proof. First by induction on $M \to_\beta M'$, then by induction on A.

- If A is of the form $A_1 \cap A_2$: Then, there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} M : A_1$ and $\Gamma_2 \vdash^{n_2} M : A_2$. By induction hypothesis on $(M \to_\beta M', A_1)$ and $(M \to_\beta M', A_2)$, there exist Γ'_1, Γ'_2, n'_1 and n'_2 such that $\Gamma_1 \subseteq \Gamma'_1$, $\Gamma_2 \subseteq \Gamma'_2$, $n_1 > n'_1$, $n_2 > n'_2$, $\Gamma'_1 \vdash^{n'_1} M' : A_1$ and $\Gamma'_2 \vdash^{n'_2} M' : A_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n'_1+n'_2} M' : A_1 \cap A_2$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma'_2$ and $n = n_1 + n_2 > n'_1 + n'_2$.
- For $\dfrac{}{(\lambda x.M_1)M_2 \to_\beta M_1\{x := M_2\}}$ with $M = (\lambda x.M_1)M_2$ and A is of the form F:

  Then, we are in one of the following cases:
  - There exist Γ_1, Γ_2, v and k such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash \lambda x.M_1 : [k]$, $\Gamma_2 \vdash M_2 : [v]$ and $F = [kv]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $B \to G$ or of the form $[\lambda v_1]$). Contradiction.
  - There exist Γ_1, Γ_2, n_1, n_2, B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} \lambda x.M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$: Then, there exists U such that $B \subseteq U$ and $\Gamma_1, x : U \vdash^{n_1} M_1 : F$. By Lemma 28.3, there exist Γ'_2 and n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n_2 \geq n'_2$ and $\Gamma'_2 \vdash^{n'_2} M_2 : U$. By Lemma 29, there exists Γ' such that $\Gamma' \approx \Gamma_1 \cap \Gamma'_2$ and $\Gamma' \vdash^{n_1+n'_2} M_1\{x := M_2\} : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2 \approx \Gamma'$ and $n = n_1 + n_2 + 1 > n_1 + n_2 \geq n_1 + n'_2$.
- For $\dfrac{M_1 \to_\beta M'_1}{\lambda x.M_1 \to_\beta \lambda x.M'_1}$ with $M = \lambda x.M_1$ and A is of the form F, we are in one of the following cases:
  - There exist B, G and U such that $F = B \to G$, $B \subseteq U$ and $\Gamma, x : U \vdash^n M_1 : G$. By induction hypothesis, there exists Γ'_1 and n' such that $(\Gamma, x : U) \subseteq \Gamma'_1$, $n > n'$ and $\Gamma'_1 \vdash^{n'} M'_1 : G$. There exist a unique Γ' and a unique U' such that $\Gamma'_1 = (\Gamma', x : U')$. Therefore, $\Gamma \subseteq \Gamma'$ and $U \subseteq U'$. Hence, $B \subseteq U'$. Therefore $\Gamma' \vdash^{n'} \lambda x.M'_1 : B \to G$.

– There exist U and v such that input(U), $F = [\lambda v]$ and $\Gamma, x : U \vdash^{n} M_1 : [v]$. By induction hypothesis, there exist Γ'_1 and n' such that $(\Gamma, x : U) \subseteq \Gamma'_1$, $n > n'$ and $\Gamma'_1 \vdash^{n'} M'_1 : [v]$. There exist a unique Γ' and a unique U' such that $\Gamma'_1 = (\Gamma', x : U')$. Therefore, $\Gamma \subseteq \Gamma'$ and $U \subseteq U'$. Hence, by Lemma 26.5, we have input(U'). Therefore, $\Gamma' \vdash^{n'} \lambda x.M'_1 : [\lambda v]$.

- For $\dfrac{M_1 \to_\beta M'_1}{M_1 M_2 \to_\beta M'_1 M_2}$ with $M = M_1 M_2$ and A is of the form F, we are in one of the following cases:
  – There exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$. By induction hypothesis, there exist Γ'_1 and n'_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $n_1 > n'_1$ and $\Gamma'_1 \vdash^{n'_1} M'_1 : B \to F$. Therefore, $\Gamma'_1 \cap \Gamma_2 \vdash^{n'_1+n_2+1} M'_1 M_2 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma_2$ and $n = n_1 + n_2 + 1 > n'_1 + n_2 + 1$.
  – There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2} M_2 : [v]$. By induction hypothesis, there exist Γ'_1 and n'_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $n_1 > n'_1$ and $\Gamma'_1 \vdash^{n'_1} M'_1 : [k]$. Therefore $\Gamma'_1 \cap \Gamma_2 \vdash^{n'_1+n_2} M'_1 M_2 : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma_2$ and $n = n_1 + n_2 > n'_1 + n_2$.
- For $\dfrac{M_2 \to_\beta M'_2}{M_1 M_2 \to_\beta M_1 M'_2}$ with $M = M_1 M_2$ and A is of the form F, we are in one of the following cases:
  – There exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$. By induction hypothesis, there exist Γ'_2 and n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n_2 > n'_2$ and $\Gamma'_2 \vdash^{n'_2} M'_2 : B$. Therefore, $\Gamma_1 \cap \Gamma'_2 \vdash^{n_1+n'_2+1} M_1 M'_2 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2$ and $n = n_1 + n_2 + 1 > n_1 + n'_2 + 1$.
  – There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2} M_2 : [v]$. By induction hypothesis, there exist Γ'_2 and n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n_2 > n'_2$ and $\Gamma'_2 \vdash^{n'_2} M'_2 : [v]$. Therefore, $\Gamma_1 \cap \Gamma'_2 \vdash^{n_1+n'_2} M_1 M'_2 : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2$ and $n = n_1 + n_2 > n_1 + n'_2$.

□

Lemma 30 (Anti-substitution lemma)**.**
If $\Gamma \vdash_{ns} M\{x := N\} : A$*, then there exist* Γ'*,* Δ *and* U *such that:*

- $\Gamma \approx \Gamma' \cap \Delta$.
- $\Gamma', x : U \vdash_{ns} M : A$ *and* $\Delta \vdash_{ns} N : U$.
- *For all* $y \notin \mathsf{Dom}(\Delta)$, $\Gamma(y) = \Gamma'(y)$.

Proof. First by induction on M, then by induction on A.

- If A is of the form $A_1 \cap A_2$: Then, there exist Γ_1 and Γ_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} M\{x := N\} : A_1$ and $\Gamma_2 \vdash_{\text{ns}} M\{x := N\} : A_2$. By induction hypothesis on (M, A_1) and (M, A_2), there exist Γ'_1, Γ'_2, Δ_1, Δ_2, U_1 and U_2 such that:
  – $\Gamma_1 \approx \Gamma'_1 \cap \Delta_1$ and $\Gamma_2 \approx \Gamma'_2 \cap \Delta_2$.
  – $\Gamma'_1, x : U_1 \vdash_{\text{ns}} M : A_1$, $\Gamma'_2, x : U_2 \vdash_{\text{ns}} M : A_2$, $\Delta_1 \vdash_{\text{ns}} N : U_1$ and $\Delta_2 \vdash_{\text{ns}} N : U_2$.
  – For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.

- For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$, $\Delta = \Delta_1 \cap \Delta_2$ and $U = U_1 \cap U_2$:

- $\Gamma = \Gamma_1 \cap \Gamma_2 \approx (\Gamma'_1 \cap \Delta_1) \cap (\Gamma'_2 \cap \Delta_2) \approx (\Gamma'_1 \cap \Gamma'_2) \cap (\Delta_1 \cap \Delta_2)$.
- We have $\Gamma'_1 \cap \Gamma'_2, x : U_1 \cap U_2 \vdash_{\mathrm{ns}} M : A_1 \cap A_2$ and, by Lemma 28.1, $\Delta_1 \cap \Delta_2 \vdash_{\mathrm{ns}} N : U_1 \cap U_2$.
- Assume $y \notin \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.

- If $M = x$ and A is of the form F: Then, $M\{x := N\} = N$. Therefore, with $\Gamma' = ()$, $\Delta = \Gamma$ and $U = F$:
  - $\Gamma = () \cap \Gamma$.
  - We have $x : F \vdash_{\mathrm{ns}} x : F$ and $\Gamma \vdash_{\mathrm{ns}} N : F$.
  - Assume $y \notin \mathsf{Dom}(\Gamma)$. Therefore, $\Gamma(y) = \omega = ()(y)$.
- If M is of the form y with $y \neq x$ and A if of the form F: We have $M\{x := N\} = y = M$. Hence, $\Gamma = (y : F)$. Therefore, with $\Gamma' = (y : F)$, $\Delta = ()$ and $U = \omega$:
  - $(y : F) = (y : F) \cap ()$.
  - We have $y : F, x : \omega \vdash_{\mathrm{ns}} M : F$ and, by the rule (ω), $\vdash_{\mathrm{ns}} N : \omega$.
  - Assume $z \notin \mathsf{Dom}(())$. Therefore, $\Gamma(z) = (y : F)(z)$.
- If M is of the form $\lambda y.M_1$ with $y \neq x$ and $y \notin \mathrm{fv}(N)$, and A is of the form $B \to F$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. Therefore, there exists V such that $\Gamma, y : V \vdash_{\mathrm{ns}} M_1\{x := N\} : F$ and $V = B$ or $V = \omega$ and $\mathrm{output}(B)$. By induction hypothesis, there exist Γ', Δ and U such that:
  - $(\Gamma, y : V) \approx \Gamma' \cap \Delta$.
  - We have $\Gamma', x : U \vdash_{\mathrm{ns}} M_1 : F$ and $\Delta \vdash_{\mathrm{ns}} N : U$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

  There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. By Lemma 28.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$. Hence, $(\Gamma'', y : V') \cap \Delta = (\Gamma'' \cap \Delta, y : V')$. Therefore, $(\Gamma, y : V) \approx \Gamma' \cap \Delta = (\Gamma'' \cap \Delta, y : V')$. Hence, $\Gamma \approx \Gamma'' \cap \Delta$ and $V \approx V'$. By the fact that $y \notin \mathsf{Dom}(\Delta)$, we have $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:
  - We have $\Gamma \approx \Gamma'' \cap \Delta$.
  - We have $\Gamma'', y : V, x : U \vdash_{\mathrm{ns}} M_1 : F$ and $\Delta \vdash_{\mathrm{ns}} N : U$. Therefore, $\Gamma'', x : U \vdash_{\mathrm{ns}} \lambda y.M_1 : B \to F$.
  - Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma(z) \approx \Gamma''(z) \cap \Delta(z) = \Gamma''(z) \cap \omega = \Gamma''(z)$.

    If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.

    If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.
- If M is of the form $\lambda y.M_1$ with $y \neq x$ and $y \notin \mathrm{fv}(N)$, and A is of the form $[\lambda v]$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. Therefore, there exist V and F such that $\Gamma, y : V \vdash_{\mathrm{ns}} M_1\{x := N\} : [v]$ and $\mathrm{input}(V)$. By induction hypothesis, there exist Γ', Δ and U such that:
  - $(\Gamma, y : V) \approx \Gamma' \cap \Delta$.
  - We have $\Gamma', x : U \vdash_{\mathrm{ns}} M_1 : [v]$ and $\Delta \vdash_{\mathrm{ns}} N : U$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

  There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. By Lemma 28.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$. Hence, $(\Gamma'', y : V') \cap \Delta = (\Gamma'' \cap \Delta, y : V')$. Therefore, $(\Gamma, y : V) \approx \Gamma' \cap \Delta = (\Gamma'' \cap \Delta, y : V')$.

