Faculty of Science and Mathematics
University of Novi Sad

Silvia Gilezan

# Intersection types in lambda calculus and logic

Novi Sad, 1993

For my parents

# Contents

# Introduction

The *lambda calculus* and the *theory of cominators* are two formal systems developed by Church, Curry and Schönfinkel, independently, in the 1920's and 1930's (see Church, 1941, and Schönfinkel, 1924). Later on they were devolped in Curry et al., 1958. They are based on similar ideas to provide a basis for the foundation of mathematics and logic. The whole system turned to be inconsistent, as shown by Kleene and Rosser. The part considering functions became of interest for further investigations since computable functions are exactly the lambda representable functions. Church and Turing proved the equivalence of the notions: a *numerical function* is recursive (Gödel); computable on a Turing machine (Turing); definable in the untyped lambda calculus (Church).

**Untyped lambda calculus.** The two basic operations of the lambda calculus are *application* and *abstraction*. Application of a function $F$ to an argument $A$ is denoted by $FA$. It is possible to have self-application $FF$ as well. Abstraction provides a function with an argument $x$, denoted by $\lambda x.M(x)$, from a term $M(x)$ which depends on $x$.

The lambda calculus is a formal system that is meant to deal with functions and constructions of new functions. A function $F$ applied to an argument $A$ yields $FA$ which is an object, or a function.

Schönfinkel's idea of introducing combinators is originated in the idea to represent functions without the use of bound variables (see Schönfinkel, 1924). More general, the role of any rule in mathematics and logic is not to determine the properties of variables in it, but to determine the properties of operations or logical connectives in it. Therefore variables are just tools used to express the required properties. That was the reason to introduce a functional calculus where functions cannot be only arguments, but values as well.

Models of the untyped lambda calculus were not really known up to the 1970's. The main problem was to interpret objects that are arguments and functions to be applied to these arguments at the same time. The domain $D$

of the semantics for the lambda calculus has to be isomorphic to its function space $D \rightarrow D$. This is impossible by Cantor's theorem. The problem was overcome by Scott (see Scott, 1980) by restricting $D \rightarrow D$ to the set of continuous functions with respect to a certain topology (the so called Scott topology) on $\mathcal{D}$. This first model of the untyped lambda calculus is called $D_\infty$. Later on various lambda models have been developed (see Barendregt, 1984). Lambda models can be given by a first order definition, but there are syntactical and categorical descriptions of these classes as well.

Illative combinatory logic was the full system of Church and was at first inconsistent, as already mentioned. The original idea to base logic on a consistent system of lambda terms or combinators was due to Church and Curry. In Barendregt et al., 1992, appropriate versions of illative combinatory logic were shown to be sound and complete with respect to the first order propositional and predicate logic.

**Typed lambda calcului.** A *type assignment system* assigns formulae (types) of a certain language to some lambda terms in order to specify the properties of these lambda terms. There are two basically different ways of formulating type assignment systems of typed lambda calculi. These differences are traced from the original approaches of Curry and Church. In the systems given *à la Curry*, it is possible to assign infinitely many types to each term variable. So it is with terms; i.e., if a term has a type, then it has infinitely many types. In the systems given *à la Church*, exactly one type is assigned to each term variable (term). The difference between these two ways of giving a type assignment system is clarified in Barendregt, 1992.

The basic type assignment system is the *simply typed lambda calculus*, sometimes called Curry's type inference system. It can be given in both ways. The only type forming operator is the *arrow*. Arrow types are assigned to functions and hence the application of lambda terms yields the arrow elimination on types, while the abstraction–(function construction) yields the arrow introduction.

Much of the untyped theory is in accordance with the introduction of arrow types. One of the important properties of the simply typed lambda calculus is that self–application is not typable, i.e., lambda terms containing selfapplication have no type.

The simply typed lambda calculus can be extended in various ways. The extensions of its Church version form *Barendregt's cube.* The *second-order lambda calculus* $\lambda 2$ can also be given in both ways, while it is not clear up to now how to obtain the Curry version of the *theory of constructions* $\lambda C$.

Intersection types had been introduced in Coppo et al., 1980, and Barendregt et al., 1983. They are introduced as a generalization of Curry's type inference system, in order to characterize a larger class of terms. The main idea is the introduction of a new type-forming operator, the intersection $\cap$.

The types of the *lambda calculus with intersection types* $\lambda\cap$ are propositional formulae with connectives $\rightarrow$ and $\cap$ , where $\cap$ is a *specific conjunction* whose properties are in accordance with its interpretation as intersection of types. The basic notions of the lambda calculus with intersection types, also called the intersection type assignment system, or Torino system, are given in Coppo et al., 1981, Barendregt et al., 1983, and can be found in the survey of typed lambda calculi in Barendregt, 1992. Up to now there are only Curry versions of the systems with intersection types.

**Lambda calculus and logic.** An interesting approach to type systems is the "*Curry-Howard isomorphism*" or *formulae-as-types interpretation.* This idea of connecting inferences in the typed systems with deductions in logical systems can be found already in Curry et al., 1958. Later in the 1960's it is developed by de Brujin, Howard and Lambek. A correspondence between constructive proofs of logical formulae and lambda terms (or combinators) of related types and conversely is established in Howard, (1969)1980. By this connection the simply typed lambda calculus is the internal language the cartesian closed categories as shown in Lambek et al., 1986.

This enables on the one hand to provide a logical meaning of type constructors, on the other logical systems can be seen in a computational way.

The simply typed lambda calculus corresponds to the implicational fragment of intuitionistic propositional logic in the following way:

- arrow-types correspond to implicational formulae;
- the elimination of an arrow-type (application) corresponds to the elim-

ination rule of implication (Modus Ponens) in logic;

- the introduction of an arrow–type (abstraction) corresponds to the introduction rule of implication (Deduction Theorem) in logic.

The simply typed lambda calculus has been extended in order to correspond to intuitionistic propositional logic with all its connectives. In this approach all inhabited types are exactly all provable formulae (see Howard, 1980). The simply typed lambda calculus can be restricted in order to present inferences in substructural logics (see Došen, 1988, Gilezan, 1988).

The Curry–Howard isomorphism of proofs as terms brings to life Brouwer's idea of a constructive proof.

All the systems of Barendregt's cube correspond by the Curry–Howard isomorphism to some constructive logic. Second–order or polymorphic lambda calculus with quantification over types corresponds to second–order propositional logic. Girard's system $F_\omega$ corresponds to higher–order propositional logic, for more details see Barendregt, 1992, Girard et al., 1991, and Geuvers, 1988.

The Curry-Howard isomorphism between formulae as types and terms as proofs does not hold for the intersection type systems. It fails since the terms do not "code" the proofs anymore. The rules of intersection elimination and intersection introduction do not change the lambda term, i.e., the lambda term is the same in the premises and in the conclusion in both of these rules. Thus in these two rules the lambda term remains the same, although the deduction grows, and hence it does not correspond to the deduction.

Some problems of interest in the systems given à la Curry are:

- *Type checking* in the system. (Is it possible to assign a given type to a given term, $M : \sigma$?).
- *Typability* in the system. (Is there a type that can be assigned to a given term, $M :?$).
- *Inhabitation* in the system. (Is there a term of a given type, $? : \sigma$).

All three problems are decidable in the simply typed lambda calculus, see Barendregt, 1992. There is an overview of these problems in all systems of Barendregt's cube.

**Lambda calculus and computer science.** Turing machines, recursive functions and the lambda calculus are models of computation of the same power. *Church's thesis* assrets that the effectively computable functions are precisely those functions that can be computed in the lambda calculus. Programming languages, denotational semantics and term rewriting systems based on lambda calculus are another wide research area. The implementations of programming languages ALGOL 68, ALPHARD, AUTOMATH, LISP, MIRANDA, ML, MODEL, PASCAL, RUSSELL, SML are founded in the lambda calculus theory. This area is out of the scope of the presented work.

**Lambda calculus, linguistics, and literature.** We shall mentioned the presence of the lambda calculus in these topics, although they are far beyond the scope of the presented work. Montegue brought the full power of lambda calculus to bear on the semantics of natural languages (see Montegue, 1979). There is a brief overview of the lambda calculus in Penrose, 1992, in the discussion of the limits of computing machines. Combinators of the combinatory logic and lambda calculus appear as birds in an interesting presentation of this subject in Smullyan, 1985.

**Short overview of the presented work.** We consider four intersection type assignment systems. Most of our work is in the syntax of the intersection type assignment systems and their application in the untyped lambda calculus.

**Chapter 1** is a brief overview of the untyped lambda calculus, theory of combinators and intersection type assignment systems.

**Chapter 2** deals with typability in the intersection type assignment systems. The problem of typability in the full intersection type assignment system $\lambda\cap$ is trivial, since every lambda term is typable by a special type denoted with $\omega$. This property changes essentially when the $(\omega)$–rule is left out.

It turns out that all strongly normalizing lambda terms are typable in these systems and that they are the only terms typable in these systems. The idea that strongly normalizing lambda terms are exactly the terms typable in the intersection type assignment systems without the $(\omega)$–rule first appeared in Pottinger, 1980, Coppo et al., 1981, and Leivant, 1983. Further, this subject is treated in Ronchi et al., 1984, Krivine, 1990, and van Bakel, 1992, with different approaches. We shall present a modified proof of this property and compare it with the proofs mentioned above. Moreover we discuss the fact that undecidability of typability is a consequence of this property.

Chapter 3 deals with the inhabitation in the intersection and union type assignment systems. These systems are introduced in Barbanera et al., 1991, as extensions of the intersection type systems with union types and corresponding inference rules. Decidability of inhabitation in the intersection type systems is still an open problem. There are various, mainly proof–thoretic approaches toward this problem in Pottinger, 1980, Lopez–Escobar, 1985, Mints, 1989, Alessi et al., 1991, and Venneri, 1992. We consider inhabitation versus provability in intuitionistic propositional logic with conjunction and disjunction. Inhabitation implies provability. We give a semantical explanation why the converse does not hold. Our contribution to the attempts mentioned above is the link between the inhabitation in the intersection and union type assignment systems and the inhabitation in the extension of the simply typed lambda calculus with conjunctive and disjunctive types.

Chapter 4 deals with the applicaton of properties of the intersection type assignment systems in proofs of properties of the untyped lambda calculus. We prove the finitness of developments property using the simply typed lambda calculus. Genericity Lemma is proved by using the intersection type assignment systems. A new topology on lambda terms is introduced using typability in the intersection type systems. Applicaton appears to be continuous with respect to this new topology. We show that the introduced topology and the filter topology which is introduced in Barendregt et al., 1983, are the same.

**Acknowledgement.** In the first place, I would like to thank Dr. Kosta Došen and Professor Henk Barendregt. Both of them have had a great influence on my work and life. Kosta Došen introduced and opened up the entire world of logic to me, encouraging me to turn my attention to the field of lambda calculus and its application. Henk Barendregt provided a creative atmosphere initiating inspiring discussion on how to live science and how to investigate life.

I am very grateful to Mariangiola Dezani-Cinacaglini and Betti Venneri for their encouragements, suggestions and remarks. I also wish to thank Jim Lambek for his support and involvement in my work.

Many thanks to Milan Grulović, Gradimir Vojvodić and Branislav Boričić for their interest in my work.

The work with Marijana Perišić-Tabaković on the improvement of the English text was a learning experience as well as a pleasure. Aleksandra Djan did most of the typing and Zoran Ovcin did most of the editing helping me getting this text into its LaTeX form.

The Department of Applied Fundamental Disciplines, University of Novi Sad, was helpful in supporting all of my trips and leaves, both professional ones as well as spiritual ones providing also the necessary financial support. I am also grateful to the Catholic University in Nijmegen, University of Turin and McGill University in Montreal for their hospitality.

My friends and family encouraged my way in the "...safe and secure world of mathematics in which nothing could go wrong, no trouble arise, no bridges collapse! So different from the world of 1939..."[1] and 1993. I wish to express my gratitude to all of them.

[1]Hodges, A., Alan Turing; The Enigma, Barnet Books, London, 1983.

# 1 Intersection type assignment systems

In this chapter we shall give a short overview of some basic notions of the untyped and typed lambda calculus. A comprehensive study of the syntax and semantics of the untyped lambda calculus is in Barendregt, 1984. The overview of typed lambda caculi is given in Barendregt, 1992. Section 1.1 and Section 1.2 contain the untyped theory of lambda calculus and combinators, respectively. Simple types and intersection types, as well as corresponding type systems are given in Section 1.3.

## 1.1 Untyped lambda calculus

The two basic operations of the lambda calculus are the *application* and the *abstraction*. The application of a function $F$ to an argument $A$ is denoted by $FA$. It is possible to have selfapplication $FF$ as well. The abstraction provides a function with an argument $x$, denoted by $\lambda x.M(x)$, from a term $M(x)$ which depends on $x$. In the following intuitive example

$$(\lambda x.x^2 + 1)3 = 3^2 + 1,$$

$\lambda x.x^2 + 1$ is a function that maps $x$ into $x^2 + 1$. If it is applied to 3 the result obtained is 10. More generally

$$(\lambda x.M(x))N = M(N),$$

and this equality, called *$\beta$-conversion*, is one of the basic axioms of the lambda calculus.

The lambda calculus is a formal system that is meant to deal with functions and constructions of new functions. A function $F$ applied to an argument $A$ yields $FA$ which is an object, or a function.

For example, the expression "$x - y$" can be seen as a function of two variables $x$ and $y$,

$$H(x, y) = x - y.$$

But the expression "$x - y$" can be seen as a function $F$ of a variable $x$ or as a function $G$ of a variable $y$. These two functions can be distinguished by introducing the symbol "$\lambda$" in the following way:

$$F = \lambda x.x - y \qquad G = \lambda y.x - y.$$

The functions $F = \lambda x.x - y$ and $G = \lambda y.x - y$ both abstracted from the expression "$x - y$" can be applied to an argument, say 0, in order to obtain $F0 = (\lambda x.x - y)0 = -y$ and $G0 = (\lambda y.x - y)0 = x$. These are examples how to construct functions of a variable $x$ and $y$, respectively, from an expression that contains "$x$" and "$y$".

The functions of multiple variables can be avoided by functions whose values are functions. Therefore in our example $H$ can be represented by a function $H^*$ given by

$$H^* = \lambda x.(\lambda y.x - y).$$

Then

$$H^*a = \lambda y.a - y$$

is a function of a variable $y$. For any two arguments $a$ and $b$

$$(H^*a)b = (\lambda y.a - y)b = a - b.$$

This is Schönfinkel's idea (see Scönfinkel, 1924) of representing functions

$$H : D_1 \times D_2 \rightarrow D_3$$

by functions

$$H^* : D_1 \rightarrow (D_2 \rightarrow D_3).$$

It is further developed by Curry (see Curry et al., 1958) and it is called *currying.*

The *alphabet* of lambda calculus consists of:

- a denumerable set of variables $\mathcal{V} = \{x, y, z, x_1, \ldots\}$;
- a denumerable set of constants $C = \{c, d, e, c_1, \ldots\}$;
- an operation of application $\cdot$;

- an operation of abstraction $\lambda x.$;
- auxiliarly symbols $), ($ .

The set of lambda terms $\Lambda$ is defined in the following way:

**Definition 1.1.1.**

*(i)* $\mathcal{V} \subseteq \Lambda$, $C \subseteq \Lambda$.

*(ii) If* $M, N \in \Lambda$, *then* $(MN) \in \Lambda$.

*(iii) If* $M \in \Lambda$ *and* $x \in \mathcal{V}$, *then* $\lambda x.M \in \Lambda$.

$M, N, P, Q, R, M_1, \ldots$ are schematic letters for lambda terms. The following abbreviations are usual

$$\lambda x_1 x_2 . x_n . M \equiv (\lambda x_1 . (\lambda x_2 . \ldots (\lambda x_n . M),$$
$$M_1 M_2 \ldots M_k \equiv (\ldots (M_1 M_2) \ldots M_k),$$

where $\equiv$ is the syntactical identity between terms.

**Definition 1.1.2.** The notion of a subterm is defined in the folowing way:

*(i)* $M$ *is a subterm of itself.*

*(ii) If* $MN$ *is a subterm of* $P$, *then both* $M$ *and* $N$ *are subterms of* $P$.

*(iii) If* $\lambda x.M$ *is a subterm of* $P$, *then* $M$ *is a subterm of* $P$.

Some elementary lambda terms we shall become very familiar with are

$$\mathsf{I} \equiv \lambda x.x, \quad \mathsf{K} \equiv \lambda xy.x, \quad \mathsf{S} \equiv \lambda xyz.xz(yz),$$
$$\mathsf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)),$$
$$\Omega \equiv (\lambda x.xx)(\lambda x.xx),$$
$$\lambda x.xx.$$

There is a historical background of their names I,K,S,... which dates from the theory of combinators.

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

**Definition 1.1.3.**

*(i)* $Fv(x) = \{x\}$.

*(ii)* $Fv(MN) = Fv(M) \cup Fv(N)$.

*(iii)* $Fv(\lambda x.M) = Fv(M) \backslash \{x\}$.

$M(x)$ denotes that $x \in Fv(M)$. The term $M$ is the domain of the abstractor $\lambda x$ in the term $\lambda x.M$. The term $M$ is closed if the set $Fv(M)$ is empty. The abstraction $\lambda x$ defined in this way binds the variable like it does $\forall x$ in the predicate calculus or $\int_a^b \dots dx$ in the integral calculus. A variable $x$ is *free* in $M$ if $x \in Fv(M)$; it can have more than one occurrence in $M$.

The *substitution instance* $M[N/x]$ is obtained from $M$ by replacing each occurrence of the variable $x$ by the term $N$ such that for every $y \in Fv(N)$ there is no subterm of $M$ of the form $\lambda y.P$, for which $x \in Fv(\lambda y.P)$. It can be formalized in the following way:

**Definition 1.1.4.**

*(i)* $x \notin Fv(M)$, *then* $M[N/x] \equiv M$.

*(ii)* $x \in Fv(M)$, *then*

- *if* $M \equiv x$, *then* $M[N/x] \equiv N$,
- *if* $M \equiv PQ$, *then* $M[N/x] \equiv P[N/x]Q[N/x]$,
- *if* $M \equiv \lambda y.P$, *then*

$$M[N/x] \equiv \begin{cases} \lambda y.P[N/x] & , \ \textit{if } y \notin Fv(N), \\ \lambda z.P[z/y][N/x] & , \ \textit{if } y \in Fv(N), \\ & \quad z \notin Fv(P), z \notin Fv(N). \end{cases}$$

**Definition 1.1.5.** The axiom–schemes and the rules of the lambda calculus

1. $\lambda x.M = \lambda y.M[y/x], \quad y \notin Fv(M)$ *($\alpha$-conversion);*
2. $(\lambda x.M)N = M[N/x]$ *($\beta$-conversion);*
3. $M = M$;
4. $\dfrac{M = N}{N = M}$;
5. $\dfrac{M = N \quad N = P}{M = P}$;
6. $\dfrac{M = N \quad P = Q}{MP = NQ}$;
7. $\dfrac{M = N}{\lambda x.M = \lambda x.N}$ *($\xi$-rule);*
8. $\lambda x.Mx = M, x \notin Fv(M)$ *($\eta$-conversion).*

The first seven axiom–schemes and rules do not provide the extensional equality of functions

$$\text{(ext)} \quad F = G \text{ iff } Fx = Gx, \text{ for all } x.$$

There are terms $M$ and $N$ such that $MP = NP$ for all lambda terms $P$, but one cannot derive their equality. Such terms are, for example

$$M \equiv \lambda x.yx \text{ and } N \equiv y,$$

since

$$MP \equiv (\lambda x.yx)P = yP \equiv NP.$$

The equality given by (1)–(7) is the original lambda equality in Church, 1941, and Hindley et al., 1972. The lambda calculus with this equality is called the calculus of $\beta$–conversion in Curry et al., 1958.

The extensional equality between lambda terms is obtained by adding the axiom scheme (8) or the following rule

$$\text{(ext)}' \text{ if } FM = GM \text{ for all lambda terms } M, \text{ then } F = G.$$

The equality given with (1)–(8) is called the $\beta\eta$–conversion in Curry et al., 1958, extensional equality in Hindley et al., 1972. In Stenlund, 1972, the notion of lambda equality is the equality with extensionality.

Each of these axiom–schemes and rules determines a property of the formalism of the lambda calculus:

(1), the principle of $\alpha$–conversion about the renaming of variables that are not free, points that the abstraction is an operation which binds variables;

(2), the principle of $\beta$–conversion is a "calculating" property which links the application and abstraction;

(3)–(6), imply that " $=$ " is a congruence with respect to the application;

(7), the $\xi$–rule provides the extensionality of abstraction;

(8), the principle of $\eta$–conversion is an "ontological" property how to create a function from any object.

Taken together (7) and (8) provide the extensionality of all functions. The relations of $\beta$–reduction and $\beta\eta$–reduction are beside the equality between terms important relations of the lambda calculus. A lambda term of the form

$$(\lambda x.P)Q \quad \text{and} \quad \lambda x.Px, \quad x \notin Fv(P)$$

are called *redex* and $\eta$–*redex*, respectively. A term of the form

$$P[Q/x] \quad \text{and} \quad P$$

is its *contractum*, respectively. The notation of these *reductions*, elsewhere called *contractions*, is

$$(\lambda x.P)Q \to_\beta P[Q/x] \text{ and } \lambda x.Px \to_\eta P, x \notin Fv(P).$$

A term $M$ is $\beta$–, $\beta\eta$–reduced to $N$, notation $M \twoheadrightarrow N$ and $M \twoheadrightarrow_{\beta\eta} N$ if there is a sequence of terms $M \equiv M_1, \ldots, M_k \equiv N$ such that for all $i$ $M_i \to_\beta M_{i+1}$, and $M_i \to_{\beta\eta} M_{i+1}$, respectively. *Expansion* is the inverse of reduction, i.e., in the previous case we say that $M$ is an expansion ($\beta\eta$–expansion) of $N$. If $M \in \Lambda$ is a lambda term the *redexes of M* are defined inductively.

**Definition 1.1.6.**

*(i) If $M$ is a variable, then $M$ has no redex.*

*(ii) If $M \equiv \lambda x.N$, then the redexes of $M$ are the redexes of $N$.*

*(iii) If $M \equiv PQ$, then the redexes of $M$ are the redexes of $P$, the redexes of $Q$ and $M$, as well, if $P$ starts with $\lambda$.*

A lambda term $N$ is a *normal form* if $N$ does not have any redex. A term $M$ *has a normal form* (is *normalizing*) if there is a normal form $N$ such that $M \twoheadrightarrow N$. A term $M$ is called *strongly normalizing* if all the reduction paths starting with $M$ are finite. All normal forms have the following form

$$\lambda y_1 \dots y_n.zN_1 \dots N_k,$$

where $N_i$, $1 \leq i \leq k$, $0 \leq k$, are again normal forms and $z$ can be one of the $y_j$, $1 \leq j \leq n$, $0 \leq n$.

A term is a *head normal form* if it is of the form $\lambda x_1 \dots x_n.yM_1 \dots M_l$, where $y$ can be one of the $x_i$, $1 \leq i \leq n$, and $M_j \in \Lambda$, $1 \leq j \leq l$, $0 \leq l$. A term *has a head normal form* (is *solvable*) if it can be reduced to a head normal form. Terms which do not have head normal forms are called *unsolvable* terms.

Normal forms are $\mathsf{I} \equiv \lambda x.x$, $\mathsf{K} \equiv \lambda xy.x$, etc. The term $\mathsf{KIS}$ is strongly normalizing. The term $\mathsf{KI}\Omega$ can be reduced in different ways for example

$$\begin{aligned}
&\mathsf{KI}\Omega \to_\beta \dots \to_\beta \mathsf{KI}\Omega \to_\beta \dots,\\
&\mathsf{KI}\Omega \to_\beta \mathsf{I},\\
&\mathsf{KI}\Omega \to_\beta \mathsf{KI}\Omega \to_\beta \dots \to_\beta \mathsf{KI}\Omega \to_\beta I.
\end{aligned}$$

Hence, $\mathsf{KI}\Omega$ has a normal form $\mathsf{I}$, but it is not strongly normalizing.

The term Y has no normal form, but it has a head normal form, i.e. is solvable, since

$$\lambda f.((\lambda x.f(xx))(\lambda x.f(xx)) \to_\beta \lambda f.f((\lambda x.f(xx))(\lambda x.f(xx)) \to_\beta \ldots$$

$$\ldots \to_\beta \lambda f.ff\ldots f((\lambda x.f(xx))(\lambda x.f(xx))) \to_\beta \ldots .$$

The term $\Omega$ has neither a normal form, nor a head normal form, i.e. is unsolvable, since

$$\Omega \equiv (\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \ldots \to_\beta \Omega \to_\beta \ldots .$$

and this is the only possible reduction path.

The explicit connection between reduction and equality is given in the following way:

**Proposition 1.1.7.** *Let $M, N \in \Lambda$. $M = N$ if and only if there is a sequence $M \equiv M_1, \ldots, M_n \equiv N$, $n \geq 1$, such that $M_i \to_\beta M_{i+1}$ or $M_{i+1} \to_\beta M_i$ for each $1 \leq i \leq n$.*

Hence, the equality is the symmetric closure of the reduction relation. A reducible lambda term can usually be reduced in various ways. An interesting question is whether the process of reduction is consistent, i.e., whether different reductions can reduce a lambda term to a unique term or not. The affirmative answer to this question is given by the Church–Rosser Theorem.

**Theorem 1.1.8.** *(Church–Rosser)*
*Let $M \in \Lambda$. If $M \twoheadrightarrow N_1$ and $M \twoheadrightarrow N_2$, then there is a term $S \in \Lambda$ such that $N_1 \twoheadrightarrow S$ and $N_2 \twoheadrightarrow S$, i.e.,*

$$\begin{tikzcd} & M \arrow[dl] \arrow[dr] & \\ N_1 \arrow[dr] & & N_2. \arrow[dl] \\ & S & \end{tikzcd}$$

Corollary 1.1.9.

*(i) A term has at most one normal form.*

*(ii) If $M = N$, then there is a lambda term $S$ such that $M \twoheadrightarrow S$ and $N \twoheadrightarrow S$.*

*(iii) If $M$ and $N$ are normalizing and $M = N$, then they have the same normal form.*

The given lambda calculus is called $\lambda$–K–calculus. The lambda calculus considered by Church (see Church, 1941), called $\lambda$–I– calculus, has a different notion of abstraction. The condition (iii) in Definition 1.1.1 has an additional request, namely it is of the form

(iii') If $M \in \Lambda$ and $x \in Fv(M)$, then $\lambda x.M \in \Lambda$.

Hence, there is no void lambda abstraction. As we saw above it can happen in the $\lambda$–K–calculus that a term has a normal form although this is not the case with some of its subterms; e.g. $\mathsf{KI}\Omega \to_\beta \mathsf{I}$ while $\Omega$ has no normal form. This is avoided in the $\lambda$–I–calculus.

A very basic result in the lambda calculus is the *Fixed* point theorem, due to the existance of the *fixed* point lambda term (combinator) $\mathsf{Y} \equiv \lambda f.(\lambda x.f(xx))\ (\lambda x.f(xx))$ which applied to any function $F$ gives the *fixed* point of $F$, i.e.,

$$\forall F \quad F(\mathsf{Y}F) = \mathsf{Y}F.$$

## 1.2 Theory of combinators

The lambda calculus and the theory of combinators, two formal systems developed about the same time, are based on similar ideas to provide a basis for the foundation of mathematics and logic. Although most of our work will be in the untyped and the typed lambda calculus we shall recall some basic notions of the theory of combinators as well.

Schönfinkel's idea of introducing combinators originated in the idea to represent functions without the use of variables (see Schönfinkel, 1924). The following examples gives a good intuition of this idea. In arithmetics the commutative rule of addition can be represented in the following way:

$$\text{for all } x \text{ and } y \text{ one has, } \quad x + y = y + x.$$

This rule can be expressed without the use of bound variables. Let us define

$$A(x, y) = x + y, \text{ for all } x \text{ and } y,$$

and let us introduce an operator–combinator $\mathsf{C}$

$$(\mathsf{C}z)(x, y) = z(y, x), \text{ for all } x, y \text{ and } z.$$

Then the given rule has the following form

$$\mathsf{C}A = A,$$

and hence it is given without variables. As an other example let us consider the tautology $\vdash (\alpha \to \beta) \to (\neg\beta \to \neg\alpha)$. This not a property of the variables "$\alpha$" and "$\beta$", but it expresses a certain relation between the logical connectives $\to$ and $\neg$.

More generally, the role of any rule in mathematics and logic is not to determine the properties of variables in it, but to determine the properties of operations or logical connectives in it. Therefore variables are just tools used to express the required properties. That was the reason to introduce a functional calculus where functions would not be only arguments, but values as well.

The *alphabet* of the theory of combinators consists of:

- a denumerable set of variables $V = \{x, y, z, x_1, \ldots\}$;
- a denumerable set of constants $C = \{c, d, e, c_1, \ldots\}$;
- a set of basic combinators $\mathcal{B} = \{\mathsf{I}, \mathsf{K}, \mathsf{S}\}$;
- an operation of application $\cdot$;

- auxiliary symbols ), (.

The set of all $c$–terms $\mathcal{C}$ is defined in the following way:

**Definition 1.2.1.**

(i) $\mathcal{V} \subseteq \mathcal{C}$, $C \subseteq \mathcal{C}$, $\mathcal{B} \subseteq \mathcal{C}$.

(ii) If $X, Y \in \mathcal{C}$, then $(XY) \in \mathcal{C}$.

The abreviation of $(\ldots((X_1X_2)X_3)\ldots X_n)$ is $X_1X_2X_3\ldots X_n$. *Combinators* are $c$–terms without variables. One notices that the notion of abstraction is not in the formalism. The equality is given by the following axiom–schemes and rules:

**Definition 1.2.2.**

1. $\mathsf{I}X = X$; (identity)
2. $\mathsf{K}XY = X$;
3. $\mathsf{S}XYZ = XZ(YZ)$;
4. $X = X$;
5. $\dfrac{X = Y}{Y = X}$;
6. $\dfrac{X = Y \quad Y = Z}{X = Z}$;
7. $\dfrac{X = X_1 \quad Y = Y_1}{XY = X_1Y_1}$.

This equality is not extensional.

Abstraction is not a part of the formalism, but it can be defined as an operator, $\lambda^* x$, on $c$–terms. It can be defined in different ways. We will follow the approach of Stenlund, 1972.

**Definition 1.2.3.**

(i) $\lambda^* x.x = \mathsf{I}$;

(ii) $\lambda^* x.Y = \mathsf{K}Y$, if $x$ does not occur in $Y$;

(iii) $\lambda^* x.Yx = Y$, if $x$ does not occur in $Y$;

(iv) $\lambda^* x.YZ = \mathsf{S}(\lambda^* x.Y)(\lambda^* x.Z)$, otherwise.

The extensional equality of $c$–terms is obtained by extending the Definition 1.2.2 with the following rule

$$(\xi') \qquad \frac{X = Y}{\lambda^* z.X = \lambda^* z.Y}.$$

With this additional notion of abstraction it is possible to establish mappings from $\mathcal{C}$ into $\Lambda$ $(\ )^\lambda : \mathcal{C} \to \Lambda$ and vice versa $(\ )^c : \Lambda \to \mathcal{C}$ such that

1. – $x^\lambda = x$;
   – $\mathsf{I}^\lambda = \lambda x.x$ , $\mathsf{K}^\lambda = \lambda xy.x$ , $\mathsf{S}^\lambda = \lambda xyz.xz(yz)$;
   $(XY)^\lambda = X^\lambda Y^\lambda$;
2. – $x^c = x$;
   – $(MN)^c = M^c N^c$;
   – $(\lambda x.M)^c = \lambda^* x.M^c$.

The lambda calculus and the theory of combinators are of the same expressive power and this is proved using these to mappings.

**Theorem 1.2.4.** *Let $X, Y \in \mathcal{C}$ and $M, N \in \Lambda$. Then:*

*(i)* $(X^\lambda)^c \equiv X$, $(\lambda^* z.X)^\lambda = \lambda z.X^\lambda$.

*(ii)* $(M^c)^\lambda = M$.

*(iii)* $X = Y$ *if and only if* $X^\lambda = Y^\lambda$.

*(iv)* $M = N$ *if and only if* $M^c = N^c$.

The original idea to base logic on a consistent system of lambda terms or combinators was due to Church and Curry. Illative combinatory logic was the full system of Church and was meant to connect logic with the lambda calculus or the theory of combinators, i.e. to formalize inference. It was shown to be inconsistent by Kleene and Rosser. Later on the main problem in this approach was that the considered "systems of illative combinatory logic were either to weak to provide a sound interpretation of logic or to strong, sometimes even inconsistent" (quotation from Barendregt et al., 1992). Sound and complete interpretations of the first-order propositional and predicate logic into certain systems of illative combinatory logic are given in Barendregt et al., 1992.

## 1.3 Basic systems with intersection types

A *type assignment system* assigns formulae (types) of a certain language to some lambda terms in order to specify the properties of these lambda terms. There are two basically different ways of formulating type assignment systems of typed lambda calculi. In the systems given à *la Curry*, it is possible to assign to each term variable infinitely many types. So it is with terms; i.e., if a term has a type, then it has infinitely many types. In the systems given à *la Church*, exactly one type is assigned to each term variable (term). Hence, first, we obtain annotated terms and afterwards by the rules of the type assignment system we obtain the set of all typed terms. The difference between these two ways of giving a type assignment system is clarified in Barendregt, 1992.

The basic type assignment system is the *simply typed lambda calculus*, else called Curry's type inference system. It can be given in both ways. The set of types is in the both cases the same. The only type forming operator is the *arrow*. Arrow types are assigned to functions and hence the application of lambda terms yields the arrow elimination on types, while the abstraction–(function construction) yields the arrow introduction.

Much of the untyped theory is in accordance with the introduction of arrow types. One of the important properties of the simply typed lambda calculus is that self–application is not typable, i.e., lambda terms containing self–application have no type.

The simply typed lambda calculus can be extended in various ways. The extensions of its Church version form *Barendregt's cube*. The *second-order lambda calculus* $\lambda 2$ can also be given in both ways, while it is not clear up to now how to obtain the Curry version of the *theory of constructions* $\lambda C$.

Intersection types had been introduced in Coppo et al., 1980, and Barendregt et al., 1983. They are introduced as a generalization of Curry's type inference system, in order to characterize a larger class of terms. The main idea is the introduction of a new type–forming operator, the intersection $\cap$.

The types of the *lambda calculus with intersection types* $\lambda\cap$ are propositional formulae with connectives $\rightarrow$ and $\cap$, where $\cap$ is a *specific conjunction* whose properties are in accordance with its interpretation as intersection of types. The basic notions of the lambda calculus with intersection types, also called intersection type assignment system, or Torino system $\lambda\cap$, are given in Coppo et al., 1980, 1981, Barendregt et al., 1983 and can be found in the survey of typed lambda calculi in Barendregt, 1992, and Krivine, 1990. Up to now there are only Curry versions of the systems with intersection types.

The set of types $T$ of $\lambda\cap$ is defined in the following way:

**Definition 1.3.1.**

*(i)* $V = \{\alpha, \beta, \gamma, \alpha_1, \ldots\} \subset T$ *(V is a denumerable set of propositional variables).*

*(ii)* $\omega \in T$.

*(iii)* *If* $\sigma, \tau \in T$, *then* $(\sigma \rightarrow \tau) \in T$.

*(iv)* *If* $\sigma, \tau \in T$, *then* $(\sigma \cap \tau) \in T$.

Let $\alpha, \beta, \gamma, \alpha_1, \ldots$ be schematic letters for type variables, and let $\sigma, \tau, \varphi, \psi, \sigma_1, \ldots$ be schematic letters for types.

**Definition 1.3.2.**

*(i) A pre-order $\leq$ is introduced on $T$ in the following way:*

1. $\sigma \leq \sigma$
2. $\sigma \leq \tau, \quad \tau \leq \rho \Rightarrow \sigma \leq \rho$
3. $\sigma \leq \omega$
4. $\omega \leq \omega \rightarrow \omega$
5. $(\sigma \rightarrow \rho) \cap (\sigma \rightarrow \tau) \leq \sigma \rightarrow (\rho \cap \tau)$
6. $\sigma \cap \tau \leq \sigma, \quad \sigma \cap \tau \leq \tau$
7. $\sigma \leq \tau, \quad \sigma \leq \rho \Rightarrow \sigma \leq \tau \cap \rho$
8. $\sigma \leq \sigma_1, \quad \tau \leq \tau_1 \Rightarrow \sigma_1 \rightarrow \tau \leq \sigma \rightarrow \tau_1$.

*(ii) $\sigma \sim \tau$ if and only if $\sigma \leq \tau$ and $\tau \leq \sigma$.*

The expression $M : \sigma$, called *statement*, where $M \in \Lambda, \sigma \in T$ links the terms of $\Lambda$ and the types of $T$. $M$ is the *subject* and $\sigma$ is the *predicate* of the statement $M : \sigma$. If $x \in V$, then $x : \tau$ is a *basic statement*. A *basis* is a set of basic statements, with different term variables.

$\Gamma, \Delta, \Gamma_1, \ldots$ are used as schematic letters for bases. Intuitively, $M : \sigma$ means that the term $M$ is of type $\sigma$.