Hence, $\Gamma \approx \Gamma'' \cap \Delta$ and $V \approx V'$. By the fact that $y \notin \mathsf{Dom}(\Delta)$, we have $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:

- We have $\Gamma \approx \Gamma'' \cap \Delta$.
- We have $\Gamma'', y : V, x : U \vdash_{\mathrm{ns}} M_1 : [v]$ and $\Delta \vdash_{\mathrm{ns}} N : U$. Therefore, $\Gamma'', x : U \vdash_{\mathrm{ns}} \lambda y.M_1 : [\lambda v]$.
- Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma(z) \approx \Gamma''(z) \cap \Delta(z) = \Gamma''(z) \cap \omega = \Gamma''(z)$.

  If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.

  If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.

- If M is of the form $M_1 M_2$ and A is of the form F: We have $(M_1 M_2)\{x := N\} = M_1\{x := N\} M_2\{x := N\}$. We are in one of the following cases:
  - There exist Γ_1, Γ_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\mathrm{ns}} M_1\{x := N\} : B \to F$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2\{x := N\} : B$. By induction hypothesis, there exist Γ_1', Γ_2', Δ_1, Δ_2, U_1 and U_2 such that:
    * $\Gamma_1 \approx \Gamma_1' \cap \Delta_1$ and $\Gamma_2 \approx \Gamma_2' \cap \Delta_2$.
    * We have $\Gamma_1', x : U_1 \vdash_{\mathrm{ns}} M_1 : B \to F$, $\Gamma_2', x : U_2 \vdash_{\mathrm{ns}} M_2 : B$, $\Delta_1 \vdash_{\mathrm{ns}} N : U_1$ and $\Delta_2 \vdash_{\mathrm{ns}} N : U_2$.
    * For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma_1'(y)$.
    * For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma_2'(y)$.

    Therefore, with $\Gamma' = \Gamma_1' \cap \Gamma_2'$, $\Delta = \Delta_1 \cap \Delta_2$ and $U = U_1 \cap U_2$:
    * $\Gamma_1 \cap \Gamma_2 \approx (\Gamma_1' \cap \Delta_1) \cap (\Gamma_2' \cap \Delta_2) \approx (\Gamma_1' \cap \Gamma_2') \cap (\Delta_1 \cap \Delta_2)$.
    * We have $\Gamma_1' \cap \Gamma_2', x : U_1 \cap U_2 \vdash_{\mathrm{ns}} M_1 M_2 : F$ and, by Lemma 28.1, $\Delta_1 \cap \Delta_2 \vdash_{\mathrm{ns}} N : U_1 \cap U_2$.
    * Assume $y \notin \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma_1'(y) \cap \Gamma_2'(y) = (\Gamma_1' \cap \Gamma_2')(y)$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\mathrm{ns}} M_1\{x := N\} : [k]$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2\{x := N\} : [v]$. By induction hypothesis, there exist Γ_1', Γ_2', Δ_1, Δ_2, U_1 and U_2 such that:
    * $\Gamma_1 \approx \Gamma_1' \cap \Delta_1$ and $\Gamma_2 \approx \Gamma_2' \cap \Delta_2$.
    * We have $\Gamma_1', x : U_1 \vdash_{\mathrm{ns}} M_1 : [k]$, $\Gamma_2', x : U_2 \vdash_{\mathrm{ns}} M_2 : [v]$, $\Delta_1 \vdash_{\mathrm{ns}} N : U_1$ and $\Delta_2 \vdash_{\mathrm{ns}} N : U_2$.
    * For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma_1'(y)$.
    * For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma_2'(y)$.

    Therefore, with $\Gamma' = \Gamma_1' \cap \Gamma_2'$, $\Delta = \Delta_1 \cap \Delta_2$ and $U = U_1 \cap U_2$:
    * $\Gamma_1 \cap \Gamma_2 \approx (\Gamma_1' \cap \Delta_1) \cap (\Gamma_2' \cap \Delta_2) \approx (\Gamma_1' \cap \Gamma_2') \cap (\Delta_1 \cap \Delta_2)$.
    * We have $\Gamma_1' \cap \Gamma_2', x : U_1 \cap U_2 \vdash_{\mathrm{ns}} M_1 M_2 : [kv]$ and, by Lemma 28.1, $\Delta_1 \cap \Delta_2 \vdash_{\mathrm{ns}} N : U_1 \cap U_2$.
    * Assume $y \notin \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma_1'(y) \cap \Gamma_2'(y) = (\Gamma_1' \cap \Gamma_2')(y)$.

□

Lemma 31 (Typing accumulators)**.**
If $\Gamma \vdash M : F$*,* $\mathit{input}(\Gamma)$ *and* $\mathit{acc}(M)$*, then* F *is of the form* $[k]$*.*

Proof. By induction on $\mathrm{acc}(M)$.

- For $\dfrac{}{\mathrm{acc}(x)}$ with $M = x$: Then, $\Gamma = (x : F)$. Hence, $\mathrm{input}(F)$. Therefore, $F = [\nabla]$ which is of the form $[k]$.

- For $\dfrac{\mathrm{acc}(M_1)}{\mathrm{acc}(M_1 M_2)}$ with $M = M_1 M_2$: Then, we are in one of the following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M_1 : A \to F$ and $\Gamma_2 \vdash M_2 : A$. By Lemma 27.10, we have $\mathrm{input}(\Gamma_1)$. By induction hypothesis, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash M_1 : [k]$ and $\Gamma_2 \vdash M_2 : [v]$. Then, we can conclude.

□

Lemma 32 (Typing normal forms)**.**
If M cannot be reduced by $\to_\beta$, then there exist Γ and v such that $\Gamma \vdash_{opt} M : [v]$.

Proof. By induction on M.

By Lemma 22, either M is of the form $\lambda x.M_1$ or we have $\mathrm{acc}(M)$. Hence, we are in one of the following cases:

- M is of the form $\lambda x.M_1$: By induction hypothesis, there exist Γ and v such that $\Gamma \vdash_{\mathrm{opt}} M_1 : [v]$. Therefore, $\Gamma \vdash_{\mathrm{ns}} M_1 : [v]$ and $\mathrm{input}(\Gamma)$. There exist a unique Γ_1 and a unique U such that $\Gamma = (\Gamma_1, x : U)$. Hence, $\mathrm{input}(\Gamma_1)$ and $\mathrm{input}(U)$. Then, $\Gamma_1 \vdash_{\mathrm{ns}} \lambda x.M_1 : [\lambda v]$ with $\mathrm{input}(\Gamma_1)$. Therefore, $\Gamma_1 \vdash_{\mathrm{opt}} M : [\lambda v]$.
- M is a variable x: Then, $x : [\nabla] \vdash_{\mathrm{opt}} M : [\nabla]$.
- M is of the form $M_1 M_2$ with $\mathrm{acc}(M_1)$: Then, M_1 and M_2 cannot be reduced by $\to_\beta$. By induction hypothesis, there exist Γ_1, Γ_2, v_1 and v_2 such that $\Gamma_1 \vdash_{\mathrm{opt}} M_1 : [v_1]$ and $\Gamma_2 \vdash_{\mathrm{opt}} M_2 : [v_2]$. Therefore, $\Gamma_1 \vdash_{\mathrm{ns}} M_1 : [v_1]$, $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : [v_2]$, $\mathrm{input}(\Gamma_1)$ and $\mathrm{input}(\Gamma_2)$. By Lemma 31, v_1 if of the form k_1. By Lemma 27.11, we have $\mathrm{input}(\Gamma_1 \cap \Gamma_2)$. Hence, $\Gamma_1 \cap \Gamma_2 \vdash_{\mathrm{ns}} M_1 M_2 : [k_1 v_2]$ with $\mathrm{input}(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\mathrm{opt}} M : [k_1 v_2]$.

□

Theorem 16 (Subject Expansion)**.**
If $\Gamma' \vdash_{opt} M' : F$ and $M \Rightarrow_h M'$, then there exists Γ such that $\Gamma \subseteq \Gamma'$ and $\Gamma \vdash_{opt} M : F$.

Proof. We prove by induction on $M \Rightarrow_h M'$ and $M \rightsquigarrow_h M'$ that if $\Gamma' \vdash_{\mathrm{ns}} M : F$ and $\mathrm{input}(\Gamma')$, and if we are in one of the following cases:

- We have $M \rightsquigarrow_h M'$
- We have $M \Rightarrow_h M'$ and $\mathrm{output}(F)$.

Then, there exists Γ such that $\Gamma \subseteq \Gamma'$, $\mathrm{input}(\Gamma)$, and $\Gamma \vdash_{\mathrm{ns}} M : F$.

- For $\dfrac{x \in \mathrm{fv}(M_1)}{(\lambda x.M_1) M_2 \rightsquigarrow_h M_1\{x := M_2\}}$ with $M' = M_1\{x := M_2\}$: By Lemma 30, there exist Γ_1, Γ_2 and U such that $\Gamma' \approx \Gamma_1 \cap \Gamma_2$, $\Gamma_1, x : U \vdash_{\mathrm{ns}} M_1 : F$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : U$. By Lemma 28.4, U is of the form A. Therefore, $\Gamma_1 \vdash_{\mathrm{ns}} \lambda x.M_1 : A \to F$. Hence, $\Gamma_1 \cap \Gamma_2 \vdash_{\mathrm{ns}} (\lambda x.M_1) M_2 : F$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma'$ and, by Lemma 27.10, $\mathrm{input}(\Gamma_1 \cap \Gamma_2)$ (because $\Gamma' \subseteq \Gamma_1 \cap \Gamma_2$).
- For $\dfrac{x \notin \mathrm{fv}(M_1) \quad M_2 \Rightarrow_h M_2'}{(\lambda x.M_1) M_2 \rightsquigarrow_h (\lambda x.M_1) M_2'}$ with $M' = (\lambda x.M_1) M_2'$: Then, we are in one of the following cases:

- There exist Γ_1, Γ_2' and A such that $\Gamma' = \Gamma_1 \cap \Gamma_2'$, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : A \to F$ and $\Gamma_2' \vdash_{\text{ns}} M_2' : A$. Therefore, by the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash_{\text{ns}} M_1 : F$ and we have either $U = A$ or $U = \omega$ and output(A). If $U = A$ then, by Lemma 28.4, $x \in \text{fv}(M_1)$: contradiction. Hence, $U = \omega$ and output(A). Therefore, A is of the form G. By Lemma 27.10, we have input(Γ_1) and input(Γ_2'). By induction hypothesis, there exists Γ_2 such that input(Γ_2), $\Gamma_2 \subseteq \Gamma_2'$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : G$. By Lemma 27.11, input$(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} (\lambda x.M_1)M_2 : F$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2' = \Gamma'$ and input$(\Gamma_1 \cap \Gamma_2)$.
- There exist Γ_1, Γ_2, k and v such that $\Gamma' = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M_2' : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.

- For $\dfrac{x \notin \text{fv}(M_1) \quad M_2 \text{ cannot be reduced by } \to_\beta}{(\lambda x.M_1)M_2 \rightsquigarrow_h M_1}$ with $M' = M_1$: By Lemma 28.4, $x \notin \mathsf{Dom}(\Gamma')$. Hence, $\Gamma' = (\Gamma', x : \omega)$. By Lemma 32, there exist Γ_2 and G such that $\Gamma_2 \vdash_{\text{opt}} M_2 : G$. Hence, input$(\Gamma_2)$, output$(G)$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : G$. Therefore, $\Gamma' \vdash_{\text{ns}} \lambda x.M_1 : G \to F$. By Lemma 27.11, we have input$(\Gamma' \cap \Gamma_2)$. Therefore, $\Gamma' \cap \Gamma_2 \vdash_{\text{ns}} (\lambda x.M_1)M_2 : F$ with $\Gamma' \cap \Gamma_2 \subseteq \Gamma'$ and input$(\Gamma' \cap \Gamma_2)$.