**Definition 1.3.3.** *The type assignment system $\lambda\cap$ is defined by the following rules:*

$$(start\ \ rule) \quad \frac{(x : \sigma) \in \Gamma}{\Gamma \vdash x : \sigma};$$

$$(\rightarrow E) \quad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau};$$

$$(\rightarrow I) \quad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : \sigma \rightarrow \tau};$$

$$(\cap E)\quad \frac{\Gamma \vdash M : \sigma \cap \tau}{\Gamma \vdash M : \sigma}, \qquad \frac{\Gamma \vdash M : \sigma \cap \tau}{\Gamma \vdash M : \tau};$$

$$(\cap I)\quad \frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \cap \tau};$$

$$(\omega)\quad \frac{}{\Gamma \vdash M : \omega};$$

$$(\leq)\quad \frac{\Gamma \vdash M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash M : \tau}.$$

The *Curry version* of the *simply typed lambda calculus* $\lambda \rightarrow$ is given by the (*start rule*), $(\rightarrow E)$ and $(\rightarrow I)$, so it can be considered as a restriction of $\lambda\cap$ (we write $\vdash_{\lambda\rightarrow}$ for the corresponding turnstile). The system $\mathcal{D}\Omega$ is obtained from $\lambda\cap$ by leaving out $(\leq)$ (we write $\vdash_{\mathcal{D}\Omega}$ for the corresponding turnstile). This is actually the original intersection system introduced in Coppo et al., 1980. It is extended by the relation $\leq$ on types and the rule $(\leq)$ in order to get the completeness result for the system, we will come back to this question later in this Section. The system $\lambda\cap_{-\omega}$ is obtained from $\lambda\cap$ by leaving out the rule $(\omega)$ (we write $\vdash_{-\omega}$ for the corresponding turnstile). In the system $\mathcal{D}$, $\omega$ is not a type; this system is given without the rules $(\omega)$ and $(\leq)$ (we write $\vdash_{\mathcal{D}}$ for the corresponding turnstile). There is an extensive study of the systems $\mathcal{D}$ and $\mathcal{D}\Omega$ in Krivine, 1990. We shall deal mostly with these four intersection type systems,

$$\begin{tikzcd} & \mathcal{D} \arrow[dl] \arrow[dr] & \\ \mathcal{D}\Omega \arrow[dr] & & \lambda\cap_{-\omega} \arrow[dl] \\ & \lambda\cap & \end{tikzcd}.$$

There is another formulation of the lambda calculus with intersection types in Pottinger, 1980, without $\leq$ but with the rule

$$\frac{\Gamma \vdash \lambda x.Mx : \sigma, x \notin Fv(M)}{\Gamma \vdash M : \sigma}\ (\eta).$$

It turns to be equivalent to $\lambda\cap_{-\omega}$. This will be considered in Chapter 2. By varying the rules of the type assignment (see Krivine, 1990, and van Bakel, 1993) it is possible to obtain a variety of different intersection type systems.

Intersection types for the theory of combinators have been introduced in Dezani et al., 1992, as a translation of the intersection type system for lambda calculus. The translations between untyped lambda calculus and combinators were mentioned in Section 1.2.

It is possible to give a sequent–system formulation of $\lambda\cap$ where the (start rule), $(\leq)$ and $(\omega)$ are the same while $(\rightarrow I)$ and $(\cap I)$ are called $(\rightarrow R)$ and $(\cap R)$, respectively. The new rules are

$$\frac{\Gamma \vdash N : \tau \quad \Delta[x:\rho] \vdash M : \sigma}{\Delta[y : \tau \rightarrow \rho, \Gamma] \vdash M[yN/x] : \sigma}\ (\rightarrow L),\ (y \text{ is fresh for } \Gamma \text{ and } \Delta)$$

and

$$\frac{\Gamma[x:\sigma] \vdash M : \rho}{\Gamma[x : \sigma \cap \tau] \vdash M : \rho}\ (\cap L).$$

Remark. As we can notice the introduced type systems are given à la Curry, i.e., types are assigned to untyped lambda terms. Typed systems given à la Church deal with typed lambda terms, i.e., lambda terms with types "built in". The simply typed lambda calculus $\lambda \rightarrow$ has a Church version as well. It is given by the (*start rule*), $(\rightarrow E)$ of Definition 1.3.3 and

$$(\rightarrow I)\quad \frac{\Gamma, x:\sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.M) : \sigma \rightarrow \tau}.$$

Hence, the fundamental difference between the Curry and Church version of $\lambda \rightarrow$ is in the $(\rightarrow I)$ rule. We shall need the Church version of $\lambda \rightarrow$ in Section 4.1. It is not yet clear whether it is possible to obtain the Church version of an intersection type system for lambda calulus. The corresponding intersection type assignment systems for combinators have both: Curry and Church versions (see Dezani et al., 1991, and Venneri, 1992).

## Basic properties of $\lambda\cap$

The basic combinators $\mathsf{I}$, $\mathsf{S}$ and $\mathsf{K}$ are typable in $\lambda\rightarrow$:

$$\begin{aligned}
&\vdash_{\lambda\rightarrow} \mathsf{I} : \alpha \rightarrow \alpha,\\
&\vdash_{\lambda\rightarrow} \mathsf{S} : (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma)),\\
&\vdash_{\lambda\rightarrow} \mathsf{K} : \alpha \rightarrow (\beta \rightarrow \alpha).
\end{aligned}$$

Therefore they are typable in $\lambda\cap$, since it is the extension of $\lambda\rightarrow$. Obviously, more types can be assigned to these terms in $\lambda\cap$. The way to obtain all the types that are assigned to a term in $\lambda\cap$, the *principal type scheme* from which all possible types can be derived, is given in Ronchi et al., 1984. We shall deal with this subject in Section 4.2. Self–application is not typable in $\lambda\rightarrow$, but it is in $\lambda\cap$. For example, $\lambda x.xx$ has no type in $\lambda\rightarrow$, but it has a type in $\lambda\cap$ since we have the following derivation in $\lambda\cap$

$$\dfrac{\dfrac{\dfrac{x:\sigma\cap(\sigma\rightarrow\tau)\vdash x:\sigma\cap(\sigma\rightarrow\tau)}{x:\sigma\cap(\sigma\rightarrow\tau)\vdash x:\sigma\rightarrow\tau \quad x:\sigma\cap(\sigma\rightarrow\tau)\vdash x:\sigma}}{x:\sigma\cap(\sigma\rightarrow\tau)\vdash xx:\tau}\ (\rightarrow E)}{\vdash \lambda x.xx:(\sigma\cap(\sigma\rightarrow\tau))\rightarrow\tau}$$

Also, the term $\Omega$ has no type in $\lambda\rightarrow$, while it is trivially typable by $\omega$ in $\lambda\cap$, i.e., $\vdash \Omega:\omega$. These examples give the intuition about the difference between the classes of terms which are typable in $\lambda\rightarrow$ and $\lambda\cap$, respectively. We shall deal with this subject in Chapter 2.

The conservativity of $\lambda\cap$ over $\lambda\rightarrow$ and some structural properties of $\lambda\cap$ given bellow are proved in Barendregt et al., 1983.

**Proposition 1.3.4. (Conservativity)**
*Let $\sigma$ be an arrow type. Then, if $\Gamma\vdash M:\sigma$ , then $\Gamma\vdash_{\lambda\rightarrow} M:\sigma$.*

**Proposition 1.3.5. (Structural properties)**

*(i) If $\Gamma\vdash MN:\sigma$, then there exists a $\tau\in T$ such that $\Gamma\vdash M:\tau\rightarrow\sigma$ and $\Gamma\vdash N:\tau$.*

*(ii) If $x$ is not in $\Gamma$, then $\Gamma\vdash\lambda x.M:\sigma\rightarrow\tau$ if and only if $\Gamma\cup\{x:\sigma\}\vdash M:\tau$.*

*(iii) If $\Gamma \vdash x : \tau$, then there exists a $\sigma \in T$ such that $(x : \sigma) \in \Gamma$ and $\sigma \leq \tau$.*

Let $\Gamma | Fv(M)$ denote the restriction of $\Gamma$ to the free variables of $M$ and let $\Gamma \cap \Delta = \{x : \sigma \cap \tau \mid (x : \sigma) \in \Gamma \text{ and } (x : \tau) \in \Delta\}$.

**Proposition 1.3.6. (Basis lemma)**
*Let $\Gamma$ be a basis.*

*(i) Let $\Gamma' \supseteq \Gamma$ be a basis. Then if $\Gamma \vdash M : \sigma$ , then $\Gamma' \vdash M : \sigma$.*

*(ii) If $\Gamma \vdash M : \sigma$ , then $\Gamma | Fv(M) \vdash M : \sigma$.*

**Proposition 1.3.7. (Subterm lemma)**
*Let $N$ be a subterm of $M$.*
*If $\Gamma \vdash M : \sigma$ , then $\Delta \vdash N : \tau$ , for some $\Delta$ and $\tau$.*

This means that if a term is typable in $\lambda\cap$, then each of its subterms is typable, as well, i.e., if a term has a meaning, then each of its subterms is meaningful in the system.

Further, let us consider reductions in the type systems. If a term $M$ is typable by a certain type $\sigma$ and if $M \twoheadrightarrow N$ the question is whether $N$ is typable by $\sigma$ as well. The answer for $\lambda \rightarrow$ is yes and we say that the subject reduction property holds for $\lambda \rightarrow$. It holds for $\lambda\cap$ as well.

**Proposition 1.3.8. (Subject reduction)**

*(i) If $\Gamma \vdash M : \sigma$ and $M \twoheadrightarrow N$, then $\Gamma \vdash N : \sigma$.*

*(ii) If $\Gamma \vdash M : \sigma$ and $M \twoheadrightarrow_{\eta} N$, then $\Gamma \vdash N : \sigma$.*

The subject reduction property holds for the systems $\lambda\cap_{-\omega}$, $\mathcal{D}\Omega$, $\mathcal{D}$ and for all eight systems of Barendregt's cube, as well (see Barendregt, 1992). The converse question is whether the type systems are closed under expansion. $\lambda\cap$ is closed under $\beta$–expansion and the reason why it is so is the characteristic of this system that a term can have more types, as shown in Barendregt,

1992. Let $(\lambda x.P)Q \to P[Q/x]$. Let us suppose $\Gamma \vdash P[Q/x] : \sigma$ in order to show $\Gamma \vdash (\lambda x.P)Q : \sigma$. The term $Q$ occurs $n \geq 0$ times in $P[Q/x]$, each occurrence having type $\tau_i$, say, $1 \leq i \leq n$. Let us define

$$\tau \equiv \begin{cases} \tau_1 \cap \ldots \cap \tau_n & , \quad n > 0 \\ \omega & , \quad n = 0. \end{cases}$$

Then $\Gamma \vdash Q : \tau$ and $\Gamma, x : \tau \vdash P : \sigma$, where $x$ is fresh for $\Gamma$. Hence, $\Gamma \vdash (\lambda x.P) : \tau \to \sigma$ and $\Gamma \vdash (\lambda x.P)Q : \sigma$.

On the other hand $\lambda \to$ is not closed under $\beta$–expansion, since as we saw that $\mathsf{KI\Omega} \to \mathsf{I}$ and $\mathsf{I}$ is typable in $\lambda \to$, i.e., $\vdash_{\lambda\to} \mathsf{I} : \alpha \to \alpha$, but $\mathsf{KI\Omega}$ is not typable in $\lambda \to$. The terms typable in $\lambda \to$ are strongly normalizing. This was first proved in Tait, 1967. Later on Girard, 1972, proved that the terms typable in $\lambda 2$ are strongly normalizing. Even all eight systems of Barendregt's cube have this property (see Barendregt, 1992).

**Proposition 1.3.9.** *If* $\Gamma \vdash_{\lambda\to} M : \sigma$, *then* $M$ *is strongly normalizing.*

Some problems of interest in the systems given à la Curry are:

- *Type checking* in the system.
  (Is it possible to assign a given type to a given term, $M : \sigma$?).
- *Typability* in the system.
  (Is there a type that can be assigned to a given term, $M$ :?).
- *Inhabitation* in the system.
  (Is there a term of a given type, ? : $\sigma$).

All three problems are decidable in $\lambda \to$, see Barendregt, 1992.

**Proposition 1.3.10.**

*(i) Type–checking is decidable in* $\lambda \to$.

*(ii) Typability is decidable in* $\lambda \to$.

*(iii) Inhabitation is decidable in* $\lambda \to$.

The proofs of decidability of typability and type-checking in $\lambda \rightarrow$ are based on the principal type scheme property given in Hindley, 1969, and on the pattern matching and unification algorithm given in Robinson, 1965. The problem of inhabitation will be discussed in Chapter 3.

The problem of typability in $\lambda\cap$ is trivial, but in some other intersecion type systems it is undecidable (see Barendregt, 1992, van Bakel, 1992, and Krivine, 1990). It will be discussed in Chapter 2. The problem of type-checking in $\lambda\cap$ is undecidable. Although variations between the considered intersection type systems may seem subtle, the answers to the questions above make these systems rather different.

**Proposition 1.3.11.**

*(i) Type-checking is undecidable in* $\lambda\cap$.

*(ii) Typability is undecidable in* $\lambda\cap_{-\omega}$.

A new proof of the undecidability of the typability in $\lambda\cap_{-\omega}$ will be given in Chapter 2.

The problem of inhabitation in the intersection type systems is still open. We shall consider this problem and connect it with the inhabitation of an extension of $\lambda \rightarrow$ in Chapter 3.

An interesting approach to type systems is the "Curry–Howard isomorphism" or formulae–as–types interpretation (see Howard, 1980). It is a mapping of constructive proofs of logical formulae into lambda terms or combinators of related types and conversely. This enables on the one hand to provide a logical meaning of type constructors, on the other logical systems can be seen in a computational way.

The simply typed lambda calculus corresponds to the implicational fragment of intuitionistic propositional logic in the following way:

– arrow–types correspond to implicational formulae;

- the elimination of an arrow–type (application) corresponds to the elimination rule of implication (Modus Ponens) in the natural deduction formulation of logic;
- the introduction of an arrow–type (abstraction) corresponds to the introduction rule of implication (Deduction Theorem) in the natural deduction formulation of logic.

The simply typed lambda calculus has been extended in order to correspond to intuitionistic propositional logic with all its connectives. In this approach all inhabited types are exactly all provable formulae (see Howard, 1980).

The Curry–Howard isomorphism of proofs as terms brings to life Brouwer's idea of a constructive proof.

All the systems of Barendregt's cube correspond by the Curry–Howard isomorphism to some constructive logic. Second–order or polymorphic lambda calculus with quantification over types corresponds to second–order propositional logic. Girard's system $F_\omega$ corresponds to higher–order propositional logic (for more details see Barendregt, 1992, Geuvers, 1988, and Girard et al., 1991).

The Curry-Howard isomorphism between formulae as types and terms as proofs does not hold for the intersection type systems. It fails since the terms do not "code" the proofs anymore. The steps $(\cap E)$ and $(\cap I)$ do not change the lambda term, i.e., the lambda term is the same in the premises and in the conclusion in both of these rules. Thus in these two rules the lambda term remains the same, although the deduction grows, and hence it does not correspond to the deduction. More details on this subject will be given in Chapter 3.

Models of the untyped lambda calculus were a matter of discussion up to the 1970's. The main problem was to interpret objects that are arguments and functions to be applied to these arguments at the same time. The domain $D$ of the semantics for the lambda calculus has to be isomorphic to its function space $D \to D$. This is impossible by Cantor's theorem. The problem was overcome by Scott (see Scott, 1980) by restricting $D \to D$ to the set of continuous functions with respect to the so called Scott topology on D. We shall deal with the Scott topology on $\Lambda$ in Chapter 4. This first model of the untyped lambda calculus is called $D_\infty$. Later on various lambda models have been developed (see Barendregt, 1984).

Lambda models can be given by a first-order definition, but there are syntactical and categorical descriptions of these classes as well. We shall recall the syntactical definition only, since we need it for the definition of models of $\lambda\cap$.

**Definition 1.3.12.** *Let $\mathcal{M} =< A, \cdot >$ be an applicative structure (groupoid).*

*(i)* $Val\,(\mathcal{M}) = \{\xi | \xi : V \to A\}$.

*(ii) An interpretation in $\mathcal{M}$ are the maps $\| \quad \|_\xi : \Lambda \to A$ satisfying the following conditions:*

1. $\|x\|_\xi = \xi(x)$,
2. $\|MN\|_\xi = \|M\|_\xi \|N\|_\xi$,
3. $\|\lambda x.M\|_\xi \cdot a = \|M\|_{\xi(a/x)}$,
4. $(\forall x \in Fv(M),\ \|x\|_\xi = \|x\|_{\xi'}) \Rightarrow \|M\|_\xi = \|M\|_{\xi'}$.

*(iii) A syntactical applicative structure is of the form $\mathcal{M} =< A, \cdot, \| \quad \| >$, where $\| \quad \|$ is an interpretation in $\mathcal{M}$.*

*(iv) A syntactical applicative structure $\mathcal{M} =< A, \cdot, \| \quad \| >$ is a lambda model if*

$$(\forall a \in A,\ \|M\|_{\xi(a/x)} = \|N\|_{\xi(a/x)}) \Rightarrow \|\lambda x.M\|_\xi = \|\lambda x.N\|_\xi.$$

The basic idea of *filter models*, models of $\lambda\cap$ introduced in Barendregt et al., 1983, is that a lambda model can be obtained from the set of intersection types $T$ with the partial ordering $\leq$.

*Filter models* are introduced in Barendregt et al., 1983, in order to show the completeness of $\lambda\cap$.

**Definition 1.3.13.** *Let $\mathcal{M} =< D, \cdot, \| \ \| >$ be a lambda model.*

1. *If $\xi$ is a valuation of term–variables $V$ in $D$, $\xi : V \to D$, then $\|M\|_{\xi}^{\mathcal{M}} \in D$ is the valuation of $M$ in $\mathcal{M}$ via $\xi$.*
2. *If $\rho$ is the interpretation of type variables in $D$, $\rho : V \to \mathcal{P}(D) = \{X | X \subseteq D\}$, then $\|\sigma\|_{\rho}^{\mathcal{M}} \in \mathcal{P}(D)$, the interpretation of $\sigma$, in $\mathcal{M}$ via $\rho$ is defined as follows:*

   *(i)* $\|\alpha\|_{\rho}^{\mathcal{M}} = \rho(\alpha)$, *for all* $\alpha \in V$.

   *(ii)* $\|\sigma \to \tau\|_{\rho}^{\mathcal{M}} = \{d \in D | \forall e \in \|\sigma\|_{\rho}^{\mathcal{M}}, \ d \cdot e \in \|\tau\|_{\rho}^{\mathcal{M}}\}$.

   *(iii)* $\|\sigma \cap \tau\|_{\rho}^{\mathcal{M}} = \|\sigma\|_{\rho}^{\mathcal{M}} \cap \|\tau\|_{\rho}^{\mathcal{M}}$.

   *(iv)* $\|\omega\|_{\rho}^{\mathcal{M}} = D$.
3. $\mathcal{M}, \xi, \rho, \models M : \sigma$ *iff* $\|M\|_{\xi}^{\mathcal{M}} \in \|\sigma\|_{\rho}^{\mathcal{M}}$.
   $\mathcal{M}, \xi, \rho \models \Gamma$ *iff* $\mathcal{M}, \xi, \rho \models x : \varphi$ *for all* $(x : \varphi) \in \Gamma$.
   $\Gamma \models M : \sigma$ *iff* $\forall \mathcal{M}, \xi, \rho(\mathcal{M}, \xi, \rho \models \Gamma \Rightarrow \mathcal{M}, \xi, \rho \models M : \sigma)$.

This kind of type interpretation appears frequently in the interpretations of logical implication (see Došen, 1992). This term valuation and type interpretation give the soundness result for $\lambda\cap$.