- For $\dfrac{M_1 \rightsquigarrow_h M_1'}{M_1M_2 \rightsquigarrow_h M_1'M_2}$ with $M' = M_1'M_2$, we are in one of the following cases:
  - There exist Γ_1', Γ_2 and A such that $\Gamma' = \Gamma_1' \cap \Gamma_2$, $\Gamma_1' \vdash_{\text{ns}} M_1' : A \to F$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : A$: Then, by Lemma 27.10, input(Γ_1') and input(Γ_2). By induction hypothesis, there exists Γ_1 such that $\Gamma_1 \subseteq \Gamma_1'$, input$(\Gamma_1)$ and $\Gamma_1 \vdash_{\text{ns}} M_1 : A \to F$. By Lemma 27.11, input$(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} M_1M_2 : F$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1' \cap \Gamma_2 = \Gamma'$ and input$(\Gamma_1 \cap \Gamma_2)$.
  - There exist Γ_1', Γ_2, k and v such that $\Gamma' = \Gamma_1' \cap \Gamma_2$, $F = [kv]$, $\Gamma_1' \vdash_{\text{ns}} M_1' : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v]$: Then, by Lemma 27.10, input(Γ_1') and input(Γ_2). By induction hypothesis, there exists Γ_1 such that $\Gamma_1 \subseteq \Gamma_1'$, input$(\Gamma_1)$ and $\Gamma_1 \vdash_{\text{ns}} M_1 : [k]$. By Lemma 27.11, input$(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} M_1M_2 : [kv]$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1' \cap \Gamma_2 = \Gamma'$ and input$(\Gamma_1 \cap \Gamma_2)$.

- For $\dfrac{\text{acc}(M_1) \quad M_2 \Rightarrow_h M_2'}{M_1M_2 \rightsquigarrow_h M_1M_2'}$ with $M' = M_1M_2'$, we are in one of the two following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma' = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} M_1 : A \to F$ and $\Gamma_2 \vdash_{\text{ns}} M_2' : A$. By Lemma 27.10, input$(\Gamma_1)$. By Lemma 31, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2', k and v such that $\Gamma' = \Gamma_1 \cap \Gamma_2'$, $F = [kv]$, $\Gamma_1 \vdash_{\text{ns}} M_1 : [k]$ and $\Gamma_2' \vdash_{\text{ns}} M_2' : [v]$. By Lemma 27.10, input$(\Gamma_1)$ and input(Γ_2'). By induction hypothesis, there exists Γ_2 such that $\Gamma_2 \subseteq \Gamma_2'$, input$(\Gamma_2)$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v]$. By Lemma 27.11, input$(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} M_1M_2 : [kv]$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2' = \Gamma'$ and input$(\Gamma_1 \cap \Gamma_2')$.

- For $\dfrac{M \rightsquigarrow_h M'}{M \Rightarrow_h M'}$: Trivial.

- For $\dfrac{M_1 \Rightarrow_h M_1'}{\lambda x.M_1 \Rightarrow_h \lambda x.M_1'}$ with $M' = \lambda x.M_1'$ and output(F): We have output(F), so F is not an arrow $A \to G$. Therefore, there exist U' and v such that $F = [\lambda v]$, input(U') and $\Gamma', x : U' \vdash_{\text{ns}} M_1' : [v]$. Hence, input$(\Gamma', x : U')$. By induction hypothesis, there exists Γ_1 such that $\Gamma_1 \subseteq (\Gamma', x : U')$, input$(\Gamma_1)$ and $\Gamma_1 \vdash_{\text{ns}} M_1 : [v]$. There exist an unique Γ and a unique U such that

$\Gamma_1 = (\Gamma, x : U)$. Therefore, $\Gamma \subseteq \Gamma'$, $U \subseteq U'$, input(Γ) and input(U). Hence, $\Gamma \vdash_{\text{ns}} \lambda x.M_1 : [\lambda v]$ with input(Γ).

□

Lemma 33 (Refined Substitution Lemma)**.**
If $\Gamma, x : U \vdash^n_{ns} M : A$ *and* $\Delta \vdash^m_{ns} N : U$*, then there exists* Γ' *such that:*

- $\Gamma' \approx \Gamma \cap \Delta$.
- $\Gamma' \vdash^{n+m}_{ns} M\{x := N\} : A$.
- *For all* $y \notin \mathsf{Dom}(\Delta)$, $\Gamma(y) = \Gamma'(y)$.

Proof. By induction on $\Gamma, x : U \vdash_{\text{ns}} M : A$.

- For $\dfrac{}{x : F \vdash^0_{\text{ns}} x : F}$ with $\Gamma = ()$, $n = 0$, $U = F$, $M = x$ and $A = F$: We have $M\{x := N\} = N$. Therefore, with $\Gamma' = \Delta$:
  - $\Delta = () \cap \Delta$.
  - $\Delta \vdash^m_{\text{ns}} M\{x := N\} : F$ and $n + m = m$.
  - Assume $y \notin \mathsf{Dom}(\Delta)$. Then, $\Delta(y) = \omega = ()(y)$.
- For $\dfrac{}{y : F \vdash^0_{\text{ns}} y : F}$ with $x \neq y$, $\Gamma = (y : F)$, $n = 0$, $U = \omega$, $M = y$ and $A = F$: We have $M\{x := N\} = y = M$. By hypothesis, $\Delta \vdash^m_{\text{ns}} N : \omega$. Hence, $m = 0$ and $\Delta = ()$. Therefore, with $\Gamma' = \Gamma = (y : F)$:
  - $(y : F) = \Gamma = \Gamma \cap ()$.
  - $y : F \vdash^0_{\text{ns}} M\{x := N\} : F$ and $n + m = 0$.
  - Assume $z \notin \mathsf{Dom}(())$. Then, $(y : F)(z) = \Gamma(z)$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\text{ns}} M : A_1 \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\text{ns}} M : A_2}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2}_{\text{ns}} M : A_1 \cap A_2}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$ and $A = A_1 \cap A_2$. By hypothesis, $\Delta \vdash^m_{\text{ns}} N : U_1 \cap U_2$. By Lemma 28.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\text{ns}} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\text{ns}} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that:
  - $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$.
  - $\Gamma'_1 \vdash^{n_1+m_1}_{\text{ns}} M\{x := N\} : A_1$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\text{ns}} M\{x := N\} : A_2$.
  - For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
  - For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

  Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$:
  - $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$.
  - We have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2}_{\text{ns}} M\{x := N\} : A_1 \cap A_2$ and $n + m = n_1 + m_1 + n_2 + m_2$.
  - Assume $y \notin \mathsf{Dom}(\Delta) = \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n_{\text{ns}} M_1 : F}{\Gamma, x : U \vdash^n_{\text{ns}} \lambda y.M_1 : B \to F}$ with $M = \lambda y.M_1$, $y \notin \text{fv}(N)$, $y \neq x$, $A = B \to F$ and $V = B$ or $V = \omega$ and output(B). We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exist Γ' such that:
  - $\Gamma' \approx (\Gamma, y : V) \cap \Delta$.
  - $\Gamma' \vdash^{n+m}_{\text{ns}} M_1\{x := N\} : F$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

By Lemma 28.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $\Gamma' = (\Gamma'', y : V')$. Hence, $(\Gamma'', y : V') \approx (\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V' \approx V$. We have $y \notin \mathsf{Dom}(\Delta)$, so $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:

- $\Gamma'' \approx \Gamma \cap \Delta$.
- We have $\Gamma'', y : V \vdash^{n+m}_{\mathrm{ns}} M_1\{x := N\} : F$. Therefore, $\Gamma'' \vdash^{n+m}_{\mathrm{ns}} \lambda y.M_1\{x := N\} : B \to F$.
- Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma''(z) \approx (\Gamma \cap \Delta)(z) = \Gamma(z) \cap \Delta(z) = \Gamma(z) \cap \omega = \Gamma(z)$.

  If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(y) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.

  If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.

- For $\dfrac{\Gamma, x : U, y : V \vdash^{n}_{\mathrm{ns}} M_1 : [v] \quad \mathrm{input}(V)}{\Gamma, x : U \vdash^{n}_{\mathrm{ns}} \lambda y.M_1 : [\lambda v]}$ with $M = \lambda y.M_1$, $y \notin \mathrm{fv}(N)$, $y \neq x$ and $A = [\lambda v]$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exist Γ' such that:
  - $\Gamma' \approx (\Gamma, y : V) \cap \Delta$.
  - $\Gamma' \vdash^{n+m}_{\mathrm{ns}} M_1\{x := N\} : [v]$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

  By Lemma 28.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $\Gamma' = (\Gamma'', y : V')$. Hence, $(\Gamma'', y : V') \approx (\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V' \approx V$. We have $y \notin \mathsf{Dom}(\Delta)$, so $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:
  - $\Gamma'' \approx \Gamma \cap \Delta$.
  - We have $\Gamma'', y : V \vdash^{n+m}_{\mathrm{ns}} M_1\{x := N\} : [v]$. Therefore, $\Gamma'' \vdash^{n+m}_{\mathrm{ns}} \lambda y.M_1\{x := N\} : [\lambda v]$.
  - Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma''(z) \approx (\Gamma \cap \Delta)(z) = \Gamma(z) \cap \Delta(z) = \Gamma(z) \cap \omega = \Gamma(z)$.

    If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(y) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.

    If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.

- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\mathrm{ns}} M_1 : B \to F \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : B}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2+1}_{\mathrm{ns}} M_1 M_2 : F}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $U = U_1 \cap U_2$, $M = M_1 M_2$ and $A = F$. We have $(M_1 M_2)\{x := N\} = M_1\{x := N\} M_2\{x := N\}$. By hypothesis, $\Delta \vdash^{m}_{\mathrm{ns}} N : U_1 \cap U_2$. By Lemma 28.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\mathrm{ns}} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\mathrm{ns}} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that:
  - $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$.
  - $\Gamma'_1 \vdash^{n_1+m_1}_{\mathrm{ns}} M_1 : B \to F$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\mathrm{ns}} M_2 : B$.
  - For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
  - For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

  Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$:
  - $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$.
  - We have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2+1}_{\mathrm{ns}} M_1\{x := N\} M_2\{x := N\} : F$ and $n + m = n_1 + m_1 + n_2 + m_2 + 1$.

- Assume $y \notin \mathsf{Dom}(\Delta) = \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.

- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\text{ns}} M_1 : [k] \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\text{ns}} M_2 : [v]}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2}_{\text{ns}} M_1 M_2 : [kv]}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$, $M = M_1 M_2$ and $A = [kv]$. We have $(M_1 M_2)\{x := N\} = M_1\{x := N\} M_2\{x := N\}$. By hypothesis, $\Delta \vdash^{m}_{\text{ns}} N : U_1 \cap U_2$. By Lemma 28.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\text{ns}} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\text{ns}} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that:
  - $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$.
  - $\Gamma'_1 \vdash^{n_1+m_1}_{\text{ns}} M_1 : [k]$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\text{ns}} M_2 : [v]$.
  - For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
  - For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

  Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$:
  - $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$.
  - We have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2}_{\text{ns}} M_1\{x := N\} M_2\{x := N\} : [kv]$ and $n + m = n_1 + m_1 + n_2 + m_2$.
  - Assume $y \notin \mathsf{Dom}(\Delta) = \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.

□

Lemma 34 (Measure of normal forms)**.**
If $\Gamma \vdash^n M : F$, M cannot be reduced by $\to_\beta$, input(Γ) and output(F), then $n = 0$.

Proof. By induction on M: By Lemma 22, M is of the form $\lambda x.M_1$ or $\mathrm{acc}(M)$. Therefore, we are in one of the following cases:

- M is of the form $\lambda x.M_1$: We have output(F), so F is not an arrow $A \to G$. Therefore, there exist U and G such that input(U), output(G) and $\Gamma, x : U \vdash^n M_1 : G$. Hence, input($\Gamma, x : U$). By induction hypothesis, $n = 0$.
- M is a variable x: Then, $n = 0$.
- M is of the form $M_1 M_2$ with $\mathrm{acc}(M_1)$: Then, we are in one of the following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M_1 : A \to F$ and $\Gamma_2 \vdash M_2 : A$. By Lemma 27.10, we have input(Γ_1). By Lemma 31, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2} M_2 : [v]$. By Lemma 27.10, input($\Gamma_1$) and input($\Gamma_2$). By induction hypothesis, $n_1 = 0$ and $n_2 = 0$. Therefore $n = 0$.

□

Theorem 18 (Refined Subject Reduction)**.**
If $\Gamma \vdash^n_{opt} M : F$ and $M \Rightarrow_h M'$, then there exists Γ' such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash^{n-1}_{opt} M' : F$.

Proof. We prove by induction on $M \Rightarrow_h M'$ and $M \rightsquigarrow_h M'$ that if $\Gamma \vdash^n_{\text{ns}} M : F$, input($\Gamma$), and if we are in one the following cases:

- We have $M \Rightarrow_h M'$ and output(F).
- We have $M \rightsquigarrow_h M'$.

Then, there exists Γ' such that $\Gamma \subseteq \Gamma'$, $\Gamma' \vdash^{n-1}_{\text{ns}} M' : F$ and then, by Lemma 27.10, we have input(Γ').

- For $\dfrac{x \in \text{fv}(M_1)}{(\lambda x.M_1)M_2 \rightsquigarrow_h M_1\{x := \ M_2\}}$ with $M = (\lambda x.M_1)M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\text{ns}} \lambda x.M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\text{ns}} M_2 : A$: By the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash^{n_1}_{\text{ns}} M_1 : F$ and $U = A$ or $U = \omega$ and output(A). By Lemma 28.4, if $U = \omega$, then $x \notin \text{fv}(M_1)$: contradiction. Therefore, $A = U$. By Lemma 33, there exist Γ' such that $\Gamma' \approx \Gamma_1 \cap \Gamma_2$ and $\Gamma' \vdash^{n_1+n_2}_{\text{ns}} M_1\{x := \ M_2\} : F$ with $\Gamma \subseteq \Gamma'$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.
- For $\dfrac{x \notin \text{fv}(M_1) \quad M_2 \Rightarrow_h M_2'}{(\lambda x.M_1)M_2 \rightsquigarrow_h (\lambda x.M_1)M_2'}$ with $M = (\lambda x.M_1)M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\text{ns}} \lambda x.M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\text{ns}} M_2 : A$: By the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash^{n_1}_{\text{ns}} M_1 : F$ and $U = A$ or $U = \omega$ and output(A). By Lemma 28.4, if $U = A$, then $x \in \text{fv}(M_1)$: contradiction. Therefore, $U = \omega$ and output(A). Hence, A is of the form G. By Lemma 27.10, input(Γ_1) and input(Γ_2). By induction hypothesis, there exists Γ_2' such that $\Gamma_2 \subseteq \Gamma_2'$ and $\Gamma_2' \vdash^{n_2-1}_{\text{ns}} M_2' : G$. Therefore, $\Gamma_1 \cap \Gamma_2' \vdash^{n_1+n_2}_{\text{ns}} (\lambda x.M_1)M_2' : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2'$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.
- For $\dfrac{x \notin \text{fv}(M_1) \quad M_2 \text{ cannot be reduced by } \to_\beta}{(\lambda x.M_1)M_2 \rightsquigarrow_h M_1}$ with $M = (\lambda x.M_1)M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\text{ns}} \lambda x.M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\text{ns}} M_2 : A$: By the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash^{n_1}_{\text{ns}} M_1 : F$ and $U = A$ or $U = \omega$ and output(A). By Lemma 28.4, if $U = A$, then $x \in \text{fv}(M_1)$: contradiction. Therefore, $U = \omega$ and output(A). Hence, A is of the form G and $\Gamma_1 = (\Gamma_1, x : U)$. By Lemma 27.10, input($\Gamma_1$) and input($\Gamma_2$). By Lemma 34, $n_2 = 0$. Therefore, $\Gamma_1 \vdash^{n_1+n_2}_{\text{ns}} M_1 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.
- For $\dfrac{M_1 \rightsquigarrow_h M_1'}{M_1M_2 \rightsquigarrow_h M_1'M_2}$ with $M = M_1M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\text{ns}} M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\text{ns}} M_2 : A$: By Lemma 27.10, input($\Gamma_1$) and input($\Gamma_2$). By induction hypothesis, there exist Γ_1' such that $\Gamma_1 \subseteq \Gamma_1'$

and $\Gamma_1' \vdash^{n_1-1} M_1' : A \to F$. Therefore, $\Gamma_1' \cap \Gamma_2 \vdash_{\text{ns}}^{n_1+n_2} M_1'M_2 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1' \cap \Gamma_2$.

- There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash_{\text{ns}}^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}}^{n_2} M_2 : [v]$: By Lemma 27.10, input($\Gamma_1$) and input($\Gamma_2$). By induction hypothesis, there exists Γ_1' such that $\Gamma_1 \subseteq \Gamma_1'$ and $\Gamma_1' \vdash_{\text{ns}}^{n_1-1} M_1' : [k]$. Therefore, $\Gamma_1' \cap \Gamma_2 \vdash_{\text{ns}}^{n_1+n_2-1} M_1'M_2 : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1' \cap \Gamma_2$.

- For $\dfrac{\text{acc}(M_1) \quad M_2 \Rightarrow_h M_2'}{M_1M_2 \rightsquigarrow_h M_1M_2'}$, we are in one of the following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} M_1 : A \to F$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : A$. By Lemma 27.10, input($\Gamma_1$) and input($\Gamma_2$). By Lemma 31, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash_{\text{ns}}^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}}^{n_2} M_2 : [v]$. By Lemma 27.10, input($\Gamma_1$) and input($\Gamma_2$). By induction hypothesis, there exists Γ_2' such that $\Gamma_2 \subseteq \Gamma_2'$ and $\Gamma_2' \vdash^{n_2-1} M_2' : [v]$. Therefore, $\Gamma_1 \cap \Gamma_2' \vdash^{n_1+n_2-1} M_1M_2' : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2'$.
- For $\dfrac{M \rightsquigarrow_h M'}{M \Rightarrow_h M'}$: Trivial.
- For $\dfrac{M_1 \Rightarrow_h M_1'}{\lambda x.M_1 \Rightarrow_h \lambda x.M_1'}$ with $M = \lambda x.M_1$ and output(F): By the fact that we have output(F), F is not an arrow $A \to G$. Therefore, there exist U and v such that input(U), $F = [\lambda v]$ and $\Gamma, x : U \vdash_{\text{ns}}^{n} M_1 : [v]$. Hence, input($\Gamma, x : U$). By induction hypothesis, there exists Γ_1 such that $(\Gamma, x : U) \subseteq \Gamma_1$ and $\Gamma_1 \vdash_{\text{ns}}^{n-1} M_1' : [v]$. There exist an unique Γ' and a unique U' such that $\Gamma_1 = (\Gamma', x : U')$. Therefore, $\Gamma \subseteq \Gamma'$ and $U \subseteq U'$. By Lemma 26.5, we have input(U'). Hence, $\Gamma' \vdash^{n-1} \lambda x.M_1' : [\lambda v]$.

□

Lemma 37 ($\Rightarrow_\alpha$ can be used on a non-normal form)**.**
If M can be reduced by $\to_\alpha$ then there exists M' such that $M \Rightarrow_\alpha M'$.

Proof. By induction on M, we prove that if M can be reduced by $\to_\alpha$ then:

- If M is of the form $\lambda x.M_1$, $\mathfrak{B}$, fix(M_1) or $cM_1 \dots M_n$ with m arity of c and $n \leq m$: then there exists M' such that $M \Rightarrow_\alpha M'$.
- In all other cases, there exists M' such that $M \rightsquigarrow_\alpha M'$. Therefore, by Theorem 24.1, we have $M \Rightarrow_\alpha M'$.

Assume M can be reduced by $\to_\alpha$.

Therefore we are in one of the following cases:

- M is a variable x or M is a constant c. Therefore, M cannot be reduced by $\to_\alpha$. Contradiction.
- M is of the form $\lambda x.M_1$. Therefore, M_1 can be reduced by $\to_\alpha$. Hence, by induction hypothesis, there exists M_1' such that $M_1 \Rightarrow_\alpha M_1'$. Therefore, $M \Rightarrow_\alpha \lambda x.M_1'$.
- M is of the form $(c_1\vec{x_1}.M_1, \dots, c_n\vec{x_n}.M_n)$. Therefore, there exist i such that M_i can be reduced by $\to_\alpha$. Hence, by induction hypothesis, there exists M_i' such that $M_i \Rightarrow_\alpha M_i'$. Therefore, $M \Rightarrow_\alpha (c_1\vec{x_1}.M_1, \dots, c_i\vec{x_i}.M_i', \dots, c_n\vec{x_n}.M_n)$.
- M is of the form fix(M_1). Therefore, M_1 can be reduced by $\to_\alpha$. Hence, by induction hypothesis, there exists M_1' such that $M_1 \Rightarrow_\alpha M_1'$. Therefore, $M \Rightarrow_\alpha \text{fix}(M_1')$.

- M is of the form $(\lambda x.M_1)M_2$.
  - If $x \in \mathrm{fv}(M_1)$, then $M \rightsquigarrow_\alpha M_1\{x := M_2\}$.
  - If $x \notin \mathrm{fv}(M_1)$ and M_2 can be reduced by $\rightarrow_\alpha$: By induction hypothesis, there exists M_2' such that $M_2 \Rightarrow_\alpha M_2'$. Therefore $M \rightsquigarrow_\alpha (\lambda x.M_1)M_2'$.
  - If $x \notin \mathrm{fv}(M_1)$ and M_2 cannot be reduced by $\rightarrow_\alpha$, then $M \rightsquigarrow_\alpha M_1$.
- M is of the form M_1M_2 and $M \in \mathrm{CF}$.
  - If $\alpha = s$, then $M \rightsquigarrow_s M$.
  - If $\alpha = u$: Then either M_1 or M_2 can be reduced by $\rightarrow_u$.

    If M_1 can be reduced by $\rightarrow_u$, then by induction hypothesis, there exists M_1' such that $M_1 \Rightarrow_u M_1'$. Therefore $M \rightsquigarrow_u M_1'M_2$.