**Proposition 1.3.14** *(Barendregt et al., 1983.)*

*(i)* *If $\sigma \leq \tau$, then $\forall \mathcal{M}, \rho$ $\|\sigma\|_{\rho}^{\mathcal{M}} \subseteq \|\tau\|_{\rho}^{\mathcal{M}}$.*

*(ii)* *(Soundness). If $\Gamma \vdash M : \sigma$, then $\Gamma \models M : \sigma$.*

**Definition 1.3.15.** *A filter is a subset $d \subseteq T$ such that:*

*(i)* $\omega \in d$;

*(ii)* $\sigma, \tau \in D \Rightarrow \sigma \cap \tau \in d$;

*(iii)* $\sigma \in d, \sigma \leq \tau \Rightarrow \tau \in d$.

**Lemma 1.3.16.** *(Barendregt et al., 1983.)*

*(i)* $\{\sigma \mid \Gamma \vdash M : \sigma\}$ *is a filter.*

*(ii)* $\Gamma \vdash x : \sigma$ *if and only if* $\sigma$ *is in the filter generated by* $\{\tau \mid (x : \tau) \in \Gamma\}$.

**Definition 1.3.17.**

*(i) Let* $\mathcal{F} = \{d | d \text{ is a filter }\}$. *For* $d_1, d_2 \in \mathcal{F}$ *define*

$$d_1 \cdot d_2 = \{\tau \in T | \exists \sigma \in d_2, \sigma \to \tau \in d_1\}.$$

*(ii) Let* $\xi$ *be a valuation in* $\mathcal{F}$. *Then* $\Gamma_\xi = \{x : \sigma | \sigma \in \xi(x)\}$.

*(iii) For* $M \in \Lambda$ *define* $\|M\|_\xi = \{\sigma \in T | \Gamma_\xi \vdash M : \sigma\} (\in \mathcal{F})$.

**Theorem 1.3.18.** *(Barendregt et al., 1983.)*

$$< \mathcal{F}, \cdot, \| \ \| > \quad \textit{is a lambda model.}$$

In order to obtain the completeness result for $\lambda\cap$ it is necessary to introduce a special type interpretation

$$\|\varphi\|_{\rho_0} = \{d \in \mathcal{F} | \varphi \in d\}$$

and a term valuation that depends on a basis $\Gamma$

$$\xi_\Gamma(x) = \{\sigma \in T | \Gamma \vdash x : \sigma\} \ (\in \mathcal{F}).$$

**Theorem 1.3.19. (Completeness theorem)** *(Barendregt et al., 1983.)*

$$\Gamma \vdash M : \sigma \ \textit{ if and only if } \ \Gamma \models M : \sigma.$$

The completeness of $\lambda\cap$ was proved in Hindley, 1982, using the term model of lambda calculus.

# 2 Typability in intersection type assignment systems

In this Chapter we shall discuss the typability in the intersection type assignment systems which are given in Section 1.3. The problem of the typability in an type system is whether there is a type of a given term, $M$ :? (see Section 1.3).

The problem of typability in the full intersection type assignment system $\lambda\cap$ is trivial, since every lambda term is typable by $\omega$. For the same reasons typability in $\mathcal{D}\Omega$ is trivial as well. This property changes essentially when the ($\omega$)–rule is left out. It turns out that all strongly normalizing lambda terms are typable in $\lambda\cap_{-\omega}$ and $\mathcal{D}$ and that they are the only terms typable in these systems. This Chapter deals with this powerful property of the intersection type assignment systems $\lambda\cap_{-\omega}$ and $\mathcal{D}$.

The idea that strongly normalizing lambda terms are exactly the terms typable in the intersection type assignment systems without the ($\omega$)–rule first appeared in Pottinger, 1980, Coppo et al., 1981, and Leivant, 1983. Further, this subject is treated in Ronchi et al., 1984, Krivine, 1990, and van Bakel, 1992, with different approaches. We shall present a modified proof of this property and compare it with the proofs mentioned above.

The aim of the extension of the simply typed lambda calculus $\lambda \to$, (also called Curry's type assignment system) in Coppo et al., 1981, was to provide a functional theory in which all solvable terms, i.e., terms with a head normal form, have "meaningful" types. This is not the case in the basic Curry's functional theory, in which every typable term is strongly normalizing, but there are lambda terms, even in normal form which are not typable, for example $\lambda x.xx$. The set of so called proper types is assigned only to solvable terms.

In Section 2.1 we shall compare deductions in $\lambda\cap_{-\omega}$ and $\mathcal{D}$ showing that these two systems are equivalent from the point of view of typability, as well as of inhabitation. In Section 2.2 we shall present a proof of strong normalization for $\mathcal{D}$ and $\lambda\cap_{-\omega}$ based on the proof of the strong normalization for $\lambda \rightarrow$ and $\lambda 2$ given in Barendregt, 1992. The strong normalization holds for all eight systems of Barendregt's cube. The very essential and outstanding property of the intersection type systems $\lambda\cap_{-\omega}$ and $\mathcal{D}$ is the converse of the strong normalization property, i.e., the fact that all strongly normalizing terms are typable in $\lambda\cap_{-\omega}$ and $\mathcal{D}$. In Section 2.3 we shall present a modified proof of this property and compare it with the proofs mentioned above. The undecidability of typability is the consequence of this property. It will also be discussed.

## 2.1 Typability and inhabitation in various intersection type systems

In this Section we shall prove in Theorem 2.1.3 that the type systems $\lambda\cap_{-\omega}$ and $\mathcal{D}$ are equivalent from the point of view of typability. On the other hand we shall prove in Theorem 2.1.6 that these systems are equivalent from the point of view of inhabitation as well.

There are differences in the formulation of $\lambda\cap_{-\omega}$ and $\mathcal{D}$. Although $\mathcal{D}$ is obtained from $\lambda\cap_{-\omega}$ by leaving out the rule $(\leq)$, these two systems are equivalent from the point of view of typability, as well as of inhabitation. In order to prove the equivalence of typability, first we prove in Proposition 2.1.2 that for any derivation of a statement in $\lambda\cap_{-\omega}$ there is a derivation in $\mathcal{D}$ with the same subject (term). A similar result is proved in van Bakel, 1992, for systems with strict types which do not contain $\omega$ on the right–hand side of the arrow. In order to prove the equivalence for inhabitation, first we prove in Proposition 2.1.5 that if a type $\sigma$ is inhabited in $\mathcal{D}$ by a term $M$ and $\sigma \leq \tau$, then $\tau$ is inhabited in $\mathcal{D}$, as well, by a term $M' =_\eta M$.

The following statement is obvious, since $\mathcal{D}$ is "lacking" the rule $(\leq)$.

**Proposition 2.1.1.** *If* $\Gamma \vdash_{\mathcal{D}} M : \sigma$, *then* $\Gamma \vdash_{-\omega} M : \sigma$.

The converse holds in the sense that a term typable in $\lambda\cap_{-\omega}$ is typable in $\mathcal{D}$ as well, but with a smaller type and in a larger basis. We say that a basis $\Gamma'$ is *larger* than the given basis $\Gamma$, we write $\Gamma' \geq \Gamma$, if for every $x : \sigma \in \Gamma$ there is one and only one $x : \sigma' \in \Gamma'$, for some $\sigma' \geq \sigma$.

**Proposition 2.1.2.** *If* $\Gamma \vdash_{-\omega} M : \sigma$*, then there are* $\Gamma'$ *and* $\sigma'$ *such that* $\Gamma' \vdash_{\mathcal{D}} M : \sigma'$ *and* $\sigma' \leq \sigma, \Gamma' \geq \Gamma$.

**Proof.** By induction on the derivation in $\lambda\cap_{-\omega}$. Here, we shall use the sequent-system formulation of $\lambda\cap_{-\omega}$.

**Case** (*start rule*). If the last applied step is (start rule), then before

$$\Gamma \ni (x : \sigma) \vdash_{-\omega} x : \sigma,$$

there are no applications of $(\leq)$, and so this is a derivation in $\mathcal{D}$.

**Case** $(\rightarrow L)$. If the last applied rule is $(\rightarrow L)$

$$\frac{\Gamma \vdash_{-\omega} N : \varphi \quad \Delta[x : \rho] \vdash_{-\omega} M : \tau}{\Delta[y : \varphi \rightarrow \rho, \Gamma] \vdash_{-\omega} M[yN/x] : \tau} \quad ,$$

then by the induction hypothesis

$$\Gamma' \vdash_{\mathcal{D}} N : \varphi' \quad , \quad \Delta'[x : \rho'] \vdash_{\mathcal{D}} M : \tau'$$

for some $\varphi' \leq \varphi, \Gamma' \geq \Gamma$ and $\tau' \leq \tau, \Delta' \geq \Delta (\rho' \geq \rho)$. Then by $(\rightarrow L)$

$$\Delta'[y : \varphi' \rightarrow \rho', \Gamma'] \vdash_{\mathcal{D}} M : \tau'$$

and $\varphi' \rightarrow \rho' \geq \varphi \rightarrow \rho$.

**Case** $(\rightarrow I)$. If the last applied rule is $(\rightarrow I)$

$$\frac{\Gamma, x : \rho \quad \vdash_{-\omega} M : \tau}{\Gamma \vdash_{-\omega} \lambda x.M : \rho \rightarrow \tau} (\rightarrow I),$$

then by the induction hypothesis $\Gamma', x : \rho' \vdash_{\mathcal{D}} M : \tau'$ for some $\tau' \leq \tau, \Gamma' \geq \Gamma$ and $\rho' \geq \rho$. Hence by $(\rightarrow I)$

$$\Gamma' \vdash_{\mathcal{D}} \lambda x.M : \rho' \rightarrow \tau'$$

where $\rho' \rightarrow \tau' \leq \rho \rightarrow \tau$.

**Case** $(\cap L)$. If the last applied rule is $(\cap L)$

$$\frac{\Gamma[x : \rho] \vdash_{-\omega} M : \tau}{\Gamma[x : \rho \cap \varphi] \vdash_{-\omega} M : \tau} (\cap L),$$

then by the induction hypothesis $\Gamma'[x : \rho'] \vdash_{\mathcal{D}} M : \tau'$ for some $\tau' \leq \tau$ and $\Gamma' \geq \Gamma$ $(\rho' \geq \rho)$. Therefore by $(\cap L)$

$$\Gamma'[x : \rho' \cap \varphi] \vdash_{\mathcal{D}} M : \tau'$$

where $\rho' \cap \varphi \geq \rho \cap \varphi$.

**Case** $(\cap I)$. If the last applied rule is $(\cap I)$

$$\frac{\Gamma \vdash_{-\omega} M : \rho \quad \Gamma \vdash_{-\omega} M : \tau}{\Gamma \vdash_{-\omega} M : \rho \cap \tau} \ (\cap I),$$

then by the induction hypothesis

$$\Gamma' \vdash_{\mathcal{D}} M : \rho' \qquad \Gamma'' \vdash_{\mathcal{D}} M : \tau'$$

for $\rho' \leq \rho, \Gamma' \geq \Gamma$ and $\tau' \leq \tau, \Gamma'' \geq \Gamma$. Then $\Gamma' \cap \Gamma'' \geq \Gamma$ and

$$\Gamma' \cap \Gamma'' \vdash_{\mathcal{D}} M : \rho' \qquad \Gamma' \cap \Gamma'' \vdash_{\mathcal{D}} M : \tau'.$$

Hence, by $(\cap I)$ $\Gamma' \cap \Gamma'' \vdash_{\mathcal{D}} M : \rho' \cap \tau'$.

**Case** $(\leq)$. If the last applied rule is $(\leq)$

$$\frac{\Gamma \vdash_{-\omega} M : \tau \quad \tau \leq \sigma}{\Gamma \vdash_{-\omega} M : \sigma} \ (\leq),$$

then by the induction hypothesis $\Gamma' \vdash_{\mathcal{D}} M : \tau'$ for some $\Gamma' \geq \Gamma$ and $\tau' \leq \tau$. Hence, $\tau' \leq \tau \leq \sigma$. □

The equivalence of typability in the two systems considered is the consequence of Proposition 2.1.1 and 2.1.2.

**Theorem 2.1.3.** *Given a term $M$. Then*

$$\Gamma \vdash_{-\omega} M : \sigma \ \textit{ if and only if } \ \Gamma' \vdash_{\mathcal{D}} M : \sigma' \ \textit{ for some } \ \sigma' \leq \sigma \ \textit{ and } \ \Gamma' \geq \Gamma.$$

That means that the same terms are typable in the two systems considered, but they are not always typable with the same types. The set of all types assigned to a certain term in $\mathcal{D}$ is a subset of the set of all types assigned to the same term in $\lambda\cap_{-\omega}$. The same property holds for the systems $\mathcal{D}\Omega$ and $\lambda\cap$.

**Corollary 2.1.4.** *Let $M$ be a given term. Then*

$$\Gamma \vdash M : \sigma \textit{ if and only if } \Gamma' \vdash_{\mathcal{D}\Omega} M : \sigma' \textit{ for some } \sigma' \leq \sigma \textit{ and } \Gamma' \geq \Gamma.$$

These systems are equivalent from the point of view of inhabitation as well. The obvious part of this equivalence, i.e.,

> if a type is inhabited in $\mathcal{D}$, then it is inhabited by the same term in $\lambda\cap_{-\omega}$,

is a consequence of Proposition 2.1.1. In order to prove the converse we need Proposition 2.1.2 again and the following statement:

**Proposition 2.1.5.** *If $\Gamma \vdash_{\mathcal{D}} M : \sigma$ and $\sigma \leq \tau$, then there exists a term $M'$ such that $M' =_\eta M$ and $\Gamma \vdash_{\mathcal{D}} M' : \tau$.*

**Proof.** By induction on the length of the derivation $\sigma \leq \tau$. The cases 3 and 4 of the Definition 1.3.2 are left out, since $\omega$ is not a type in $\mathcal{D}$. Let

$$\Gamma \vdash_{\mathcal{D}} M : \sigma \text{ and } \sigma \leq \tau,$$

where $\sigma \leq \tau$ is obtained by:

1. $\sigma \leq \sigma$, then $M' \equiv M$;
2. $\sigma \leq \rho, \ \ \rho \leq \tau \Rightarrow \sigma \leq \tau$, then by the induction hypothesis there exists $N =_\eta M$, such that
$$\Gamma \vdash_{\mathcal{D}} N : \rho$$
and again by the induction hypothesis there exists $M' =_\eta N$, such that
$$\Gamma \vdash_{\mathcal{D}} M' : \tau.$$
However $M' =_\eta M$.

5. $(\rho \to \tau) \cap (\rho \to \varphi) \leq \rho \to (\tau \cap \varphi)$, then by $(\cap E)$

$$\Gamma \vdash_{\mathcal{D}} M : \rho \to \tau \text{ and } \Gamma \vdash_{\mathcal{D}} M : \rho \to \varphi.$$

Let $y$ be a fresh variable for $\Gamma$, then $y \notin Fv(M)$. So by $(\to E)$

$$\Gamma, y : \rho \vdash My : \tau \ \ , \ \ \Gamma, y : \rho \vdash_{\mathcal{D}} My : \varphi$$

and by $(\cap I)$

$$\Gamma, y : \rho \ \ \vdash_{\mathcal{D}} My : \tau \cap \varphi.$$

Hence, by $(\to I)$

$$\Gamma \vdash_{\mathcal{D}} \lambda y.My : \rho \to \tau \cap \varphi$$

and $M' \equiv \lambda y.My \to_\eta M$ since $y \notin Fv(M)$.

6. $\sigma \cap \tau \leq \sigma \ \ (\tau)$, then $M' \equiv M$ and by $(\cap E)$ $\Gamma \vdash_{\mathcal{D}} M : \sigma \ \ (\tau)$.

7. $\sigma \leq \tau, \ \ \sigma \leq \rho \Rightarrow \sigma \leq \tau \cap \rho$, then by the induction hypothesis there exist terms $N$ and $P$ such that $N =_\eta M =_\eta P$ and $\Gamma \vdash_{\mathcal{D}} N : \tau$ and $\Gamma \vdash_{\mathcal{D}} P : \rho$. By Corollary 1.1.8 there is a term $M$ such that $N \twoheadrightarrow_\eta S$ and $P \twoheadrightarrow_\eta S$. Further by the subject reduction property given in Proposition 1.3.8(ii) we have that

$$\Gamma \vdash_{\mathcal{D}} S : \tau \text{ and } \Gamma \vdash_{\mathcal{D}} S : \rho.$$

Thus by $(\cap I)$

$$\Gamma \vdash_{\mathcal{D}} S : \tau \cap \rho,$$

i.e., $M' \equiv S$.

8. $\rho \leq \rho', \ \ \varphi \leq \varphi' \Rightarrow \rho' \to \varphi \leq \rho \to \varphi'$, and suppose $\Gamma \vdash_{\mathcal{D}} M : \rho' \to \varphi$. Let $x$ be a fresh variable for $\Gamma$, i.e., $x \notin Fv(M)$. By (*start rule*)

$$\Gamma, x : \rho \ \ \vdash_{\mathcal{D}} x : \rho$$

and by the induction hypothesis there is a term $N$, such that $N =_\eta x$ and

$$\Gamma, x : \rho \vdash_{\mathcal{D}} N : \rho'.$$

Then by $(\to E)$

$$\Gamma, x : \rho \vdash_{\mathcal{D}} MN : \varphi.$$

Since $\varphi \leq \varphi'$, by the induction hypothesis there exists a term $P$, such that $P =_\eta MN$ and

$$\Gamma, x : \rho \vdash_{\mathcal{D}} P : \varphi'.$$

Then by $(\rightarrow I)$

$$\Gamma \vdash_{\mathcal{D}} \lambda x.P : \rho \rightarrow \varphi'$$

and $\lambda x.P =_\eta \lambda x.MN =_\eta \lambda x.Mx \rightarrow_\eta M$, since $x \notin Fv(M)$. Thus $M' \equiv \lambda x.P$. $\square$

The equivalence of the inhabitation in $\lambda\cap_{-\omega}$ and $\mathcal{D}$ is given by the following statement:

**Theorem 2.1.6.** *Let $\sigma$ be a given type. Then*

$$\Gamma \vdash_{-\omega} M : \sigma \text{ if and only if } \Gamma' \vdash_{\mathcal{D}} M' : \sigma \text{ for some } \Gamma' \geq \Gamma \text{ and } M' =_\eta M.$$

**Proof.** $(\Leftarrow)$ Obvious.
$(\Rightarrow)$. Suppose $\Gamma \vdash_{-\omega} M : \sigma$, then by Proposition 2.1.2 there are $\Gamma'$ and $\sigma'$ such that $\Gamma' \vdash_{\mathcal{D}} M : \sigma', \sigma' \leq \sigma$ and $\Gamma' \geq \Gamma$. By Proposition 2.1.5 there is a term $M'$, such that $M' =_\eta M$ and $\Gamma' \vdash_{\mathcal{D}} M' : \sigma$. $\square$

This means that if a type is inhabited by a certain term in $\lambda\cap_{-\omega}$, then it is inhabited in $\mathcal{D}$ by an $\eta$-conversion of that term. Thus the set of inhabitants of a certain type in $\mathcal{D}$ is a subset of the set of inhabitants of the same type in $\lambda\cap_{-\omega}$. The same property holds for the systems $\mathcal{D}\Omega$ and $\lambda\cap$.