    By a similar argument, if M_2 can be reduced by $\rightarrow_u$ then there exists M_2' such that $M \rightsquigarrow_u M_1M_2'$.
- M is of the form $(c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n)(c_i\vec{N})$ with $\vec{N} = N_1 \ldots N_m$, $\vec{x_i} = y_1 \ldots y_m$ and m arity of c.
  - If there exists j such that $i \neq j$ and M_j can be reduced by $\rightarrow_\alpha$: Therefore, by induction hypothesis, there exists M_j' such that $M_j \Rightarrow_\alpha M_j'$. Hence $M \rightsquigarrow_\alpha (c_1\vec{x_1}.M_1, \ldots, c_j\vec{x_i}.M_j', \ldots, c_n\vec{x_n}.M_n)(c_i\vec{N})$.
  - If there exists j such that $y_j \notin \mathrm{fv}(M_i)$ and N_j can be reduced by $\rightarrow_\alpha$: Therefore, by induction hypothesis, there exists N_j' such that $N_j \Rightarrow_\alpha N_j'$. Hence $M \rightsquigarrow_\alpha (c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n)(cN_1 \ldots N_j' \ldots N_m)$.
  - If for all $j \neq i$, M_j cannot be reduced by $\rightarrow_\alpha$ and for all j, if $y_j \notin \mathrm{fv}(M_i)$ then N_j cannot be reduced by $\rightarrow_\alpha$: Then $M \rightsquigarrow_\alpha M_i\{\vec{x_i} := \vec{N}\}$.
- M is of the form $\mathfrak{B}M_1$ and M_1 is not of the form $\lambda x.M_2$, $\mathfrak{B}'$, $\mathrm{fix}(M_2)$ or $cN_1 \ldots N_n$ with m arity of c and $n \leq m$:
  - If M_1 can be reduced by $\rightarrow_\alpha$: By induction hypothesis, there exists M_1' such that $M_1 \rightsquigarrow_\alpha M_1'$. Therefore, $\mathfrak{B}M_1 \rightsquigarrow_\alpha \mathfrak{B}M_1'$.
  - If M_1 cannot be reduced by $\rightarrow_\alpha$: By Lemma 36, we have $\mathrm{accu}_\alpha(M_1)$. We can also deduce that $\mathfrak{B}$ can be reduced by $\rightarrow_\alpha$. Hence, by induction hypothesis, there exists $\mathfrak{B}'$ such that $\mathfrak{B} \Rightarrow_\alpha \mathfrak{B}'$. Therefore $M \rightsquigarrow_\alpha \mathfrak{B}'M_1$.
- M is of the form $\mathrm{fix}(M_1)M_2$: Therefore $M \rightsquigarrow_\alpha M_1M_2\mathrm{fix}(M_1)$.
- M is of the form $cM_1 \ldots M_n$ with m arity of c and $n \leq m$: Therefore, there exists i such that M_i can be reduced by $\rightarrow_\alpha$. Hence, there exists M_i' such that $M_i \Rightarrow_\alpha M_i'$. Therefore, $M \Rightarrow_\alpha cM_1 \ldots M_i' \ldots M_n$.
- M is of the form M_1M_2 and M_1 is not of the form $\lambda x.M_3$, $\mathfrak{B}$, $\mathrm{fix}(M_3)$, $cN_1 \ldots N_n$ with m arity of c and $n \leq m$:
  - If M_1 can be reduced by $\rightarrow_\alpha$: By induction hypothesis, there exists M_1' such that $M_1 \rightsquigarrow_\alpha M_1'$. Therefore, $M \rightsquigarrow_\alpha M_1'M_2$.
  - If M_1 cannot be reduced by $\rightarrow_\alpha$: By Lemma 36, we have $\mathrm{accu}_\alpha(M_1)$. We can also deduce that M_2 can be reduced by $\rightarrow_\alpha$. By induction hypothesis, there exists M_2' such that $M_2 \Rightarrow_\alpha M_2'$. Therefore $M \rightsquigarrow_\alpha M_1M_2'$.

□

Lemma 40 (Substitution lemma)**.**
If $\Gamma, x : U \vdash_\alpha^n M : A$ *and* $\Delta \vdash_\alpha^m N : U$*, then there exists* Γ' *such that* $\Gamma' \approx \Gamma \cap \Delta$ *and* $\Gamma' \vdash_\alpha^{n+m} M\{x := N\} : A$*.*

Proof. By induction on $\Gamma, x : U \vdash_\alpha^n M : A$.

- For $\dfrac{}{x : F \vdash^0_\alpha x : F}$ with $M = x$, $A = F$ and $\Gamma = ()$: We have $M\{x := N\} = N$. By hypothesis, $\Delta \vdash^m_\alpha N : U$. We also have $\Gamma \cap \Delta = () \cap \Delta = \Delta$ and $n + m = 0 + m = m$.
- For $\dfrac{}{y : F \vdash^0_\alpha x : F}$ with $M = y$, $\Gamma = (y : F)$, $U = \omega$ and $y \neq x$: We have $y\{x := N\} = y$. By the fact that $U = \omega$, we have $m = 0$ and $\Delta = ()$. We also have $\Gamma \cap \Delta = \Gamma \cap () = \Gamma = (x : F)$ and $n + m = 0 + 0 = 0 = n$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n_\alpha M_1 : F \quad B \subseteq V}{\Gamma, x : U \vdash^n_\alpha \lambda y.M_1 : B \to F}$ with $M = \lambda x.M_1$, $A = B \to F$ and $y \notin \mathrm{fv}(N)$: We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$ and $(\Gamma, x : U, y : V) = (\Gamma, y : V, x : U)$. By induction hypothesis, there exists Γ' such that $\Gamma' \approx (\Gamma, y : V) \cap \Delta$ and $\Gamma' \vdash^{n+m}_\alpha M_1\{x := N\} : F$. By Lemma 39.4, $y \notin \mathsf{Dom}(\Delta)$. Hence, $\Delta = (\Delta, y : \omega)$. There exists Γ'' and V' such that $\Gamma' = (\Gamma'', y : V')$. Hence, $(\Gamma'', y : V') \approx (\Gamma \cap \Delta, y : V' \cap \omega)$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V \approx V'$. Hence, $B \subseteq V'$ and $\Gamma'' \vdash^{n+m}_\alpha \lambda y.M_1\{x := N\} : B \to F$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_\alpha M_1 : F \quad \Gamma_2, x : U_2 \vdash^{n_2}_\alpha M_2 : B \quad F@_\alpha B : G}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2+1}_\alpha M_1 M_2 : G}$ with $M = M_1 M_2$, $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $G = A$ and $U = U_1 \cap U_2$: We have $(M_1 M_2)\{x := N\} = M_1\{x := N\}M_2\{x := N\}$. By Lemma 39.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_\alpha N : U_1$ and $\Delta_2 \vdash^{m_2}_\alpha N : U_2$.

  By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1}_\alpha M_1\{x := N\} : F$, $\Gamma'_2 \vdash^{n_1+m_1}_\alpha M_2\{x := N\} : B$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+n_2+1}_\alpha M_1\{x := N\}M_2\{x := N\} : G$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$, $n_1 + m_1 + n_2 + m_2 + 1 = n_1 + n_2 + m_1 + m_2 + 1 = n + m + 1$.
- We use similar proofs for the other rules.

□

Theorem 27 (Subject reduction)**.**
If $\Gamma \vdash^n_\alpha M : A$ *and* $M \to_\alpha M'$*, then there exists* Γ' *and* n' *such that* $\Gamma \subseteq \Gamma'$*,* $n > n'$ *and* $\Gamma' \vdash^{n'}_\alpha M' : A$*.*

Proof. First by induction on $M \to_\alpha M'$ then by induction on A.

- If A is of the form $A_1 \cap A_2$: By Lemma 39.1, there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1}_\alpha M : A_1$ and $\Gamma_2 \vdash^{n_2}_\alpha M : A_2$. By induction hypothesis on $(M \to_\alpha M', A_1)$ and $(M \to_\alpha M', A_2)$, there exist Γ'_1, Γ'_2, n'_1 and n'_2 such that $\Gamma_1 \subseteq \Gamma'_1$, $\Gamma_2 \subseteq \Gamma'_2$, $n'_1 < n_1$, $n'_2 < n_2$, $\Gamma'_1 \vdash^{n'_1}_\alpha M' : A_1$ and $\Gamma'_2 \vdash^{n'_2}_\alpha M' : A_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n'_1+n'_2}_\alpha M : A_1 \cap A_2$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma'_2$ and $n'_1 + n'_2 < n_1 + n_2$.
- For $\dfrac{}{(\lambda x.M_1)M_2 \to_\alpha M\{x := N\}}$ with A is of the form F: There exist Γ_1, Γ_2, n_1, n_2, G and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_\alpha M_1 : G$, $\Gamma_2 \vdash^{n_2}_\alpha M_2 : B$ and $G@_\alpha B : F$. Hence, there exist A_1, F_1 and U such that $G = A_1 \to F_1$, $A_1 \subseteq U$, and $\Gamma_1, x : U \vdash^{n_1}_\alpha M_1 : F_1$. Therefore, $A_1 = B$ and $F_1 = F$. By Lemma 39.3, there exist Γ'_2 and m such that $\Gamma_2 \subseteq \Gamma'_2$, $m \leq n_2$ and $\Gamma'_2 \vdash^m_\alpha M_2 : U$. By Lemma 40, there exists Δ such that $\Delta \approx \Gamma_1 \cap \Gamma'_2$ and $\Delta \vdash^{n_1+m}_\alpha M_1\{x := M_2\} : F$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Delta$ and $n_1 + m \leq n_1 + n_2 < n_1 + n_2 + 1$.

- For $\dfrac{}{\mathrm{fix}(M_1)M_2 \to_\alpha M_1M_2\mathrm{fix}(M_2)}$ with A is of the form F: There exist Γ_1, Γ_2, n_1, n_2, G and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash_\alpha^{n_1} \mathrm{fix}(M_1) : G$, $\Gamma_2 \vdash_\alpha^{n_2} M_2 : B$ and $G@_\alpha B : F$. Therefore, $G \neq \delta$. Hence, there exist Γ_3, Γ_4, n_3, n_4, F_1, A_1, A_2, F_2 and F_3 such that $\Gamma_1 = \Gamma_3 \cap \Gamma_4$, $n_1 = n_3 + n_4$, $\Gamma_3 \vdash_\alpha^{n_3} M_1 : F_1$, $F_1@_\alpha A_1 : F_2$, $\Gamma_4 \vdash_\alpha^{n_4} \mathrm{fix}(M_1) : A_2$, $F_2@_\alpha A_2 : F_3$, and $G = A_1 \to F_3$. Therefore, $A_1 = B$ and $F_3 = F$. Hence, $\Gamma_3 \cap \Gamma_2 \vdash_\alpha^{n_3+n_2} M_1M_2 : F_2$. Therefore, $(\Gamma_3 \cap \Gamma_2) \cap \Gamma_4 \vdash_\alpha^{n_3+n_2+n_4} M_1M_2\mathrm{fix}(M_1) : F$ with $n_3 + n_2 + n_4 = n_1 + n_2 = n$ and $(\Gamma_3 \cap \Gamma_2) \cap \Gamma_4 \approx \Gamma_1 \cap \Gamma_2$.
- For $\dfrac{}{(c_1\vec{x_1}.M_1, \ldots, c_n\vec{x_n}.M_n)(c_i\vec{N})}$ with A is of the form F, $\vec{x_i} = y_1 \ldots y_m$ and $\vec{N} = N_1 \ldots N_m$: we adapt the proof for a simple β-redex.
- For $\dfrac{M \in \mathrm{CF}}{M \to_s M}$ with A is of the form F: Then we have $\alpha = s$ and we cannot type such a M if $\alpha = s$. Contradiction.
- For $\dfrac{M \to_\alpha M'}{\lambda x.M_1 \to_\alpha \lambda x.M_1'}$ with A is of the form F: Therefore, there exist B, G and U such that $F = B \to G$, $B \subseteq U$ and $\Gamma, x : U \vdash_\alpha^n M_1 : G$. By induction hypothesis, there exists Γ' and n' such that $(\Gamma, x : U) \subseteq \Gamma'$, $n' < n$ and $\Gamma' \vdash_\alpha^{n'} M_1' : G$. There exists a unique Γ_1 and a unique U' such that $\Gamma' = \Gamma_1, x : U'$. Hence $\Gamma \subseteq \Gamma_1$ and $U \subseteq U'$. Therefore, $B \subseteq U'$ and $\Gamma_1 \vdash_\alpha^{n'} \lambda x.M_1' : B \to G$.
- For $\dfrac{M_i \to_\alpha M_i'}{(c_1\vec{x_1}.M_1, \ldots, c_n.\vec{x_n}.M_n) \to_\alpha (c_1\vec{x_1}.M_1, \ldots, c_i\vec{x_i}.M_i', \ldots, c_n\vec{x_n}.M_n)}$ with A is of the form F: We adapt the proof in the previous case.
- If A is of the form F, the other propagation rules are straightforward.