**Corollary 2.1.7.** *Let $\sigma$ be a given type. Then*

$$\Gamma \vdash M : \sigma \text{ if and only if } \Gamma' \vdash_{\mathcal{D}\Omega} M' : \sigma \text{ for some } \Gamma' \geq \Gamma \text{ and } M' =_\eta M.$$

## 2.2 Strong normalization for intersection type systems

The proof of strong normalization for $\mathcal{D}$ and $\lambda\cap_{-\omega}$, we present here in Theorem 2.2.7 is based on the proof of strong normalization for $\lambda \rightarrow$ and $\lambda 2$ in Barendregt, 1992, (subsection 4.3).

The idea that strongly normalizing lambda terms are exactly the terms typable in the intersection type assignment systems without the $(\omega)$-rule first appeared in Pottinger, 1980, Coppo et al., 1981, and Leivant, 1983. Further, this subject is treated in Ronchi et al., 1984, Krivine, 1990, and van Bakel, 1992, with different approaches. We shall compare our modified proof with the proofs mentioned above.

In Section 1.1 we saw that a lambda term is *normalizing* if at least one of its reduction paths is finite. A lambda term is *strongly normalizing* if each of its reduction paths is finite. For example:

- $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ is not normalizing, since all of its reduction paths are of the form $\Omega \rightarrow \Omega \rightarrow \ldots \rightarrow \Omega \rightarrow \ldots$;
- $\mathsf{KI}\Omega$ is normalizing since $\mathsf{KI}\Omega \rightarrow \mathsf{I}$, but not strongly normalizing since $\mathsf{KI}\Omega \rightarrow \mathsf{KI}\Omega \rightarrow \ldots \rightarrow \mathsf{KI}\Omega \rightarrow \ldots$;
- $\mathsf{KIS}$ is strongly normalizing.

All terms typable in the simply typed lambda calculus $\lambda \rightarrow$ are strongly normalizing. This was first proved in Tait, 1967. The same property for the polymorphic lambda calculus $\lambda 2$ $(F)$ was proved by Girard, 1972. This proof cannot be performed within the second-order Peano arithmetic. A general approach towards the proof of strong normalization for both $\lambda \rightarrow$ and $\lambda 2$ is given in Barendregt, 1992. It is on the lines of the proofs in Girard, 1972, and Tait, 1975.

Obviously, this property does not hold for $\lambda\cap$, since all lambda terms are typable in $\lambda\cap$. But, by leaving out the rule $(\omega)$ the situation changes essentially: not only that terms typable in $\mathcal{D}$ and $\lambda\cap_{-\omega}$ are strongly normalizing, but strongly normalizing terms are exactly all terms typable in $\mathcal{D}$ and $\lambda\cap_{-\omega}$. This result and the same result for some equivalent systems was proved first in Pottinger, 1980, Coppo et al., 1980, Leivant, 1983, and further in Ronchi et al., 1984, Krivine, 1990, and van Bakel, 1992.

First of all, let us recall some notions and notation given in Barendregt, 1992. The set of all strongly normalizing lambda terms is denoted by $SN$.

If $A$ and $B$ are sets of lambda terms, then

$$A \to B =_{def} \{M \in \Lambda | \forall N \in A \;\; MN \in B\}.$$

The *interpretation of types* $\|\;\| : T \to \mathcal{P}(\Lambda)$ is defined inductively.

**Definition 2.2.1.**

*(i)* $\|\alpha\| = SN$, *for all type variables* $\alpha$*;*

*(ii)* $\|\varphi \to \psi\| = \|\varphi\| \to \|\psi\|$, *for all types* $\varphi$ *and* $\psi$ *;*

*(iii)* $\|\varphi \cap \psi\| = \|\varphi\| \cap \|\psi\|$, *for all types* $\varphi$ *and* $\psi$.

The notion of a *saturated set* of lambda terms is defined inductively as follows:

**Definition 2.2.2.** *A set* $X \subseteq \Lambda$ *is called* <u>*saturated*</u> *if*

*(i)* $(\forall n \geq 0)$ $(\forall M_1, \ldots, M_n \in SN)$ $xM_1 \ldots M_n \in X$, *where* $x$ *is any term variable;*

*(ii)* $(\forall n \geq 0)$ $(\forall M_1, \ldots, M_n \in SN)$ $\forall N \in SN$
$M[N/x]M_1 \ldots M_n \in X \Rightarrow (\lambda x.M)NM_1 \ldots M_n \in X$.

**Proposition 2.2.3.**

*(i)* $SN$ *is saturated.*

*(ii)* *If* $A$ *and* $B$ *are saturated, then* $A \to B$ *is saturated.*

*(iii)* *If* $A$ *and* $B$ *are saturated, then* $A \cap B$ *is saturated.*

*(iv)* $\|\varphi\|$ *is saturated for all* $\varphi \in T$.

The subcase 2.2.3. (iii) in Barendregt, 1992 (Lemma 4.3.3.) is stated for an infinite intersection, hence it holds for a finite intersection as well.

Let $\rho$ be a mapping of term variables into lambda terms, i.e., $\rho : \mathrm{V} \rightarrow \Lambda$. Then the *valuation of terms* $\| \|_\rho : \Lambda \rightarrow \Lambda$ is defined inductively.

**Definition 2.2.4.**

*(i)* $\|x\|_\rho = \rho(x)$, *for all term variables* $x$ *;*

*(ii)* $\|M\|_\rho = M[\rho(x_1)/x_1, \ldots, \rho(x_n)/x_n]$, *where* $Fv(M) = \{x_1, \ldots, x_n\}$.

The relation of satisfaction, $\models$. is defined in terms of the type interpretation $\| \| : T \rightarrow \mathcal{P}(\Lambda)$ and the term valuation $\| \|_\rho : \Lambda \rightarrow \Lambda$ induced by a map $\rho : \mathrm{V} \rightarrow \Lambda$.

**Definition 2.2.5.**

*(i)* $\rho$ *satisfies* $M : \sigma$, *notation* $\rho \models M : \sigma$, *if* $\|M\|_\rho \in \|\sigma\|$.

*(ii)* $\rho$ *satisfies a basis* $\Gamma$, *notation* $\rho \models \Gamma$, *if*

$$\rho \models x : \sigma, \quad \textit{for all} \ \ (x : \sigma) \in \Gamma.$$

*(iii) A basis* $\Gamma$ *satisfies* $M : \varphi$, *notation* $\Gamma \models M : \sigma$ *if*

$$\forall \rho (\rho \models \Gamma \Rightarrow \rho \models M : \sigma).$$

Now, the proof of soundness of $\lambda \rightarrow$ given in Barendregt, 1992, can be extended to the proof of soundness of $\mathcal{D}$.

**Proposition 2.2.6. (Soundness of $\mathcal{D}$).**

$$\textit{If} \ \ \Gamma \vdash_{\mathcal{D}} M : \sigma, \ \ \textit{then} \ \ \Gamma \models M : \sigma.$$

**Proof.** By induction on the derivation of $\Gamma \vdash_{\mathcal{D}} M : \sigma$.

Case 1. The last step is the application of (*start rule*), i.e.,

$$(x : \sigma) \in \Gamma \vdash_{\mathcal{D}} x : \sigma,$$

then by Definition 2.2.5. (i) and (iii)

$$\forall \rho (\rho \models \Gamma \Rightarrow \rho \models x : \sigma),$$

and thus $\Gamma \models x : \sigma$.

**Case 2.** The last applied rule is $(\to E)$, i.e.,

$$\frac{\Gamma \vdash_{\mathcal{D}} P : \varphi \to \sigma \quad \Gamma \vdash_{\mathcal{D}} Q : \varphi}{\Gamma \vdash_{\mathcal{D}} PQ : \sigma}.$$

In order to show $\Gamma \models PQ : \sigma$, let us take an arbitrary $\rho$ and suppose $\rho \models \Gamma$. By the induction hypothesis

$$\rho \models P : \varphi \to \sigma \text{ and } \rho \models Q : \varphi.$$

Therefore by Definition 2.2.5. (i)

$$\|P\|_\rho \in \|\varphi\| \to \|\sigma\| \text{ and } \|Q\|_\rho \in \|\varphi\|,$$

so $\|P\|_\rho \|Q\|_\rho \in \|\sigma\|$, i.e., $\|PQ\|_\rho \in \|\sigma\|$. Hence $\rho \models PQ : \sigma$.

**Case 3.** The last applied rule is $(\to I)$, i.e.,

$$\frac{\Gamma, x : \varphi \vdash_{\mathcal{D}} N : \psi}{\Gamma \vdash_{\mathcal{D}} \lambda x.N : \varphi \to \psi}.$$

Again in order to show $\Gamma \models \lambda x.N : \varphi \to \psi$, let us take an arbitrary $\rho$ and suppose $\rho \models \Gamma$. Then we have to show that

$$\|\lambda x.N\|_\rho P \in \|\psi\| \text{ for all } P \in \|\varphi\|.$$

Suppose $P \in \|\varphi\|$. Then we can construct a new valuation

$$\rho'(y) = \begin{cases} P & , \ y = x \\ \rho(y) & , \ y \neq x. \end{cases}$$

Thus $\rho' \models \Gamma, x : \varphi$, since $\rho'(x) = P \in \|\varphi\|$. Then by the induction hypothesis $\rho' \models N : \psi$, i.e.,

$$\|N\|_{\rho'} \in \|\psi\|.$$

Since

$$\begin{aligned}\|\lambda x.N\|_\rho P &= \|\lambda x.N\|_{\rho'}\|x\|_{\rho'}\\ &= \|(\lambda x.N)x\|_{\rho'}\\ &\to_\beta \|N\|_{\rho'}\end{aligned}$$

and since $\|\psi\|$ is saturated, by Proposition 2.2.3. (iv), it follows that

$$\|\lambda x.N\|_\rho P \in \|\psi\|.$$

Hence $\rho \models \lambda x.N : \varphi \to \psi$.

**Case 4.** The last applied rule is $(\cap E)$, i.e.,

$$\frac{\Gamma \vdash_{\mathcal{D}} M : \varphi \cap \psi}{\Gamma \vdash_{\mathcal{D}} M : \varphi (M : \psi)}.$$

In order to show $\Gamma \models M : \varphi$ and $\Gamma \models M : \psi$, let us take an arbitrary $\rho$ and suppose that $\rho \models \Gamma$. Then by the induction hypothesis

$$\rho \models M : \varphi \cap \psi \ \text{, i.e., } \ \|M\|_\rho \in \|\varphi \cap \psi\|.$$

Thus $\|M\|_\rho \in \|\varphi\|$ and $\|M\|_\rho \in \|\psi\|$. Hence $\rho \models M : \varphi$ and $\rho \models M : \psi$.

**Case 5.** The last rule applied is $(\cap I)$, i.e.,

$$\frac{\Gamma \vdash_{\mathcal{D}} M : \varphi \quad \Gamma \vdash_{\mathcal{D}} M : \psi}{\Gamma \vdash_{\mathcal{D}} M : \varphi \cap \psi}.$$

If we proceed as in the previous cases and suppose that $\rho \models \Gamma$ for an arbitrary $\rho$, then by the induction hypothesis

$$\rho \models M : \varphi \ \text{ and } \ \rho \models M : \psi.$$

Thus $\|M\|_\rho \in \|\varphi\|$ and $\|M\|_\rho \in \|\psi\|$. It follows that $\|M\|_\rho \in \|\varphi\| \cap \|\psi\|$, i.e., $\rho \models M : \varphi \cap \psi$. □

The proof of soundness of $\mathcal{D}$ is a little more general than the proof of soundness of $\lambda \to$ and it differs in Cases 4 and 5. Once we have the soundness of $\mathcal{D}$, then, in order to obtain the strong normalization for $\mathcal{D}$, we proceed similarly as in the case of $\lambda \to$.

**Theorem 2.2.7. (Strong normalization for $\mathcal{D}$)**

*If $\Gamma \vdash_{\mathcal{D}} M : \sigma$, then $M$ is strongly normalizing.*

**Proof.** Suppose $\Gamma \vdash_{\mathcal{D}} M : \sigma$, then by Proposition 2.2.6 $\Gamma \models M : \sigma$, thus for each $\rho$

$$\rho \models \Gamma \quad \Rightarrow \quad \|M\|_{\rho} \in \|\sigma\|.$$

Define $\rho_0(x) = x$ for all term variables $x$. Then we have $\rho_0 \models \Gamma$ since by Proposition 2.2.3 $\|\tau\|$ is saturated for each type $\tau$, and hence $x \in \|\tau\|$. Therefore $\|M\|_{\rho_0} \in \|\sigma\|$, $\|M\|_{\rho_0} = M$ and $\|\sigma\| \subseteq SN$. So $M \in SN$. □

The proofs of Proposition 2.2.6 and Theorem 2.2.7 are in a sense dual to the proofs of the same properties presented in Krivine, 1990. In the forthcoming lines we shall try to explain what this duality is about. It is necessary to introduce various type interpretations as well in order to extend the proof of soundness of $\lambda \rightarrow$ to the proof of soundness of $\lambda 2$. This is done in Barendregt, 1992, and Girard et al., 1991, by taking various mappings $\xi$, which map type variables into saturated sets, thereby obtaining the type interpretation $\| \ \|_{\xi}$ from $\xi$ by setting

$$\|\alpha\|_{\xi} = \xi(\alpha), \quad \text{for all type variables } \alpha,$$

and proceeding as in Definition 2.2.1.

In the presented proof of soundness of $\mathcal{D}$, a unique mapping of type variables is given by

$$\xi(\alpha) = SN \quad \text{for all type variables } \alpha,$$

determining a unique type interpretation $\| \ \| : T \rightarrow \mathcal{P}(\Lambda)$. Actually, this means that the considered type system has one (ground) type variable and all the types are obtained from it by the two type constructors: implication and intersection. Also, there is a variety of term variable valuations $\rho : \mathrm{V} \rightarrow \Lambda$ which determine term valuations $\| \ \|_{\rho} : \Lambda \rightarrow \Lambda$ depending on $\rho$. And then, as we saw, the strong normalization for $\mathcal{D}$, Theorem 2.2.7, is proved by choosing a single term (variable) valuation

$$\rho_0(x) = x, \quad \|M\|_{\rho_0} = M.$$

In the proof of soundness of $\mathcal{D}$ given in Krivine, 1990, there is a variety of type variable interpretations $\xi$, which map type variables into saturated sets and the corresponding type interpretations $\| \ \|_\xi : T \to \mathcal{P}(\Lambda)$, as in the proof of soundness of $\lambda 2$ in Barendregt, 1992, mentioned above. On the other hand, there is a fixed term (variable) valuation

$$\rho(x) = x, \quad \text{for all term variables } x,$$

and hence a fixed term valuation $\| \ \| : \Lambda \to \Lambda$, for which $\|M\| = M$. This is dual to the proof of Proposition 2.2.6, were we had a single type interpretation and a variety of term valuations.

The soundness of $\mathcal{D}$ in Krivine, 1990, is given in the so called Lemme d'adequation.

**Proposition 2.2.8. (Lemme d'adequation)**
*If* $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma$ *and* $M_i \in \|\sigma_i\|_\xi$ *for each* $i, 0 \leq i \leq n$, *then* $M[M_1/x_1, \ldots, M_n/x_n] \in \|\sigma\|_\xi$.

The strong normalization for $\mathcal{D}$ in Krivine, 1990, is proved by choosing a single type (variable) interpretation

$$\xi_0(\alpha) = SN.$$

This is dual again to the choice of a single term valuation in the proof of Theorem 2.2.7.

The notion of computability, defined in Pottinger, 1980, is used in van Bakel, 1992, in order to prove the strong normalization theorem for $\lambda\cap_{-\omega}$. It seems doubtful that the strong normalisation for $\mathcal{D}$ can be proved via computability. One of the reasons to believe this is the fact that $\eta$-conversion is involved in the notion of computability, while it is a "property" of the rule $(\leq)$ as shown in Proposition 2.1.5. For the analysis of other proofs of strong normalization for the intersection type systems see van Bakel, 1992.

## 2.3 Typability of strongly normalizing terms

The strong normalization property holds for all eight systems of Barendregt's cube. We discussed in the previous Section that it holds for $\lambda\cap_{-\omega}$ and $\mathcal{D}$ as well. The very essential and outstanding property of the intersection type systems $\lambda\cap_{-\omega}$ and $\mathcal{D}$ is the converse of the strong normalization property, i.e., the fact that all strongly normalizing terms are typable in $\lambda\cap_{-\omega}$ and $\mathcal{D}$. Here we present a proof of this property in Theorem 2.3.6 and compare it with the proofs in Krivine, 1990, and van Bakel, 1992.

In order to show that every strongly normalizing term is typable in $\lambda\cap_{-\omega}$ and $\mathcal{D}$, first it is necessary to notice that every normal form is typable in these systems. The following characterization of the set of terms in normal form is given in Barendregt, 1984 (8.3.18).

**Theorem 2.3.1.** *The set of normal forms* $NF = \{M \in \Lambda | M$ *is a normal form* $\}$ *can be characterized inductively:*

*(i)* $x \in NF$ *for all term variables* $x$.

*(ii) If* $N_1, \ldots, N_k \in NF$, *then* $xN_1 \ldots N_k \in NF$.

*(iii) If* $M \in NF$, *then* $\lambda x.M \in NF$.

The consequence of this characterization is the following characterization of lambda terms in normal form:

**Proposition 2.3.2.** *If* $M$ *is a normal form, then*

$$M \equiv \lambda x_1 \ldots x_n . y N_1 \ldots N_k,$$

*for some normal forms* $N_1, \ldots, N_k$, *where* $n \geq 0$ *and* $k \geq 0$.

Now, we can show that every term in normal form can be typed in $\mathcal{D}$.

**Proposition 2.3.3.** *If* $M$ *is a normal form, then there is a basis* $\Gamma$ *and a type* $\sigma$ *such that*

$$\Gamma \vdash_{\mathcal{D}} M : \sigma.$$

**Proof.** By induction on the construction of a normal form $M$.
Let $M \equiv x$. Then for each type $\sigma$

$$x : \sigma \vdash_{\mathcal{D}} x : \sigma.$$

Let $M \not\equiv x$. Then by Proposition 2.3.2.

$$M \equiv \lambda x_1 \ldots x_n . y N_1 \ldots N_k,$$

for some normal forms $N_1, \ldots, N_k$, where $n \neq 0$ or $k \neq 0$. By the induction hypothesis there are bases $\Gamma_i$ and types $\sigma_i, 0 \leq i \leq k$, such that

$$\Gamma_i \vdash_{\mathcal{D}} N_i : \sigma_i \text{ for each } 1 \leq i \leq k.$$

Let $\Gamma' = \bigcap_{i=1}^{k} \Gamma_i$. If $\alpha$ is a fresh type variable for $\Gamma'$, then we can obtain the basis $\Delta$ from $\Gamma'$ in the following way

$$\Delta = \Gamma' \cap \{y : \sigma_1 \to \ldots \to \sigma_k \to \alpha\}.$$

Then

$$\Delta \vdash_{\mathcal{D}} y : \sigma_1 \to \ldots \to \sigma_n \to \alpha \text{ and } \Delta \vdash_{\mathcal{D}} N_i : \sigma_i, 1 \leq i \leq k,$$

so by $(\to E)$

$$\Delta \vdash_{\mathcal{D}} y N_1 \ldots N_k : \alpha.$$

If $(x_j : \varphi_j) \in \Delta$ for all $1 \leq j \leq n$, then we can obtain the basis $\Gamma$ from $\Delta$ in the following way:

$$\Gamma = \Delta \backslash \{x_j : \varphi_j | 1 \leq j \leq n\},$$

and so

$$\Gamma \vdash_{\mathcal{D}} \lambda x_1 \ldots x_n . y N_1 \ldots N_k : \varphi_1 \to \ldots \to \varphi_n \to \alpha,$$

i.e., $\Gamma \vdash_{\mathcal{D}} M : \varphi_1 \to \ldots \to \varphi_n \to \alpha$ when $y \neq x_i$, for all $1 \leq i \leq n$, or

$$\Gamma \vdash_{\mathcal{D}} \lambda x_1 \ldots x_n . y N_1 \ldots N_k : \varphi_1 \to \ldots \to \varphi_{i-1} \to$$

$$\bigcap_{i=1}^{k}$$

Hence, $\Gamma \vdash_{\mathcal{D}} M : \varphi_1 \to \ldots \to \varphi_{i-1} \to (\bigcap_{j=1}^{k} \psi_j \cap (\sigma_1 \to \ldots \to \sigma_k \to \alpha)) \to \varphi_{i+1} \to \ldots \to \varphi_n \to \alpha$, when $y = x_i$ for some $1 \leq i \leq n$ and $\Gamma' \vdash_{\mathcal{D}} y : \psi_j$, for all $1 \leq j \leq k$.