□

Lemma 42 (Typing accumulators)**.**
If $\Gamma, x_1 : U_1, \ldots, x_n : U_n \vdash_\alpha M : F$ and $accu_\alpha(M, \{x_1, \ldots, x_n\})$ (we have $n = 0$ or $n = 1$), then for all G there exists U_1', …, U_n' such that $\Gamma, x_1 : U_1', \ldots, x_n : U_n' \vdash_\alpha M : G$.

Proof. By induction on $\mathrm{accu}_\alpha(M, \{x_1, \ldots, x_n\})$.

- For $\dfrac{}{\mathrm{accu}_\alpha(x, x)}$: Then $n = 1$, $x_1 = x$ and $\Gamma = ()$. We also have $x : G \vdash_\alpha x : G$. Therefore, $\Gamma, x : G \vdash_\alpha x : G$.
- For $\dfrac{\mathrm{accu}_\alpha(M_1, E)}{\mathrm{accu}_\alpha(M_1M_2, E)}$: Then, there exist Γ_1, Γ_2, H and A such that $(\Gamma, x_1 : U_1, \ldots, x_n : U_n) = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_\alpha M_1 : H$, $\Gamma_2 \vdash_\alpha M_2 : A$ and $H@_\alpha A : F$. There exist Γ_1', Γ_2', V_1, …, V_n, W_1, …, W_n such that $\Gamma_1 = (\Gamma_1', x_1 : V_1, \ldots, x_n : V_n)$, $\Gamma_2 = (\Gamma_2', x_1 : W_1, \ldots, x_n : W_n)$. Therefore, $\Gamma = \Gamma_1' \cap \Gamma_2'$, $U_1 = V_1 \cap W_1$, …, $U_n = V_n \cap W_n$. By induction hypothesis, there exist V_1', …, V_n', such that $\Gamma_1', x_1 : V_1', \ldots, x_n : V_n' \vdash_\alpha M_1 : A \to G$. Therefore, $(\Gamma_1' \cap \Gamma_2', x_1 : V_1' \cap W_1, \ldots, x_nV_n' \cap W_n \vdash_\alpha M_1M_2 : G$.
- For $\dfrac{\mathrm{accu}_\alpha(N, E)}{\mathrm{accu}_\alpha(\mathfrak{B}N, E)}$: There exist Γ_1, Γ_2, V_1, …, V_n, W_1, …, W_n, H and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $U_1 = V_1 \cap W_1, \ldots, U_n = V_n \cap W_n$, $\Gamma_1, x_1 : V_1, \ldots, x_n : V_n \vdash_\alpha \mathfrak{B} : H$, $\Gamma_2, x_1 : W_1, \ldots, x_n : W_n \vdash_\alpha N : A$ and $H@_\alpha A : F$. Hence $\Gamma_1, x_1 : V_1, \ldots, x_n : V_n \vdash_\alpha$

$\mathfrak{B} : \nabla \to G$. We can also prove that A is of the form F_1. By induction hypothesis, there exist $W'_1, \dots, W'_n$ such that $\Gamma_2, x_1 : W'_1, \dots, x_n : W'_n \vdash_\alpha N : \nabla$. Therefore, $\Gamma_1 \cap \Gamma_2, x_1 : V_1 \cap W'_1, \dots, x_n : V_n \cap W'_n \vdash_\alpha \mathfrak{B}N : G$.

- For $\dfrac{cM_1 \dots M_{m+1} \in \mathrm{CF}}{\mathrm{accu}_u(cM_1 \dots M_{m+1}, \epsilon)}$ with m arity of c: Then $n = 0$, $\alpha = u$ and there exist Γ_1, Γ_2 H and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_u cM_1 \dots M_m : H$, $\Gamma_2 \vdash_u M_{m+1} : A$ and $H @_\alpha A : F$. We can prove that H is of the form $c\vec{B}$ and $(c\vec{B}) @_\alpha A : G$. Therefore $\Gamma_1 \cap \Gamma_2 \vdash_u cM_1 \dots M_{m+1} : G$.
- For $\dfrac{\mathfrak{B}N \in \mathrm{CF}}{\mathrm{accu}_u(\mathfrak{B}N, \epsilon)}$ with N is of the form $\lambda x.M_1$, $\mathfrak{B}'$, $\mathrm{fix}(M_1)$ or $cN_1 \dots N_{m'}$ with m arity of c and $m' < m$: Then $\alpha = u$, $n = 0$ and there exist Γ_1, Γ_2, H and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_u \mathfrak{B} : H$, $\Gamma_2 \vdash_u N : A$ and $H @_\alpha A : F$. We can prove that A is of the form F_1. Therefore, F_1 is either of the form $B \to F_2$ or $F_1 = \delta$. Hence, $\Gamma_1 \vdash_u \mathfrak{B} : F_1 \to G$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_u \mathfrak{B}N : G$.
- For $\dfrac{\mathfrak{B}(c\vec{N}) \in \mathrm{CF}}{\mathrm{accu}_u(\mathfrak{B}(c\vec{N}), \epsilon)}$ with $\vec{N} = N_1 \dots N_m$, m arity of c, $\mathfrak{B} = (c_1 \vec{x_1}.M_1, \dots, c_n \vec{x_{m'}}.M_{m'})$ and for all i, $c_i \neq c$: Then $\alpha = u$, $n = 0$ and there exist Γ_1, Γ_2, H and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_u \mathfrak{B} : H$, $\Gamma_2 \vdash_u c\vec{N} : A$. We can prove that A is of the form F_1. Therefore, F_1 is of the form $c\vec{B}$. Hence, $\Gamma_1 \vdash_u \mathfrak{B} : c\vec{B} \to G$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_u \mathfrak{B}(c\vec{N}) : G$.

□

Lemma 43 (Typing normal forms)**.**
If M cannot be reduced by $\to_\alpha$, then there exist Γ and F such that $\Gamma \vdash_\alpha M : F$.

Proof. By induction on M.

Assume M cannot be reduced by $\to_\alpha$. By Lemma 36, we are in one of the following cases:

- M is of the form $\lambda x.M_1$: By induction hypothesis, there exist Γ and F such that $\Gamma \vdash_\alpha M_1 : F$. There exist Γ' and U such that $\Gamma = \Gamma', x : U$. Let $A := \nabla$ if $U = \omega$ and $A := U$ otherwise. Hence, $A \subseteq U$. Therefore $\Gamma' \vdash_\alpha \lambda x.M_1 : A \to F$.
- M is of the form $\mathfrak{B}$ with $\mathfrak{B} = (c_1 \vec{x_1}.M_1, \dots, c_n \vec{x_n}.M_n)$. By induction hypothesis, we can type each M_i. By a similar argument from the previous point we can type each $\lambda \vec{x_i}.M_i$. Therefore, there exist Γ such that $\Gamma \vdash_\alpha \mathfrak{B} : \nabla \to \nabla$.
- M is of the form $\mathrm{fix}(N)$: By induction hypothesis, there exist Γ and F such that $\Gamma \vdash_\alpha N : F$. Therefore $\Gamma \vdash_\alpha \mathrm{fix}(N) : \delta$.
- M is of the form $cM_1 \dots M_n$ with m arity of c and $n \leq m$: By induction hypothesis, there exist $\Gamma_1, \dots, \Gamma_n$, $F_1, \dots, F_n$ such that for all i, $\Gamma_i \vdash_\alpha M_i : F_i$. Let $F_{n+1} := \nabla, \dots, F_m := \nabla$. Therefore, there exist Γ such that $\Gamma \vdash_\alpha cM_1 \dots M_n : F_{n+1} \to \dots \to F_m \to cF_1 \dots F_m$.
- We have $\mathrm{accu}_\alpha(M)$: We adapt the proof of Lemma 42.

□

Theorem 29 (Subject expansion)**.**
Assume $\Gamma, x_1 : U_1, \dots, x_n : U_n \vdash_\alpha M' : A$ and $E = \{x_1, \dots, x_n\}$:

- *If $M \rightsquigarrow_{E,\alpha} M'$ then there exists Γ', U'_1, $\dots$, U'_n such that $\Gamma \approx \Gamma'$ and $\Gamma', x_1 : U'_1, \dots, x_n : U'_n \vdash_\alpha M : A$.*

- *If* $M \Rightarrow_{E,\alpha} M'$ *then there exists* Γ', U'_1, …, U'_n *and* B *such that* $\Gamma \approx \Gamma'$ *and* $\Gamma', x_1 : U'_1, \ldots, x_n : U'_n \vdash_\alpha M : B$.

Proof. First by induction on $M \rightsquigarrow_{E,\alpha} M'$ and on $M \Rightarrow_{E,\alpha} M'$, then by induction on A.