If there is a variable $x_\ell$, $1 \leq \ell \leq n$ which is not free in $yN_1 \ldots N_k$, then again

$$\Gamma \vdash_{\mathcal{D}} \lambda x_1 \ldots x_\ell \ldots x_n . yN_1 \ldots N_k : \varphi_1 \to \ldots \to \varphi_\ell \to \ldots \to \varphi_n \to \alpha,$$

where $\varphi_\ell$ is an arbitrary type. $\Box$

There is another problem that is to be overcome in order to show that every strongly normalizing term is typable in $\mathcal{D}$. It is known that $\mathcal{D}$ is closed under $\beta$ reduction, but not under $\beta$ expansion, i.e., types are preserved under $\beta$-reduction, but it is not the case under $\beta$-expansion. The counter example given in Krivine, 1990, takes into account the terms $\lambda y.(\lambda x.y)(yy)$ and $\lambda y.y$ since

$$\lambda y.(\lambda x.y)(yy) \to_\beta \lambda y.y$$

and $\vdash_{\mathcal{D}} \lambda y.y : \alpha \to \alpha$, where $\alpha$ is a type variable. It is shown that if we suppose

$$\vdash_{\mathcal{D}} \lambda y.(\lambda x.y)(yy) : \alpha \to \alpha,$$

then

$$y : \alpha \vdash_{\mathcal{D}} (\lambda x.y)(yy) : \alpha,$$

so

$$y : \alpha \vdash_{\mathcal{D}} (\lambda x.y) : \varphi \to \alpha' \text{ and } y : \alpha \vdash yy : \varphi,$$

for a certain type $\varphi$ and $\alpha' = \alpha \cap \psi_1 \cap \ldots \cap \psi_k$. Thus

$$y : \alpha \vdash_{\mathcal{D}} y : \rho \to \varphi' \text{ and } y : \alpha \vdash_{\mathcal{D}} y : \rho$$

for a certain type $\rho$ and for $\varphi' = \varphi \cap \varphi_1 \cap \ldots \cap \varphi_\ell$. And this is actually a contradiction, since it is not possible to derive an arrow type for a term variable from a type variable. The term variable $y$ has to be an arrow type in order to type self-application $yy$ in $\mathcal{D}$. This problem is avoided in $\mathcal{D}\Omega$, since $yy$ can be typed by $\omega$.

But still, the term $\lambda y.(\lambda x.y)(yy)$ is typable in $\mathcal{D}$ because there exists the following derivation in $\mathcal{D}$:

$$\dfrac{\dfrac{\dfrac{y:\alpha\cap(\alpha\to\beta)\vdash y:\alpha\to\beta \quad y:\alpha}{y:\alpha\cap(\alpha\to\beta)\vdash yy:\beta} \qquad \dfrac{x:\beta, y:\alpha\cap(\alpha\to\beta)\vdash y:\alpha\to\beta\ (\alpha)}{y:\alpha\cap(\alpha\to\beta)\vdash \lambda x.y:\beta\to(\alpha\to\beta)\ (\beta\to\alpha)}\,(\to I)}{\dfrac{y:\alpha\cap(\alpha\to\beta)\vdash(\lambda x.y)(yy):\alpha\to\beta\ (\alpha)}{y:\alpha\cap(\alpha\to\beta)\vdash(\lambda x.y)(yy):\alpha\cap(\alpha\to\beta)}\,(\cap I)}}{\vdash \lambda y.(\lambda x.y)(yy):(\alpha\cap(\alpha\to\beta))\to(\alpha\cap(\alpha\to\beta))}\,(\to I)$$

Thus both of the considered terms are typable in $\mathcal{D}$, i.e., $\vdash_{\mathcal{D}} \lambda y.y:\alpha\to\alpha$ and $\vdash_{\mathcal{D}} \lambda y.(\lambda x.y)(yy):(\alpha\cap(\alpha\to\beta))\to(\alpha\cap(\alpha\to\beta))$, but $\lambda y.(\lambda x.y)(yy):\alpha\to\alpha$ cannot be derived without the rule $(\omega)$.

The counter example in van Bakel, 1992, deals with the terms $\lambda yz.(\lambda x.z)(yz)$ and $\lambda yz.z$ typable in $\mathcal{D}$, since

$$\lambda yz.(\lambda x.z)(yz)\to_\beta \lambda yz.z$$

and $\vdash_{\mathcal{D}} \lambda yz.z:\beta\to(\alpha\to\alpha)$ and $\vdash_{\mathcal{D}} \lambda yz.(\lambda x.z)(yz):(\alpha\to\beta)\to(\alpha\to\alpha)$. Again, $\lambda yz.(\lambda x.z)(yz)$ cannot be typed with $\beta\to(\alpha\to\alpha)$ without the rule $(\omega)$ because $y$ has to be of an arrow type in order to type the application $yz$ in $\mathcal{D}$.

Since each normal form is typable in $\mathcal{D}$ we need some property that "pastes together" the steps of $\beta$-reduction in a reduction tree of a strongly normalizing term with the steps in the type assignment. This is done by the following statements proved in Krivine, 1990, and van Bakel, 1992:

**Proposition 2.3.4.** *Let* $\Gamma\vdash_{\mathcal{D}} M[N/x]:\sigma$. *If* $N$ *is typable in the basis* $\Gamma$, *then*

$$\Gamma\vdash_{\mathcal{D}}(\lambda x.M)N:\sigma.$$

The consequence of Proposition 2.3.4 is the following property of type preservation under $\beta$-expansion.

**Proposition 2.3.5.** *Let* $\Gamma\vdash_{\mathcal{D}} M[S[N/x]]:\sigma$. *If* $N$ *is typable in the basis* $\Gamma$, *then*

$$\Gamma\vdash_{\mathcal{D}} M[(\lambda x.S)N]:\sigma.$$

The main point in both Proposition 2.3.4 and 2.3.5 is the fact that $N$ is typable in $\mathcal{D}$ in the given basis. In the previous counter examples subterms

$yy$ and $yz$ were not typable in $\mathcal{D}$ in the given bases. That is the reason why those $\beta$–expansions do not preserve types.

**Theorem 2.3.6.** (Converse of the Strong Normalization Theorem)

*If $M$ is strongly normalizing, then there is a basis $\Gamma$ and a type $\sigma$ such that*

$$\Gamma \vdash_{\mathcal{D}} M : \sigma.$$

**Proof.** By induction on the construction of $M$.

**Case 1.** $M$ is a term variable, $M \equiv x$, so it is a normal form. We have

$$x : \sigma \vdash_{\mathcal{D}} x : \sigma$$

for each type $\sigma$.

**Case 2.** Let $M \equiv PQ$. Then we shall prove the statement by induction on the size of reduction of $M$, i.e., the sum of the lengths of all possible reductions of $M$, as defined in Krivine, 1990, notation $n(M)$.

**Subcase 1.** $n(M) = 0$, then $M$ is a normal form. Hence by Proposition 2.3.3 there is a basis $\Gamma$ and a type $\sigma$ such that

$$\Gamma \vdash_{\mathcal{D}} M : \sigma.$$

**Subcase 2.** If $n(M) > 0$, i.e., $M$ is not a normal form, then it has a subterm of the form $(\lambda x.S)N$, i.e., $M[(\lambda x.S)N]$. Thus $(\lambda x.S)N$ can be a subterm of $P$, $Q$, or both. Say it is of $P$, so

$$M[(\lambda x.S)N] \equiv P[(\lambda x.S)N]Q.$$

This term can be $\beta$–reduced to $P[S[N/x]]Q$ for which $n(M) > n(P[S[N/x]]Q)$, and $n(M) > n(N)$.

By the induction hypothesis

$$\Gamma' \vdash_{\mathcal{D}} P[S[N/x]]Q : \sigma \text{ and } \Gamma'' \vdash_{\mathcal{D}} N : \tau.$$

Take $\Gamma = \Gamma' \cap \Gamma''$, then by Proposition 2.3.5

$$\Gamma \vdash_{\mathcal{D}} P[(\lambda x.S)N]Q : \sigma.$$

We proceed similarily in the cases when $(\lambda x.S)N$ is a subterm of $Q$, or of both $P$ and $Q$.

**Case 3.** Let $M \equiv \lambda x.P$. The induction hypothesis holds for $P$; so there is a basis $\Delta$ and a type $\tau$ such that

$$\Delta \vdash_{\mathcal{D}} P : \tau.$$

If $x$ is not a fresh variable for $\Delta$, then

$$\Delta \backslash \{x : \rho\} \vdash_{\mathcal{D}} \lambda x.P : \rho \rightarrow \tau.$$

If $x$ is a fresh variable for $\Delta$, then

$$\Delta \vdash_{\mathcal{D}} \lambda x.P : \rho \rightarrow \tau$$

where $\rho$ is an arbitrary type. $\square$

This converse of the strong normalization property is proved in van Bakel, 1992, by using "the inside–out reduction" of lambda terms. The inductive argument in Krivine, 1990, is the sum of all possible reductions of a term, while using the property that each term can be expressed in the form

$$\lambda x_1 \dots x_n.NM_1 \dots M_k,$$

where $N$ is a variable or a redex.

The proof suggested by R. Statman is by induction on the argument (length of term, size of reduction tree) ordered lexicographically. For the analysis of other proofs of this property see van Bakel, 1992.

The consequence of Theorem 2.2.7 and 2.3.6 is that the strongly normalising terms are exactly the terms typable in $\mathcal{D}$.

**Corollary 2.3.7.** *Let $M$ be a lambda term.*

*There are $\Gamma$ and $\sigma$ such that $\Gamma \vdash_{\mathcal{D}} M : \sigma$ if and only if $M$ is strongly normalizing.*

The systems $\mathcal{D}$ and $\lambda\cap_{-\omega}$ are equivalent from the point of view of typability, as shown in Theorem 2.1.3, so the same property holds for the system $\lambda\cap_{-\omega}$

**Corollary 2.3.8.** *Let $M$ be a lambda term.*

*There are $\Gamma$ and $\sigma$ such that $\Gamma \vdash_{-\omega} M : \sigma$ if and only if $M$ is strongly normalizing.*

A consequence of this equivalence is the undecidability of the typability in the systems $\mathcal{D}$ and $\lambda\cap_{-\omega}$, since the set of strongly normalizing terms, SN, is not recursive.

# 3 Inhabitation in intersection and union type assignment systems

In this Chapter we shall discuss the inhabitation in the intersection and union type assignment systems. These systems are introduced in Barbanera et al., 1991, as extensions of intersection type systems with union types and corresponding inference rules. The problem of the inhabitation in a type system is whether there is a term of a given type, $? : \sigma$ (see Section 1.3). The problem of inhabitation in the intersection type assignment systems is still open. There are various approaches toward the problem of inhabitation. The Curry-Howard isomorphism between formulae as types and lambda terms as proofs does not hold for the intersection type systems. There are attempts in Lopez-Escobar, 1985, and Pottinger, 1980, to give the logical "meaning" of the intersection. The relation between the provable realizability and the inhabitation intersection type systems is investigated in Alessi et al., 1991, and Mints, 1989.

The intersection type assignment system for combinators is given in Dezani et al., 1992. It turned out that it is possible to give an equivalent Church version of combinators with intersection types. This typed version of combinators with intersection types is formulated in Venneri, 1992, together with the Hilbert-style logical system which corresponds to it by the Curry–Howard isomorphism. This result gives some new hope that some of the attempts to find a logical system corresponding to the lambda calculus with intersection types will be fruitful.

The extension $\lambda\cap\cup$ of $\lambda\cap$ with union types and the corresponding rules of the type assignment system are introducedd and investigated in Barbanera et al., 1991. Terms typable in $\lambda\cap\cup$ are exactly the terms typable in $\lambda\cap$, while the set of inhabited types in $\lambda\cap\cup$ is larger that the set of inhabited types in $\lambda\cap$ (see Barbanera et al., 1991).

Section 3.1 presents a brief overview of union type assignment systems. In Section 3.2 we shall consider the inhabitation in $\lambda\cap\cup$ versus provability in intuitionistic propositional logic with conjunction and disjunction. In Hindley, 1984, it is shown in a syntactical way that types inhabited in $\lambda\cap$

do not correspond to formulae provable in intuitionistic propositional logic. We shall give a semantical proof of this fact. In Section 3.3 we shall link the inhabitation in the given type assignment systems and the inhabitation in the extension of the simply typed lambda calculus with conjunctive and disjunctive types.

## 3.1 Basic systems with intersection and union types

Let us recall some basic notions and notations of intersection and union type assignment systems which are given in Barbanera et al., 1991.

The types $T$ of $\lambda\cap$ are propositional formulae with connectives $\rightarrow$ and $\cap$ given in Definition 1.3.1. The type forming operator $\cap$ is a *specific conjunction* whose properties are in accordance with its interpretation as intersection of types.

The set of types $T_\cup$ of $\lambda\cap\cup$ is defined in the following way:

**Definition 3.1.1.**

*(i)* $V = \{\alpha, \beta, \gamma, \alpha_1, \ldots\} \subset T_\cup$ *($V$ is a denumerable set of propositional variables).*

*(ii)* $\omega \in T_\cup$.

*(iii)* *If* $\sigma, \tau \in T_\cup$, *then* $(\sigma \rightarrow \tau) \in T_\cup$.

*(iv)* *If* $\sigma, \tau \in T_\cup$, *then* $(\sigma \cap \tau) \in T_\cup$.

*(v)* *If* $\sigma, \tau \in T_\cup$, *then* $(\sigma \cup \tau) \in T_\cup$.

Actually the types $T$ of $\lambda\cap$ are the restriction of $T\cup$ to the types with $\cap$ and $\rightarrow$. Let $\alpha, \beta, \gamma, \alpha_1, \ldots$ be schematic letters for type variables, and let $\sigma, \tau, \varphi, \psi, \sigma_1, \ldots$ be schematic letters for types. We shall use the usual convention for omitting parentheses according to the precedence rule: "$\cap$ and $\cup$ over $\rightarrow$".

**Definition 3.1.2.** *A pre-order $\leq$ on $T_\cup$ is an extension of the pre-order $\leq$ on $T$ (given in Definition 1.3.2) obtained by adding*

1. $\sigma \leq \sigma \cup \tau$ , $\tau \leq \sigma \cup \tau$
2. $\sigma \cup \sigma \leq \sigma$
3. $\sigma \leq \rho, \tau \leq \rho \Rightarrow \sigma \cup \tau \leq \rho$
4. $\sigma \cap (\tau \cup \rho) \leq (\sigma \cap \tau) \cup (\sigma \cap \rho)$

**Definition 3.1.3.** *The type assignment system $\lambda \cap \cup$ is the extension of $\lambda\cap$ with*

$$(\cup E) \quad \frac{\Gamma, x : \sigma \vdash_{\lambda\cap\cup} M : \rho \quad \Gamma, x : \tau \vdash_{\lambda\cap\cup} M : \rho \quad \Gamma \vdash_{\lambda\cap\cup} N : \sigma \cup \tau}{\Gamma \vdash_{\lambda\cap\cup} M[N/x] : \rho}$$

$$(\cup I) \quad \frac{\Gamma \vdash_{\lambda\cap\cup} M : \sigma}{\Gamma \vdash_{\lambda\cap\cup} M : \sigma \cup \tau} \quad , \quad \frac{\Gamma \vdash_{\lambda\cap\cup} M : \tau}{\Gamma \vdash_{\lambda\cap\cup} M : \sigma \cup \tau} \quad ;$$

*(The rule $(\leq)$ is taking into account the pre-order $\leq$ on $T_\cup$).*

The system $\mathcal{D}\Omega\cup$ is obtained from $\lambda \cap \cup$ by leaving out $(\leq)$, while $\mathcal{D}\cup$ is given without rules $(\omega)$ and $(\leq)$ (we write $\vdash_{\mathcal{D}\cup}$ for the corresponding turnstile).

The difference between the special conjunction $\cap$, called *intersection*, and the arbitrary propositional conjunction $\wedge$ is in the rule $(\cap I)$. In order to show that the term $M$ has the intersection type $\sigma \cap \tau$ in the basis $\Gamma$ it is necessary to show that $M$ has both types $\sigma$ and $\tau$ in the basis $\Gamma$, and it has to be so if $\cap$ is meant to represent intersection. $M$ is the same in the conclusion as in both premises of the rule $(\cap I)$. So it is in the rule $(\cap E)$. Thus in these two steps $M$ remains the same although the deduction grows and the lambda terms do not correspond to the deductions. With the usual propositional conjunction $\wedge$ we have that if $N$ and $P$ are two different terms of type $\sigma$ and $\tau$, respectively, in the basis $\Gamma$, then it is possible to obtain from them a term of type $\sigma \wedge \tau$ in the basis $\Gamma$ (see the rule $(\wedge - I)$ below in Definition 3.2.5). Similarly is with the rule $(\wedge - E)$. Thus the lambda terms correspond to the deductions.

Something similar happens with the special disjunction $\cup$, further on called *union* and the arbitrary propositional disjunction $\vee$. Although the main difference in this case is in the elimination rules. In order to eliminate $\sigma \cup \tau$, i.e., to eliminate the basic statements with the same subjects and different predicates $\sigma$ and $\tau$ from two bases which differ only in these basic statements it is necessary to assign the same type to the same term in both of these bases. In the case of usual disjunction $\vee$ the elimination can be done if the same type is assigned to two different terms in both of the bases (see the rule $(\vee - E)$ below in Definiiton 3.2.5). Again, the lambda terms do not correspond to the deduction in $(\cup I)$ either. The logical framework of unioin types is discussed in Dezani et al., 1992.

## 3.2 Inhabitation and provability

We consider the inhabitation in $\lambda\cap\cup$ versus provability in the intuitionistic propositional logic with conjunction and disjunction. We prove in Proposition 3.2.8 that inhabitation implies provability. The other way round does not hold, since intersection types inhabited in $\lambda\cap$ do not correspond to formulae provable in intuitionistic propositional logic. There are provable but not inhabited formulae, as shown in Hindley, 1984. We present a semantic explanation of this fact.