- If A is of the form $A_1 \cap A_2$: Then, there exist Γ_1, Γ_2, V_1, …, V_n, W_1, …, W_n such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $U_1 = V_1 \cap W_1$, …, $U_n = V_n \cap W_n$, $\Gamma_1, x_1 : V_1, \ldots, x_n : V_n \vdash_\alpha M' : A_1$ and $\Gamma_2, x_1 : W_1, \ldots, x_n : W_n \vdash_\alpha M' : A_2$.
  - For $M \rightsquigarrow_{E,\alpha} M'$: By induction hypothesis on $(M \rightsquigarrow_{E,\alpha} M', A_1)$ and $(M \rightsquigarrow_{E,\alpha} M', A_2)$, there exist Γ'_1, Γ'_2, V'_1, …, V'_n, W'_1, …, W'_n such that $\Gamma'_1, x_1 : V'_1, \ldots, x_n : V'_n \vdash_\alpha M : A_1$ and $\Gamma'_2, x_1 : W'_1, \ldots, x_n : W'_n \vdash_\alpha M : A_2$. Therefore $\Gamma'_1 \cap \Gamma'_2, x_1 : V'_1 \cap W'_1, \ldots, V'_n \cap W'_n \vdash_\alpha M : A_1 \cap A_2$ with $\Gamma \approx \Gamma'_1 \cap \Gamma'_2$.
  - For $M \Rightarrow_{E,\alpha} M'$: By induction hypothesis on $(M \Rightarrow_{E,\alpha} M', A_1)$ and $(M \Rightarrow_{E,\alpha} M', A_2)$, there exist Γ'_1, Γ'_2, V'_1, …, V'_n, W'_1, …, W'_n, B_1 and B_2 such that $\Gamma'_1, x_1 : V'_1, \ldots, x_n : V'_n \vdash_\alpha M : B_1$ and $\Gamma'_2, x_1 : W'_1, \ldots, x_n : W'_n \vdash_\alpha M : B_2$. Therefore $\Gamma'_1 \cap \Gamma'_2, x_1 : V'_1 \cap W'_1, \ldots, V'_n \cap W'_n \vdash_\alpha M : B_1 \cap B_2$ with $\Gamma \approx \Gamma'_1 \cap \Gamma'_2$.
- For $\dfrac{x \in \mathrm{fv}(M_1)}{(\lambda x.M_1)M_2 \rightsquigarrow_{\emptyset,\alpha} M_1\{x := M_2\}}$ with A is of the form F: Then $n = 0$. By Lemma 41, there exist Γ_1, Γ_2 and U such that $\Gamma \approx \Gamma_1 \cap \Gamma_2$, $\Gamma_1, x : U \vdash_\alpha M_1 : F$ and $\Gamma_2 \vdash_\alpha M_2 : U$. We have $x \in \mathrm{fv}(M_1) = \mathsf{Dom}(\Gamma_1, x : U)$. Therefore U is of the form B and $B \subseteq U$. Hence $\Gamma_1 \vdash_\alpha \lambda x.M : B \to F$. Therefore $\Gamma_1 \cap \Gamma_2 \vdash_\alpha (\lambda x.M)N : F$.
- For $\dfrac{x \notin \mathrm{fv}(M_1) \quad M_2 \not\longrightarrow_\alpha}{(\lambda x.M_1)M_2 \rightsquigarrow_{\mathrm{fv}(M_2),\alpha} M_1}$ with A is of the form F and $x \notin E$: Then $\mathrm{fv}(M_2) = \{x_1, \ldots, x_n\}$. By Lemma 43, there exist Γ_1 and B such that $\Gamma_1 \vdash_\alpha M_2 : B$. Since $E = \mathrm{fv}(M_2)$, there exist V_1, …, V_n such that $\Gamma_1 = x_1 : V_1, \ldots, x_n : V_n$. Since $x \notin E$ and $x \notin \mathrm{fv}(M_1)$, we have $\Gamma, x_1 : U_1, \ldots, x_n : U_n = \Gamma, x_1 : U_1, \ldots, x_n : U_n, x : \omega$. Therefore, $\Gamma, x_1 : U_1, \ldots, x_n : U_n \vdash_\alpha \lambda x.M_1 : B \to F$. Hence, $\Gamma, x_1 : U_1 \cap V_1, \ldots, x_n : U_n \cap V_n \vdash_\alpha (\lambda x.M_1)M_2 : F$.
- For $\dfrac{x \notin \mathrm{fv}(M_1) \quad M_2 \Rightarrow_{E,\alpha} M'_2}{(\lambda x.M_1)M2 \rightsquigarrow_{E,\alpha} (\lambda x.M_1)M'_2}$ with A is of the form F and $x \notin E$: There exist Γ_1, Γ_2, V_1, …, V_n, W_1, …, W_n and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $U_1 = V_1 \cap W_1$, …, $U_n = V_n \cap W_n$, $\Gamma_1, x_1 : V_1, \ldots, x_n : V_n \vdash_\alpha \lambda x.M_1 : B \to F$ and $\Gamma_2, x_1 : W_1, \ldots, x_n : W_n \vdash_\alpha M'_2 : B$. Since $x \notin E$, there exist U such that $B \subseteq U$ and $\Gamma_1, x_1 : V_1, \ldots, x_n : V_n, x : U \vdash_\alpha M_1 : F$. Since $x \notin \mathrm{fv}(M_1)$, we have $U = \omega$. By induction hypothesis, there exist Γ'_2, W'_1, …, W'_n and B' such that $\Gamma_2 \approx \Gamma'_2$ and $\Gamma'_2, x_1 : W'_1, \ldots, x_n : W'_n \vdash_\alpha M_2 : B'$. We have $B' \subseteq U$ because $\omega = U$. Hence $\Gamma_1, x_1 : V_1, \ldots, x_n : V_n \vdash_\alpha \lambda x.M_1 : B' \to F$. Therefore $\Gamma_1 \cap \Gamma'_2, x_1 : V_1 \cap W'_1, \ldots, x_n : V_n \cap W'_n \vdash_\alpha (\lambda x.M_1)M_2 : F$ with $\Gamma_1 \cap \Gamma_2 \approx \Gamma_1 \cap \Gamma'_2$.
- For the three rules who are applied to a matching redex, we adapt the rules for a simple β-redex.
- For $\dfrac{}{\mathrm{fix}(M_1)M_2 \rightsquigarrow_{\omega,\alpha} M_1 M_2 \mathrm{fix}(M_1)}$ with A is of the form F: Then $n = 0$. There exist Γ_1, Γ_2, G and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_\alpha M_1 M_2 : G$, $\Gamma_2 \vdash_\alpha \mathrm{fix}(M_1) : B$ and $G @_\alpha B : F$. There exist Γ_3, Γ_4, H and C such that $\Gamma_1 = \Gamma_3 \cap \Gamma_4$, $\Gamma_3 \vdash_\alpha M_1 : H$, $\Gamma_4 \vdash_\alpha M_2 : C$ and $H @_\alpha C : G$. Hence

$\Gamma_3 \cap \Gamma_2 \vdash_\alpha \mathrm{fix}(M_1) : C \to F$. Therefore $(\Gamma_3 \cap \Gamma_2) \cap \Gamma_4 \vdash_\alpha \mathrm{fix}(M_1)M_2 : F$ with $\Gamma \approx (\Gamma_3 \cap \Gamma_2) \cap \Gamma_4$.

- For the rules that propagate $\rightsquigarrow_\alpha$ and $\Rightarrow_\alpha$ with A is of the form F: Straightforward.
- For the rules that have $\rightsquigarrow_\alpha$ as their conclusion and $\Rightarrow_\alpha$ and $\mathrm{accu}_\alpha(,)$ as their premises with A is of the form A: We adapt the proof of Lemma 42.
- For $\dfrac{M \rightsquigarrow_{E,\alpha} M'}{M \Rightarrow_{E,\alpha} M'}$ and $\dfrac{M \in \mathrm{CF}}{M \rightsquigarrow_{\emptyset,u} M}$: Trivial.

□

Appendix B

A simple presentation of de Bruijn indices

B.1 Definitions

Assume we have a triplet (Var, O, S) such that :

- Var is a set, $O \in \mathrm{Var}$ and $S : \mathrm{Var} \to \mathrm{Var}$.
- $\forall x \in \mathrm{Var}, O \neq Sx$.
- $\forall x, y \in \mathrm{Var}, Sx = Sy \Rightarrow x = y$.
- $\forall x \in \mathrm{Var}, x = O \vee \exists y \in \mathrm{Var}, x = Sy$.

Definition 47 (Terms)**.**
We define Term the set of terms M with the following grammar:

$$\begin{array}{lll} M, N & ::= & x \mid MN \mid \lambda M \\ x & \in & Var \end{array}$$

Definition 48 (Free variables of a term)**.**
If $M \in Term$ we define $fv(M)$ a finite subset of Var by induction on M as follows:

$$\begin{array}{lcl} fv(x) & = & \{x\} \\ fv(MN) & = & fv(M) \cup fv(N) \\ fv(\lambda M) & = & \{x \in Var \mid S(x) \in fv(M)\} \end{array}$$

Definition 49 (Size of a term)**.**
If $M \in Term$ we define $|M|$ an integer defined by induction on M as follows:

$$\begin{array}{lcl} |x| & = & 1 \\ |MN| & = & |M| + |N| + 1 \\ |\lambda M| & = & |M| + 1 \end{array}$$

B.2 Binding

Definition 50 (Mapping)**.**
If $M \in Term$, then for all $f : Var \to Var$, we define $map(M, f)$ the term defined by induction on M as follows:

$$\begin{array}{lcl} map(x, f) & = & f(x) \\ map(MN, f) & = & map(M, f) map(N, f) \\ map(\lambda M, f) & = & \lambda map(M, f') \end{array}$$

with $f' : Var \to Var$ *the unique function such that :*

$$\begin{array}{lcl} f'(O) & = & O \\ f'(S(x)) & = & S(f(x)) \quad (\forall x \in Var) \end{array}$$

Theorem 31 (Size after a map)**.**
Assume $M \in Term$ *and* $f : Var \to Var$*. Then we have* $|map(M, f)| = |M|$*.*

Proof. By induction on M.

□

Definition 51 (Successor of a term)**.**
If $x \in Var$ *then* $map(x, S) = S(x)$*. Therefore, if* M *is a term, we can write* $S(M)$ *for* $map(M, S)$*.*

Definition 52 (Binding)**.**
If $M \in Term$*, then for all* $f : Var \to Term$*, we define* $bind(M, f)$ *the term defined by induction on* M *as follow:*

$$\begin{array}{lcl} bind(x, f) & = & f(x) \\ bind(MN, f) & = & bind(M, f)bind(N, f) \\ bind(\lambda M, f) & = & \lambda bind(M, f') \end{array}$$

with $f' : Var \to Term$ *the unique function such that :*

$$\begin{array}{lcl} f'(O) & = & O \\ f'(S(x)) & = & S(f(x)) \quad (\forall x \in Var) \end{array}$$

Theorem 32 (A mapping is a binding)**.**
Assume $M \in Term$ *and* $f : Var \to Var$*. Then* $f : Var \to Term$ *and* $bind(M, f) = map(M, f)$*.*

Proof. By induction on M.

□

Theorem 33 (Binding with identity)**.**
Assume $M \in Term$*, then* $id : Var \to Term$ *and* $bind(M, id) = M$*.*

Proof. By induction on M.

□

Theorem 34 (Compare two bindings)**.**
Assume $M \in Term$*. Then for all* $f, g : Var \to Term$*, if we have* $\forall x \in fv(M), f(x) = g(x)$ *then* $bind(M, f) = bind(M, g)$*.*

Proof. By induction on M.

□

Lemma 46. *Assume* $M \in Term$ *and* $f : Var \to Var$*. Then* $fv(map(M, f)) = \{f(x) \mid x \in fv(M)\}$*.*

Proof. By induction on M.

□

Corollary 4. $\forall M \in Term, fv(S(M)) = \{S(x) \mid x \in fv(M)\}$*.*

Theorem 35 (Free variables of a binding)**.**
Assume $M \in Term$ *and* $f : Var \to Term$*. Then:*

$$fv(bind(M, f)) = \bigcup_{x \in fv(M)} fv(f(x))$$

Proof. By induction on M.

□

Lemma 47. *Assume $M \in Term$, $f : Var \to Var$ and $g : Var \to Term$. Then $g \circ f : Var \to Term$ and $bind(map(M, f), g) = bind(M, g \circ f)$.*

Proof. By induction on M.

□

Lemma 48. *Assume $M \in Term$, $f : Var \to Term$ and $g : Var \to Var$. We write $h : Var \to Term$ defined by:*

$$\forall x \in Var, h(x) = map(f(x), g)$$

Then we have $map(bind(M, f), g) = bind(M, h)$.

Proof. By induction on M.

□

Theorem 36 (Composition of bindings)**.**

Assume $M \in Term$, $f : Var \to Term$, and $g : Var \to Term$. We write $h : Var \to Term$ defined by:

$$\forall x \in Var, h(x) = bind(f(x), g)$$

Then we have $bind(bind(M, f), g) = bind(M, h)$.

Proof. By induction on M.

□

B.3 Substitutions

Definition 53 (Substitution)**.**

Assume $M \in Term$, $x \in Var$ and $N \in Term$.
We write $M\{x := N\}$ for $bind(M, f)$ with $f : Var \to Term$ defined as follow:

$$\begin{array}{lcll} f(x) & = & N & \\ f(y) & = & y & (\forall y \neq x) \end{array}$$

Theorem 37 (Properties of substutions)**.**

Assume $M, N, M_1, M_2, N_1, N_2 \in Term$ and $x, y \in Var$. Then:

- *If $x \notin fv(M)$, then $M\{x := N\} = M$.*
- *If $x \in fv(M)$, then $fv(M\{x := N\}) = (fv(M) - \{x\}) \cup fv(N)$.*
- *$x\{x := N\} = N$.*
- *If $x \neq y$, then $y\{x := N\} = y$.*
- *$(M_1 M_2)\{x := N\} = M_1\{x := N\}M_2\{x := N\}$.*
- *$(\lambda M)\{x := N\} = \lambda M\{S(x) := S(N)\}$.*
- *$M\{x := x\} = M$.*
- *If $y \notin fv(M)$ then $M\{x := y\}\{y := N\} = M\{x := N\}$.*
- *$bind(M\{x := N\}, f) = bind(M, h)$ with $f, g : Var \to Term$, $g(x) = bind(N, f)$ and for all $y \in Var$, if $x \neq y$ then $g(y) = f(y)$.*
- *$bind(M\{x := N\}, f) = bind(M, f)\{f(x) := bind(N, f)\}$ with $f : Var \to Term$, $f(x) \in Var$ and for all $y \in fv(M)$, if $f(x) \in fv(f(y))$ then $x = y$.*
- *If $x \neq y$, and $x \notin fv(N_2)$ then $M\{x := N_1\}\{y := N_2\} = M\{y := N_2\}\{x := N_1\{y := N_2\}\}$.*
- *If $x \neq y$, $y \notin fv(N_1)$ and $x \notin fv(N_2)$ then $M\{x := N_1\}\{y := N_2\} = M\{y := N_2\}\{x := N_1\}$.*

Proof. Straightforward.

□

Definition 54 (Abstraction)**.**
Assume $x \in Var$ and $M \in Term$. We write $\lambda x.M$ for $\lambda bind(M, f)$ with $f : Var \to Term$ defined as follow:

$$\begin{array}{lcll} f(x) & = & O & \\ f(y) & = & S(y) & (\forall y \neq x) \end{array}$$

Theorem 38 (Properties of abstraction)**.**
Assume $M, N \in Term$ and $x, y \in Var$. Then:

- *$fv(\lambda x.M) = fv(M) - \{x\}$.*
- *$bind(\lambda x.M, f) = \lambda f(x).bind(M, f)$ if $f : Var \to Term$, $f(x) \in Var$, and for all $y \in fv(M)$, if $f(x) \in fv(f(y))$ then $x = y$.*
- *$(\lambda y.M)\{x := N\} = \lambda y.M\{x := N\}$ if $y \neq x$ and $y \notin fv(S(N)$.*
- *If $y \notin fv(M)$ then $\lambda x.M = \lambda y.M\{x := y\}$.*
- *If $\lambda x.M = \lambda y.N$ then $M\{x := y\} = N$ and if $x \neq y$ then $y \notin fv(M)$.*
- *If $\lambda x.M = \lambda x.M$ then $M = N$.*

Proof. Straightforward.

□

Definition 55 (Terms built with usual abstraction)**.**
Assume we have E an infinite subset of Term.
We write T_E the smallest subset of Term such that:

- *For all $x \in Var$, $x \in T_E$.*
- *For all $M, N \in T_E$, $MN \in T_E$.*
- *For all $x \in E$, $M \in T_E$, $\lambda x.M \in T_E$.*

Theorem 39 (Shapes of λ-terms)**.**
Assume E is an infinite subset of Var and $M \in Term$.
Then $M \in T_E$.

Proof. By induction on $|M|$.

□

Definition 56 (Equivalence between λ-terms)**.**
Assume M and N are terms.
For all $x, y \in Var$ we define $< x, y >$: $Var \to Var$ defined as follow:

$$\begin{array}{lcll} < x, y > (x) & = & y & \\ < x, y > (y) & = & x & \\ < x, y > (z) & = & z & (\forall z \notin \{x, y\}) \end{array}$$

We define $M =_\alpha N$ with the rules of figure B.1.

Theorem 40 (Relation between equality and α-equivalence)**.**
Assume $M, N \in Term$.
Then $M = N$ if and only if $M =_\alpha N$.

Proof. One way is by induction on $M \in T_{\text{Var}}$ and the other way is by induction on $M =_\alpha N$.

□

$$\frac{}{x =_\alpha x} \qquad \frac{M =_\alpha M' \quad N =_\alpha N'}{MN =_\alpha M'N'} \qquad \frac{M =_\alpha N}{\lambda x.M =_\alpha \lambda x.N}$$

$$\frac{x \neq y \quad y \notin \mathrm{fv}(M) \quad \mathrm{map}(M, < x, y >) =_\alpha N}{\lambda x.M =_\alpha \lambda y.N}$$

Figure B.1: α-equivalence

B.4 β-reduction

Definition 57 (β-rededuction)**.**
We define the β-reduction $\rightarrow_\beta$ with the congruence extension of the following rule:

$$(\lambda M)N \rightarrow \mathit{bind}(M, f)$$

with f defined as follow:

$$\begin{array}{lcll} f(0) & = & N & \\ f(S(x)) & = & x & (\forall x \in \mathit{Var}) \end{array}$$

Theorem 41 (β-reduction and abstraction)**.**
Assume $x \in \mathit{Var}$ and $M \in \mathit{Term}$. Then we have :

$$(\lambda x.M)N \rightarrow_\beta M\{x := N\}$$

Proof. Straightforward.

□

Theorem 42 (Free variables and β-reduction)**.**
Assume $M, M' \in \mathit{Term}$ and $M \rightarrow_\beta M'$ then $\mathit{fv}(M') \subseteq \mathit{fv}(M)$.

Proof. By induction on $M \rightarrow_\beta M'$.

□

Theorem 43 (β-reduction and binding)**.**
Assume $M, M' \in \mathit{Term}$, $f : \mathit{Var} \rightarrow \mathit{Term}$ and $M \rightarrow_\beta M'$. Then:

$$\mathit{bind}(M, f) \rightarrow_\beta \mathit{bind}(M', f)$$

Proof. By induction on $M \rightarrow_\beta M'$.

□

Corollary 5.
Assume $M, M', N \in \mathit{Term}$, $x \in \mathit{Var}$ and $M \rightarrow_\beta M'$. Then:

$$\begin{array}{lll} M\{x := N\} & \rightarrow_\beta & M'\{x := N\} \\ \lambda x.M & \rightarrow_\beta & \lambda x.M' \end{array}$$

Theorem 44 (β-reduction and mapping)**.**
Assume $M, M' \in \mathit{Term}$, $f : \mathit{Var} \rightarrow \mathit{Var}$ and $\mathit{map}(M, f) \rightarrow_\beta M'$.
Then, there exists $M'' \in \mathit{Term}$ such that $M \rightarrow_\beta M''$ and $\mathit{map}(M'', f) = M'$.

Proof. By induction on $M \rightarrow_\beta M'$.

□

Corollary 6. *Assume $M, M', N \in \mathit{Term}$ and $x, y \in \mathit{Var}$. Then:*

- *If $M\{x := y\} \rightarrow_\beta M'$ then there exists $M'' \in \mathit{Term}$ such that $M \rightarrow_\beta M''$ and $M' = M''\{x := y\}$.*

$$\frac{x \in E}{(\lambda x.M)N \to_E M\{x := N\}} \qquad \frac{M \to_E M'}{MN \to_E M'N}$$

$$\frac{N \to_E N'}{MN \to_E MN'} \qquad \frac{M \to_E M'}{\lambda x.M \to_E \lambda x.M'}$$

Figure B.2: Usual β-reduction

- *If* $\lambda x.M \to_\beta M'$ *then there exists* $M'' \in Term$ *such that* $M \to_\beta M''$ *and* $M' = \lambda x.M''$.

Definition 58 (Usual β-reduction)**.**
Assume E *is an infinite subset of* Var.
We define the reduction $\to_E$ *with the rules of figure B.2.*

Theorem 45 (Comparaison bewteen the two reductions)**.**
Assume $M, M' \in Term$ *and* E *is an infinite subset of* Var.
Then $M \to_E M'$ *if and only if* $M \to_\beta M'$.

Proof. One way is by induction on $M \to_E M'$ and the other way is by induction on $M \in T_E$.

□

B.5 Semantics

Assume we have a 4-uplet $(C, U, \text{inj}, \text{prj})$ such that:

- C is a cartesian closed category and $U : C$.
- $\text{inj} : (U \Rightarrow U) \to U : C$ and $\text{prj} : U \to (U \Rightarrow U)$ such that $\text{prj} \circ \text{inj} = \text{id}_{U \Rightarrow U}$.

Definition 59 (Environments)**.**
If $X : C$ *then we write* Env_X *the set of* ρ *such that for all* $x \in Var$, $\rho(x) : X \to U : C$.

Definition 60 (Interpretation of terms)**.**
Assume $X : C$, $\rho \in Env_X$, *and* $M \in Term$. *We define* $[M]_\rho : X \to U$ *by induction on* M *as follow:*

$$\begin{array}{lcl} [x]_\rho & = & \rho(x) \\ [MN]_\rho & = & ev \circ < prj \circ [M]_\rho, [N]_\rho > \\ [\lambda M]_\rho & = & inj \circ \Lambda([M]_{\rho'}) \end{array}$$

with $\rho' \in Env_{X \times U}$ *defined as follow:*

$$\begin{array}{lcl} \rho'(O) & = & \pi_2 \\ \rho'(S(x)) & = & \rho(x) \circ \pi_1 \quad (\forall x \in Var) \end{array}$$

Lemma 49. *Semantics of a mapping*
Assume $X, Y : C$, $\rho_1 \in Env_X$, $\rho_2 \in Env_Y$, $\varphi : X \to Y : C$, $f : Var \to Var$, *and* $M \in Term$ *such that for all* $x \in Var$, $\rho_1(f(x)) = \rho_2(x) \circ \varphi$.
Then $[map(M, f)]\rho_1 = [M]\rho_2 \circ \varphi$.

Proof. By induction on M.

□

Corollary 7. *Assume $X : C$, $\rho \in Env_X$ and $M \in Term$. Then $[S(M)]\rho' = [M]\rho \circ \pi_1$ with $\rho' \in Env_{X \times U}$ defined as follow:*

$$\begin{array}{lcll} \rho'(O) & = & \pi_2 & \\ \rho'(S(x)) & = & \rho(x) \circ \pi_1 & (\forall x \in Var) \end{array}$$

Theorem 46 (Semantics of a binding)**.**
Assume $X, Y : C$, $\rho_1 \in Env_X$, $\rho_2 \in Env_Y$, $\varphi : X \to Y : C$, $f : Var \to Term$, and $M \in Term$ such that for all $x \in fv(M)$, $[f(x)]_{\rho_1} = \rho_2(x) \circ \varphi$.
Then $[bind(M, f)]_{\rho_1} = [M]_{\rho_2} \circ \varphi$.

Proof. By induction on M.

□

Corollary 8. *Assume $X, Y : C$, $\rho \in Env_Y$, $\varphi : X \to Y : C$, $M \in Term$. Then $[M]_{\rho'} = [M]_\rho \circ \varphi$ with $\rho' \in Env_X$ defined by $\rho'(x) = \rho(x) \circ \varphi$.*

Theorem 47 (Soundness)**.**
Assume $X : C$, $\rho \in Env_X$, $M, N \in Term$ and $M \to_\beta N$. Then $[M]_\rho = [N]_\rho$.

Proof. By induction on $M \to_\beta N$.

□

Theorem 48 (Semantics of a substitution)**.**
Assume $X : C$, $\rho \in Env_X$, $x \in Var$ and $M, N \in Term$. Then $[M\{x := N\}]_\rho = [M]_{\rho'}$ with $\rho' \in Env_X$ defined as follow:

$$\begin{array}{lcll} \rho'(x) & = & [N]_\rho & \\ \rho'(y) & = & \rho(y) & (\forall y \neq x) \end{array}$$

Proof. Straightforward.

□

Theorem 49 (Coherence)**.**

- *If U is final then $U \approx 1$.*
- *If $\pi_1 = \pi_2 (rel. U \times U)$ then U is final.*
- *Assume $x, y \in Var$, $x \neq y$, and for all $X : C$ and $\rho \in Env_X$, $[x]_\rho = [y]_\rho$. Then U is final.*
- *If there exists $X : C$ and $\rho \in Env_X$ such that $[\lambda\lambda O]_\rho = [\lambda\lambda S(O)]_\rho$, then U is final.*

Proof. Straightforward

Theorem 50 (Free variables and semantics)**.**
Assume $X : C$, $\rho_1, \rho_2 \in Env_X$, $M \in Term$ and for all $x \in fv(M)$, $\rho_1(x) = \rho_2(x)$. Then $[M]_{\rho_1} = [M]_{\rho_2}$.

Proof. Straightforward