### Type Interpretations

Starting from any model of the untyped lambda calculus (lambda model) $\mathcal{M} =< D, \cdot, \| \quad \|>$ an interpretation of types for $\lambda\cap$ is constructed in Barendregt et al., 1983, in the following way:

**Definition 3.2.1.** *If $v$ is a interpretation of type variables in $\mathcal{P}(D)$*

$$v : V \rightarrow \mathcal{P}(D) = \{X, X \subseteq D\},$$

*then $\| \sigma \|_v^{\mathcal{M}} \in \mathcal{P}(D)$ the interpretation of $\sigma \in T$ in $\mathcal{M}$ via $v$ is defined as follows*

*1.* $\| \alpha \|_v^{\mathcal{M}} = v(\alpha)$,

2. $\| \sigma \to \tau \|_v^{\mathcal{M}} = \| \sigma \|_{D} \| \tau \| = \{d \in D. \forall e \in \| \sigma \|_v^{\mathcal{M}} \; d \cdot e \in \| \tau \|_v^{\mathcal{M}}\}$,

3. $\| \sigma \cap \tau \|_v^{\mathcal{M}} = \| \sigma \|_v^{\mathcal{M}} \cap \| \tau \|_v^{\mathcal{M}}$.

4. $\| \omega \|_v^{\mathcal{M}} = D$.

5. $\| \sigma \cup \tau \|_v^{\mathcal{M}} = \| \sigma \|_v^{\mathcal{M}} \cup \| \tau \|_v^{\mathcal{M}}$ .

The satisfaction relation $\models$ is defined as in Chapter 1.

**Definition 3.2.2.**

(i) $\mathcal{M}, \rho, v \models N : \sigma \quad \Leftrightarrow \quad \| N \|_\rho^{\mathcal{M}} \in \| \sigma \|_v^{\mathcal{M}}$ .

(ii) $\mathcal{M}, \rho, v \models \Gamma \quad \Leftrightarrow \quad \mathcal{M}, \rho, v \models x : \tau$ *for all* $x : \tau$ *in* $\Gamma$.

(iii) $\Gamma \models N : \sigma \quad \Leftrightarrow \quad (\forall \mathcal{M}, \rho, v \models \Gamma \Rightarrow \mathcal{M}, \rho, v \models N : \sigma)$.

The interpretation of the relation $\leq$ on types will naturally be the relation $\subseteq$ on the subsets of $D$. This kind of model is good enough to prove the soundness result for $\lambda\cap$ and $\lambda \cap \cup$.

**Lemma 3.2.3.**(Barendregt et al., 1983, Barbanera et al., 1991.) *If* $\sigma \leq \tau$, *then* $\| \sigma \|_v^{\mathcal{M}} \subseteq \| \tau \|_v^{\mathcal{M}}$, *for all* $\mathcal{M}$ *and* $v$.

**Proof.** By induction on the length of the derivation of $\sigma \leq \tau$. □

**Proposition 3.2.4. (Soundness)** (Barendregt et al., 1983, Barbanera et al., 1991.)

$$\textit{If} \quad \Gamma \vdash_{\lambda\cap\cup} M : \sigma, \quad \textit{then} \quad \Gamma \models M : \sigma.$$

**Proof.** By induction on the derivation in $\lambda\cap$ and $\lambda \cap \cup$ using Lemma 3.2.3. □

In Section 1.3 we discussed filter models which are introduced in Barendregt et al., 1983, in order to prove the competeness results of $\lambda\cap$. The completeness of $\lambda\cap$ is proved in Hindley, 1982, by using the term model of the lambda calculus. Moreover, the completeness of $\lambda \cap \cup$ is proved in Barbanera et al., 1991, also using the term model.

## Connections between $\lambda\cap$, $\lambda \cap \cup$ and Intuitionistic Logic

The intuitive difference between the intersection $\cap$ and the conjunction $\wedge$ was mentioned above. Now, we are going to make it more precise by relating derivations in $\lambda\cap$ and provability in the fragment of propositional intuitionistic logic $H$ with implication $\rightarrow$, conjunction $\wedge$ and truth $\top$ on the one hand, while on the other, derivations in $\lambda\cap\cup$ are going to be related with provability in propositional intuitionistic logic with $\rightarrow, \wedge, \top$ and disjunction $\vee$.

First, let us recall the natural deduction formulation of intuitionistic propositional logic $H$ given in Prawitz, 1965. Actually we are going to consider the fragment of the intuitionistic propositional logic $H$, dealing with implication $\rightarrow$, conjunction $\wedge$, truth $\top$ and disjunction $\vee$.

**Definition 3.2.5**

*(i) The propositional language $L$ consists of a denumerable set of propositional variables $V = \{\alpha, \beta, \gamma, \alpha_1, \ldots\}$ and a constant $\top$. Formulae of the language are built up in the following way:*

- *all propositional variables and the constant $\top$ are formulae;*
- *if $\varphi$ and $\psi$ are formulae, then $\varphi \rightarrow \psi$, $\varphi\wedge\psi$ and $\varphi\vee\psi$ are formulae.*

*(ii) The natural deduction system $H$ is given by the following rules:*

$$(MP)\quad \frac{\sigma \rightarrow \rho \quad \sigma}{\rho}\,; \quad (DT)\quad \frac{\begin{array}{c}[\sigma]\\ \vdots \\ \rho\end{array}}{\sigma \rightarrow \rho}\,;$$

$$(\wedge - E)\quad \frac{\sigma \wedge \rho}{\sigma}\,,\ \frac{\sigma \wedge \rho}{\rho}\,; \quad (\wedge - I)\quad \frac{\sigma \quad \rho}{\sigma \wedge \rho}\,;$$

$$(\top)\ \frac{}{\top}\ ;$$

$$(\vee - E)\ \frac{\begin{matrix}[\sigma] & [\tau] \\ \vdots & \vdots \\ \rho & \rho\end{matrix} \quad \sigma \vee \tau}{\rho}\ ,\ (\vee - I)\ \frac{\sigma}{\sigma \vee \rho}\ ,\ \frac{\rho}{\sigma \vee \rho}.$$

*(iii) The system $H \rightarrow$ is the subsystem of $H$ given by the rules $(MP)$ and $(DT)$. Obviously, the language of $H \rightarrow$ consists only of implicational formulae. Similarly, $H_{\overline{\wedge}}$ is the extension of $H \rightarrow$ by $(\wedge E)$ and $(\wedge I)$.*

The notation $\varphi_1, \ldots, \varphi_n \vdash \varphi(H)$ means that $\varphi$ is derived in $H$ from the assumptions $\varphi_1, \ldots, \varphi_n$, while $\varphi(H)$ stands for $\vdash \varphi(H)$.

Now, let us set a mapping from the set of intersection and union types into the set of propositional formulae that is replacing each occurrence of $\cap$ and $\cup$ in a type by $\wedge$ and $\vee$, respectively. The image of $\omega$ is the intuitionistic truth.

**Definition 3.2.6.**

*(i)* $\alpha^{\wedge} = \alpha$

*(ii)* $(\sigma \rightarrow \tau)^{\wedge} = \sigma^{\wedge} \rightarrow \tau^{\wedge}$

*(iii)* $(\sigma \cap \tau)^{\wedge} = \sigma^{\wedge} \wedge \tau^{\wedge}$

*(iv)* $\omega^{\wedge} = \top$

*(v)* $(\sigma \cup \tau)^{\wedge} = \sigma^{\wedge} \vee \tau^{\wedge}$

*(vi)* *If $\Gamma$ is a basis $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ then $\Gamma^{\wedge}$ is $x_1 : \sigma_1^{\wedge}, \ldots, x_n : \sigma_n^{\wedge}$.*

Further, it is possible to connect the partial ordering on $T$ and $T_{\cup}$ and provability in $H$.

**Proposition 3.2.7.** *If $\sigma \leq \tau$, then $\sigma^\wedge \to \tau^\wedge (H)$.*

**Proof.** By induction on the derivation of $\sigma \leq \tau$.

1. $\sigma \leq \sigma$, then $\sigma^\wedge \to \sigma^\wedge (H)$.

2. $\dfrac{\sigma \leq \tau \quad \tau \leq \rho}{\sigma \leq \rho}$, then by the induction hypothesis $\sigma^\wedge \to \tau^\wedge (H)$, $\tau^\wedge \to \rho^\wedge (H)$ and there is the following derivation in $H$:

$$\dfrac{\dfrac{\dfrac{\sigma^\wedge \to \tau^\wedge \quad [\sigma^\wedge]^1}{\tau^\wedge}(MP) \quad \tau^\wedge \to \rho^\wedge}{\rho^\wedge}(MP)}{\sigma^\wedge \to \rho^\wedge}{}^1(DT) .$$

3. $\sigma \leq \omega$, then $\sigma^\wedge \to \top \ (H)$ by $(\top)$ and $(DT)$.

4. $\omega \leq \omega \to \omega$, then $\top \to (\top \to \top) \ (H)$.

5. $(\sigma \to \tau) \cap (\sigma \to \rho) \leq (\sigma \to (\tau \cap \rho))$, then there is the following derivation in $H$:

$$\dfrac{\dfrac{\dfrac{\dfrac{{}^1[\sigma^\wedge] \quad \dfrac{{}^2[(\sigma^\wedge \to \tau^\wedge) \wedge (\sigma^\wedge \to \rho^\wedge)]}{\sigma^\wedge \to \tau^\wedge}(\wedge E)}{\tau^\wedge}(MP) \quad \dfrac{{}^1[\sigma^\wedge] \quad \dfrac{{}^2[(\sigma^\wedge \to \tau^\wedge) \wedge (\sigma^\wedge \to \rho^\wedge)]}{\sigma^\wedge \to \rho^\wedge}(\wedge E)}{\rho^\wedge}(MP)}{\tau^\wedge \wedge \rho^\wedge}}{\sigma^\wedge \to (\tau^\wedge \wedge \rho^\wedge)}{}^1(DT)}{(\sigma^\wedge \to \tau^\wedge) \wedge (\sigma^\wedge \to \rho^\wedge) \to \sigma^\wedge \to (\tau^\wedge \wedge \rho^\wedge)}{}^2(DT)$$

6. $\sigma \cap \tau \leq \sigma$, then $\sigma^\wedge \wedge \tau^\wedge \to \sigma^\wedge \ (H)$ by $(\wedge - E)$ and $(DT)$.
   $\sigma \cap \tau \leq \tau$, then $\sigma^\wedge \wedge \tau^\wedge \to \tau^\wedge \ (H)$ by $(\wedge - E)$ and $(DT)$.

7. $\dfrac{\sigma \leq \rho \quad \sigma \leq \tau}{\sigma \leq \rho \cap \tau}$, then by the induction hypothesis $\sigma^\wedge \to \rho^\wedge (H)$ and $\sigma^\wedge \to \tau^\wedge (H)$ and the following can be derived in $H$:

$$\dfrac{\dfrac{{}^1[\sigma^\wedge]\ \ \sigma^\wedge \to \rho^\wedge}{\rho^\wedge}(MP) \qquad \dfrac{{}^1[\sigma^\wedge]\ \ \sigma^\wedge \to \tau^\wedge}{\tau^\wedge}(MP)}{\dfrac{\rho^\wedge \wedge \tau^\wedge}{\sigma^\wedge \to \rho^\wedge \wedge \tau^\wedge}{}^1(DT)}(\wedge - I)$$

8. $\dfrac{\sigma \leq \tau \quad \rho \leq \varphi}{\tau \to \rho \leq \sigma \to \varphi}$ by the induction hypothesis $\sigma^\wedge \to \tau^\wedge, \rho^\wedge \to \varphi^\wedge$ $(H)$

and the following derivation can be performed in $H$:

$$\dfrac{\rho^\wedge \to \varphi^\wedge \qquad \dfrac{[\tau^\wedge \to \rho^\wedge]^2 \qquad \dfrac{\sigma^\wedge \to \tau^\wedge \quad [\sigma^\wedge]^1}{\tau^\wedge}(MP)}{\rho^\wedge}(MP)}{\dfrac{\dfrac{\varphi^\wedge}{\sigma^\wedge \to \varphi^\wedge}{}^1(DT)}{(\tau^\wedge \to \rho^\wedge) \to (\sigma^\wedge \to \varphi^\wedge)}{}^2(DT)}\ (MP)\,.$$

9. $\sigma \leq \sigma \cup \tau$, then $\sigma^\wedge \to \sigma^\wedge \vee \tau^\wedge (H)$ by $(\vee - I)$ and $(DT)$.

10. $\sigma \cup \sigma \leq \sigma$, then $\sigma^\wedge \vee \sigma^\wedge \to \sigma^\wedge (H)$ by $(\vee - E)$ and $(DT)$.

11. $\dfrac{\sigma \leq \rho \ \tau \leq \rho}{\sigma \cup \tau \leq \rho}$, then by the induction hypothesis $\sigma^\wedge \to \rho^\wedge$, $\tau^\wedge \to \rho^\wedge (H)$ and there is the following derivation in $H$:

$$\dfrac{\dfrac{\sigma^\wedge \to \rho^\wedge \quad [\sigma^\wedge]}{\rho^\wedge}(MP) \qquad \dfrac{\tau^\wedge \to \rho^\wedge \quad [\tau^\wedge]}{\rho^\wedge}(MP) \qquad [\sigma^\wedge \vee \tau^\wedge]^1}{\dfrac{\rho^\wedge}{\sigma^\wedge \vee \tau^\wedge \to \rho^\wedge}{}^1\ (DT)}\ (\vee - E)\,.$$

12. $\sigma \cap (\tau \cup \rho) \leq (\sigma \cap \tau) \cup (\sigma \cap \rho)$, then there is the following derivation in $H$

$$\dfrac{\dfrac{\dfrac{[\sigma^{\wedge} \wedge (\tau^{\wedge} \vee \rho^{\wedge})]^1}{\tau^{\wedge} \vee \rho^{\wedge}}\,(\wedge E) \quad \dfrac{\dfrac{\sigma^{\wedge} \quad [\tau^{\wedge}]}{\sigma^{\wedge} \wedge \tau^{\wedge}}\,(\wedge I)}{(\sigma^{\wedge} \wedge \tau^{\wedge}) \vee (\sigma^{\wedge} \wedge \rho^{\wedge})} \quad \dfrac{\dfrac{\sigma^{\wedge} \quad [\rho^{\wedge}]}{\sigma^{\wedge} \wedge \rho^{\wedge}}\,(\wedge I)}{(\sigma^{\wedge} \wedge \tau^{\wedge}) \vee (\sigma^{\wedge} \wedge \rho^{\wedge})}\,(\vee I)}{(\sigma^{\wedge} \wedge \tau^{\wedge}) \vee (\sigma^{\wedge} \wedge \rho^{\wedge})}\,(\vee E)}{\sigma^{\wedge} \wedge (\tau^{\wedge} \vee \rho^{\wedge}) \rightarrow (\sigma^{\wedge} \wedge \tau^{\wedge}) \vee (\sigma^{\wedge} \wedge \rho^{\wedge})}\,{}^{1}\,(DT)\,. \quad \square$$

Further on we are going to write $\sigma$ instead of the formula $\sigma^{\wedge}$, when this is not ambiguous but although the notation will be the same, we are going to keep in mind that the formula $\sigma$ of the propositional language is obtained from the type $\sigma \in T$ by replacing each occurrence of $\cap$ by $\wedge, \cup$ by $\vee$, and $\omega$ by $\top$ and vice versa.

The Curry-Howard isomorphism given by Howard, 1980, is the one-to-one correspondence between types inhabited in the simply typed lambda calculus, $\lambda \rightarrow$ and formulae provable in $H \rightarrow$. We shall come back to this property in Section 3.3. This isomorphism can be extended in order to link inhabitation in $\lambda\cap$ and $\lambda \cap \cup$ and provability in $H_{\bar{\wedge}}$ and $H$, respectively.

**Proposition 3.2.8.** *Let $\tau \in T$ be a type. If there exists a lambda term $M$ such that*

$$x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash_{\lambda\cap\cup} M : \tau, \ \textit{ then } \ \sigma_1, \ldots, \sigma_n \vdash \tau(H).$$

**Proof.** By induction on the derivation of $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ in $\lambda \cap \cup$. Let $\Gamma$ denote the basis $x_1 : \sigma_1, \ldots, x_n : \sigma_n$. If the last step in the derivation is:

- **(start rule)** $\dfrac{}{x_1 : \sigma_1, \ldots, x_i : \sigma_i, \ldots, x_n : \sigma_n \vdash x_i : \sigma_i}$, then obviously

$$\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n \vdash \sigma_i(H);$$

- **($\rightarrow$ E)** $\dfrac{\Gamma \vdash N : \tau \rightarrow \rho \qquad \Gamma \vdash P : \tau}{\Gamma \vdash NP : \rho}$, then by the induction hypothesis we have in $H$ $\sigma_1, \ldots, \sigma_n \vdash \tau \rightarrow \rho$ and $\sigma_1, \ldots, \sigma_n \vdash \tau$, so $\sigma_1, \ldots, \sigma_n \vdash \rho$ by $(MP)$.

- $(\rightarrow \mathrm{I})$ $\dfrac{\Gamma, x : \tau \vdash N : \rho}{\Gamma \vdash \lambda x.N : \tau \rightarrow \rho}$, then by the induction hypothesis we have in $H$
$\sigma_1, \ldots, \sigma_n, \tau \vdash \rho$. Thus $\sigma_1, \ldots, \sigma_n \vdash \tau \rightarrow \rho$ follows by $(DT)$.

- $(\cap \mathrm{E})$ $\dfrac{\Gamma \vdash N : \tau \cap \rho}{\Gamma \vdash N : \tau}$, then by the induction hypothesis we have in $H$
$\sigma_1, \ldots, \sigma_n \vdash \tau \wedge \rho$, so $\sigma_1, \ldots, \sigma_n \vdash \tau$ is obtained by $(\wedge - E)$.

- $(\cap \mathrm{I})$ $\dfrac{\Gamma \vdash N : \tau \quad \Gamma \vdash N : \rho}{\Gamma \vdash N : \tau \cap \rho}$, then by the induction hypothesis we have in $H$ $\sigma_1, \ldots, \sigma_n \vdash \tau \quad \sigma_1, \ldots, \sigma_n \vdash \rho$, so $\sigma_1, \ldots, \sigma_n \vdash \tau \wedge \rho$ is obtained by $(\wedge - I)$.

- $(\cup \mathrm{E})$ $\dfrac{\Gamma, x : \sigma \vdash M : \rho \; \Gamma, x : \tau \vdash M : \rho \; \Gamma \vdash N : \sigma \cup \tau}{\Gamma \vdash M[N/x] : \rho}$, then by the induction hypothesis we have in $H$

$$\sigma_1, \ldots, \sigma_n, \sigma \vdash \rho \quad \sigma_1, \ldots, \sigma_n, \tau \vdash \rho \;\; \sigma_1, \ldots, \sigma_n \vdash \sigma \vee \tau$$

and then by $(\vee - E)$ we obtain $\sigma_1, \ldots, \sigma_n \vdash \rho$.

- $(\cup \mathrm{I})$ $\dfrac{\Gamma \vdash M : \rho}{\Gamma \vdash M : \rho \cup \tau}$, then by the induction hypothesis we obtain
$\sigma_1, \ldots, \sigma_n \vdash \rho (H)$, and then by $(\vee - I)$ $\sigma_1, \ldots, \sigma_n \vdash \rho \vee \tau (H)$.

- $(\omega)$ $\dfrac{}{\Gamma \vdash N : \omega}$, then by $(\top)$ $\sigma_1, \ldots, \sigma_n \vdash \top (H)$.

- $(\leq)$ $\dfrac{\Gamma \vdash N : \tau \quad \tau \leq \rho}{\Gamma \vdash N : \rho}$, then by the induction hypothesis $\sigma_1, \ldots, \sigma_n \vdash \tau \,(H)$. By Proposition 3.2.7 we obtain $\tau \rightarrow \rho \,(H)$. Hence we obtain $\sigma_1, \ldots, \sigma_n \vdash \rho \,(H)$ by $(MP)$. $\square$

**Corollary 3.2.9.** *Let $\sigma \in T$ be a type. If there exists a closed lambda term $M$ such that*

$$\vdash_{\lambda \cap \cup} M : \sigma \quad , \quad \text{then} \quad \sigma \,(H).$$

In Corollary 3.2.9, which is the special case of Proposition 3.2.8 when the term $M$ is closed, we see that the neccessary condition for a type to be inhabited in $\lambda \cap \cup$ is to be provable in $H$. The other way round does not hold, i.e., provability in $H$ is not the sufficient condition for inhabitation in $\lambda \cap \cup$. There are provable formulae which are not inhabited!

The fact that types inhabited in $\lambda\cap$ do not correspond to the provable formulae in intuitionistic propositional logic with $\to$ and $\wedge$, was clearly shown in Hindley, 1984. It is shown that the type

$$\sigma = (\alpha \to \alpha) \cap ((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$$

is not inhabited in $\lambda\cap$ although $\sigma$ is provable in intuitionistic logic. This is based on the fact that if a closed term $M$ is an inhabitant of the type $\alpha \to \alpha$ in the simply typed lambda calculus $\lambda \to$, then $M =_\beta \lambda x.x$ and if $M$ is an inhabitant of the type $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$, then $M =_\beta \lambda xyz.xz(yz)$, which is proved in Ben-Yelles, 1979. If we suppose that $\sigma$ is inhabited by some term $M$ in $\lambda\cap$, then

$$\frac{\vdash M : \sigma}{\vdash M : \alpha \to \alpha \quad \vdash M : ((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))}.$$

By Theorem 1.3.4, the conservative-extension theorem in Barendregt et al., 1983,

$$\vdash_{\lambda\to} M : \alpha \to \alpha \quad \vdash_{\lambda\to} M : ((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)),$$

so $M$ has two different normal forms, which is impossible by Corollary 1.1.9(i).

In order to show that some provable formulae are not inhabited we are going to construct a model of $\lambda\cap$ which is not a model of some provable formula, i.e., its interpretation in this model is going to be empty. Let us consider the following provable formula in $H_\wedge$

$$\alpha \to (\beta \to (\alpha \wedge \beta)).$$

It corresponds to the introduction of conjunction. Its corresponding intersection type is $\alpha \to (\beta \to (\alpha \cap \beta))$. Let $< D, \cdot, \| \, \| >$ be the term model of $\beta$-equality, and let us choose the type interpretation $v$ such that

$$v(\alpha) = \{a\} \, , \, v(\beta) = \{b\}$$

and

$$a \neq b.$$

This condition is not very restrictive since if all the elements of $D$ were equal the model would be trivial.

Now a usual type interpretation given in Definition 3.2.1 can be built using

$$\| \alpha \|_v = \{a\}, \| \beta \|_v = \{b\}.$$

Then

$$\begin{aligned} \| \alpha \to \beta \to (\alpha \cap \beta) \|_v &= \{m | \forall n \in \| \alpha \|_v \; m \cdot n \in \| \beta \to (\alpha \cap \beta) \|_v\} \\ &= \{m | \forall n \in \{a\} \; m \cdot n \in \{p | \forall q \in \| \beta \|_v \; p \cdot q \in \| \alpha \cap \beta \|_v\}\} \\ &= \{m | m \cdot a = \{p | \forall q \in \{b\} \;\; p \cdot q \in \{a\} \cap \{b\}\}\} \\ &= \{m | m \cdot a \cdot b = a, m \cdot a \cdot b = b\}. \end{aligned}$$

If $\| \alpha \to \beta \to (\alpha \cap \beta) \|_v \neq \emptyset$, then

$$a = m \cdot a \cdot b = b.$$

Hence, the interpretation of $\alpha \to \beta \to (\alpha \cap \beta)$ is empty.

## 3.3 Inhabitation in $\mathcal{D}\cup$ and $\lambda_c$

The problem of the decidability of inhabitation in $\lambda\cap$ is open. There are various , mainly proof-thoretic approaches toward this problem in Pottinger, 1980, Lopez-Escobar, 1985, Mints, 1989, Alessi et al., 1991, and Venneri, 1992. Our contribution to these attempts in Theorem 3.3.12 will be the link between the inhabitation in the intersection and union type assignment systems and the inhabitation in the extension of the simply typed lambda calculus with conjunctive and disjunctive types.

The connection between formulae provable in intuitionistic logic and simply typed lambda terms as their constructions is called the Curry–Howard isomorphism or the formulae–as–types interpretation. This idea of connecting inferenes in the typed systems with deductions in logical systems can be found already in Curry et al., 1958. Later in the 1960's it is developed by de Brujin, Howard and Lambek. By this connection the simply typed lambda calculus is the internal language the cartesian closed categories as shown in Lambek et al., 1986. A correspondence between constructive proofs of logical formulae and lambda terms (or combinators) of related types and conversely is established in Howard, (1969)1980.

**Theorem 3.3.1.** (Howard, 1980.) $\varphi(H \rightarrow)$ *iff there exists a closed term* $M$ *such that* $\vdash_{\lambda\rightarrow} M : \varphi$.

In order to obtain a similar result for the extension of $H \rightarrow$ with conjunction, $H_{\bar{\wedge}}$, and disjunction as well, $H$, the following notions are changed in Howard, 1980:

(i) the set of types $T$,

(ii) the language of $\lambda \rightarrow$,

(iii) the type assignment system.

These changes are obtained in the following way:

(i) $T = V|T \rightarrow T|T \wedge T|T \vee T$;

(ii) the set of lambda terms, $\Lambda$, is expanded with new constants $c, c_1$ and $c_2$, for the conjunction and $d, d_1$ and $d_2$ for the disjunction, i.e. $\Lambda_c = V|\{c, c_1, c_2, d, d_1, d_2\}|\Lambda_c\Lambda_c|\lambda V.\Lambda_c$;

(iii) the type assignment system $\lambda_{\bar{\wedge}}$ is obtained from $\lambda \rightarrow$ by adding the rules

$$(\vee E)\quad \frac{\Gamma \vdash M : \varphi \wedge \psi}{\Gamma \vdash c_1 M : \varphi} \qquad \frac{\Gamma \vdash M : \varphi \wedge \psi}{\Gamma \vdash c_2 M : \psi}$$

$$(\vee I)\quad \frac{\Gamma \vdash M : \varphi \quad \Gamma \vdash N : \psi}{\Gamma \vdash cMN : \varphi \wedge \psi};$$

(iv) the type assignment system $\lambda_c$ is obtained from $\lambda_{\bar{\wedge}}$ by adding the rules

$$(\vee E)\quad \frac{\Gamma, x : \varphi \vdash M : \rho \quad \Gamma, x : \psi \vdash N : \rho \quad \Gamma \vdash P : \varphi \vee \psi}{\Gamma \vdash dxMNP : \rho}$$

$$(\vee I)\quad \frac{\Gamma \vdash M : \varphi}{\Gamma \vdash d_1 M : \varphi \vee \psi} \qquad \frac{\Gamma \vdash M : \psi}{\Gamma \vdash d_2 M : \varphi \vee \psi}.$$

For the contraction in $\lambda_c$ it is necessary to add

$$c_1(cMN) \to M \quad , \quad c_2(cMN) \to N$$

(see Howard, 1980). Then a result similar to Theorem 3.3.1 is obtained for $H_{\bar{\wedge}}$ and $H$ and the expanded type assignment system $\lambda_{\bar{\wedge}}$ and $\lambda_c$, respectively.

**Theorem 3.3.2.** (Howard, 1980.)

*(1)* $\varphi(H_{\bar{\wedge}})$ *iff there exists a closed term* $M \in \Lambda_{\bar{\wedge}}$ *such that* $\vdash_{\lambda_{\bar{\wedge}}} M : \varphi$.

*(2)* $\varphi(H)$ *iff there exists a closed term* $M \in \Lambda_c$ *such that* $\vdash_{\lambda_c} M : \varphi$.

The formula $((\alpha \to \beta) \to \alpha) \to \alpha$ is called Peirce's law. It is provable in classical propositional logic, but it is not intuitionisticaly provable. Hence, a corollary to Theorem 3.3.2 is the fact that Peirce's law is not inhabited in $\lambda \to$, but this can be proved directly in $\lambda \to$, as well (see Ghilezan, 1992).

These propositions provide a link between the question of decidability of provability in logics and the question of decidability of inhabitation in the corresponding type assignment systems. For a given formula $\varphi$ it is decidable whether $\varphi(H \to)$ or not, so by Theorem 3.3.1 it is decidable whether there exists $M$, such that

$$\vdash_{\lambda\to} M : \varphi.$$

We proceed similarly with logical extensions $H_{\bar{\wedge}}$ and $H$ and corresponding type system extensions $\lambda_{\bar{\wedge}}$ and $\lambda_c$ using Theorem 3.3.2. Hence, we have the following:

**Theorem 3.3.3.**

*(1) Inhabitation is decidable in* $\lambda_{\bar{x}}$.

*(2) Inhabitation is decidable in* $\lambda_c$.

From the point of view of inhabitation the system $\lambda\cap$, and the system $\mathcal{D}$ are equivalent, as shown in Theorem 2.1.6. Actually the rule $(\leq)$ is not encreasing the set of inhabited types, but it is encreasing the set of inhabitants of a certain already inhabited type. It is known that if the rule $(\leq)$ is replaced by the rule

$$\frac{\Gamma \vdash \lambda x.Mx : \sigma \quad x \notin Fv(M)}{\Gamma \vdash M : \sigma} \ (\eta)$$

we obtain an equivalent system, this system is given in Pottinger, 1980. Anyhow, we saw in Theorem 2.1.6 and Corollary 2.1.7 that a type is inhabited in $\lambda\cap$ and $\lambda_{-\omega}$ by a term $M$ if and only if it is inhabited in $\mathcal{D}\Omega$ and $\mathcal{D}$, respectively, by an $\eta$-conversion of $M$.

It is not so easy to avoid the rule $(\leq)$ in $\lambda\cap\cup$, since it is not equivalent to the rule $(\eta)$; i.e., $(\eta)$-rule is admissible in $\lambda\cap\cup$, but $(\leq)$ is increasing the set of inhabited types. For example the type $\sigma\cap(\tau\cup\rho)\to(\sigma\cap\tau)\cup(\sigma\cap\rho)$ is not inhabited without $(\leq)$.

Further on we are going to consider the question of inhabitation in the type assignment systems $\mathcal{D}$ and $\mathcal{D}\cup$. Let us first define inductively a partial mapping $(\quad)^\cap : \Lambda_c \to \Lambda$ from lambda terms containing constants $c, c_1, c_2, d, d_1$ and $d_2$ into arbitary terms not containing these constants.

**Definition 3.3.4.**

*(i) For each variable* $x \in V$

$$(x)^\cap = x .$$

*(ii)* $(MN)^\cap = \begin{cases} M^\cap N^\cap & \textit{if} \quad M^\cap \neq\uparrow \quad \textit{and} \quad N^\cap \neq\uparrow, \\ \uparrow & \textit{otherwise.} \end{cases}$

*(iii)* $(\lambda x.M)^\cap = \begin{cases} \lambda x.M^\cap & \textit{if} \quad M^\cap \neq\uparrow, \\ \uparrow & \textit{otherwise.} \end{cases}$

*(iv)* $(c_1 M)^\cap = M^\cap$, $(c_2 M)^\cap = M^\cap$.

*(v)* $(cMN)^\cap = \begin{cases} M^\cap & \textit{if} \quad M^\cap = N^\cap \\ \uparrow & \textit{otherwise.} \end{cases}$

*(vi)* $(dxMPN)^\cap = \begin{cases} M^\cap[N^\cap/x] & \textit{if} \quad M^\cap = P^\cap \\ \uparrow & \textit{otherwise.} \end{cases}$

*(vii)* $(d_1 M)^\cap = M^\cap$, $(d_2 M)^\cap = M^\cap$.

If $\varphi \in T$, then $\varphi^\cap$ is obtained from $\varphi$ by replacing all occurrences of $\wedge$ and $\vee$ with $\cap$ and $\cup$, respectively, i.e., we have the following:

**Definition 3.3.5.**

*(i)* $\alpha^\cap = \alpha$.

*(ii)* $(\varphi \wedge \psi)^\cap = \varphi \cap \tau$.

*(iii)* $(\varphi \rightarrow \psi)^\cap = \varphi^\cap \rightarrow \psi^\cap$.

*(iv)* $(\varphi \vee \psi)^\cap = \varphi^\cap \cup \psi^\cap$.

*(v)* *If* $\Gamma$ *is* $x_1 : \sigma_1, \ldots, x_n : \sigma_n$, *then* $\Gamma^\cap$ *is* $x_1 : \sigma_1^\cap, \ldots, x_n : \sigma_n^\cap$.

**Lemma 3.3.6.**
*(1) If* $\Gamma \vdash_{\lambda_{\bar{\wedge}}} N : \varphi$ *and there exists a term* $M \in \Lambda$ *such that* $N^\cap = M$, *then*

$$\Gamma^\cap \vdash_{\mathcal{D}} M : \varphi^\cap.$$

*(2) If* $\Gamma \vdash_{\lambda_c} N : \varphi$ *and there exists a term* $M \in \Lambda$ *such that* $N^\cap = M$, *then*

$$\Gamma^\cap \vdash_{\mathcal{D}\cup} M : \varphi^\cap.$$

**Proof.** (1) By induction on the derivation in $\lambda_{\bar{\wedge}}$ we show that $\Gamma \vdash_{\lambda_{\bar{\wedge}}} N : \varphi$ and $N^\cap \neq \uparrow$ implies $\Gamma^\cap \vdash_{\mathcal{D}} N^\cap : \varphi^\cap$.

(i) If $\dfrac{(x:\sigma)\in\Gamma}{\Gamma\vdash_{\lambda_{\bar{\wedge}}} x:\sigma}$ *(start rule)*
is the last step of the derivation in $\lambda_{\bar{\wedge}}$, then

$$\frac{(x:\ \sigma^{\cap})\in\Gamma^{\cap}}{\Gamma^{\cap}\vdash_{\mathcal{D}} x:\ \sigma^{\cap}}$$

because $x^{\cap}=x$.

(ii) Let $\dfrac{\Gamma\vdash_{\lambda_{\bar{\wedge}}} P:\ \sigma\to\tau\quad \Gamma\vdash_{\lambda_{\bar{\wedge}}} Q:\ \sigma}{\Gamma\vdash_{\lambda_{\bar{\wedge}}} PQ:\ \tau}(\to E)$.
and $(PQ)^{\cap}\neq\uparrow$. Then by Definition 3.3.4. (ii) $(PQ)^{\cap}=P^{\cap}Q^{\cap}$, and $P^{\cap}\neq\uparrow$ and $Q^{\cap}\neq\uparrow$. Then by the induction hypothesis and $(\to E)$

$$\frac{\Gamma^{\cap}\vdash_{\mathcal{D}} P^{\cap}:\ \sigma^{\cap}\to\tau^{\cap}\quad \Gamma^{\cap}\vdash_{\mathcal{D}} Q^{\cap}:\ \sigma^{\cap}}{\Gamma^{\cap}\vdash_{\mathcal{D}} P^{\cap}Q^{\cap}:\ \tau^{\cap}}\ (\to E).$$

Thus $\vdash_{\mathcal{D}} (PQ)^{\cap}:\ \tau^{\cap}$.

(iii) Let

$$\frac{\Gamma,x:\ \sigma\ \vdash_{\lambda_{\to}} P:\ \tau}{\Gamma\vdash_{\lambda_{\bar{\wedge}}} (\lambda x.P):\ \sigma\to\tau}\ (\to I)$$

and $(\lambda x.P)^{\cap}\neq\uparrow$. Then by Definition 3.3.4. (iii) $(\lambda x.P)^{\cap}=\lambda x.P^{\cap}$, and $P^{\cap}\neq\uparrow$. Then by the induction hypothesis and $(\to I)$

$$\frac{\Gamma^{\cap},x:\ \sigma^{\cap}\vdash_{\mathcal{D}} P^{\cap}:\ \tau^{\cap}}{\Gamma^{\cap}\vdash_{\mathcal{D}} (\lambda x.P)^{\cap}:\ \sigma^{\cap}\to\tau^{\cap}}\ (\to I).$$

Thus $\vdash_{\mathcal{D}} (\lambda x.P)^{\cap}:\ (\sigma\to\tau)^{\cap}$.

(iv) Let

$$\frac{\Gamma\vdash_{\lambda_{\bar{\wedge}}} P:\ \sigma\wedge\tau}{\Gamma\vdash_{\lambda_{\bar{\wedge}}} c_1P:\ \sigma}\ (\wedge E)$$

and there is a term $(c_1P)^{\cap}=P^{\cap}\neq\uparrow$. By the induction hypothesis and by $(\cap E)$

$$\frac{\Gamma^{\cap}\vdash_{\mathcal{D}} P^{\cap}:\ \sigma^{\cap}\cap\tau^{\cap}}{\Gamma^{\cap}\vdash_{\mathcal{D}} P^{\cap}:\ \sigma^{\cap}}\ (\cap E).$$

Similarly for the other projection.

(v) Let

$$\frac{\Gamma \vdash_{\lambda_{\bar{\kappa}}} P : \sigma \quad \Gamma \vdash_{\lambda_{\bar{\kappa}}} Q : \tau}{\Gamma \vdash_{\lambda_{\bar{\kappa}}} (cPQ) : \sigma \wedge \tau} \ (\wedge I)$$

and $(cPQ)^{\cap} = P^{\cap} \neq\uparrow$ . So by Definition 3.3.4. (v) $P^{\cap} \neq\uparrow, Q^{\cap} \neq\uparrow$ and $P^{\cap} = Q^{\cap}$. Then by the induction hypothesis and by $(\cap I)$

$$\frac{\Gamma^{\cap} \vdash_{\mathcal{D}} P^{\cap} : \sigma^{\cap} \quad \Gamma^{\cap} \vdash_{\mathcal{D}} P^{\cap} : \tau^{\cap}}{\Gamma^{\cap} \vdash_{\mathcal{D}} P^{\cap} : \sigma^{\cap} \cap \tau^{\cap}} \ (\cap I).$$

Thus $\vdash_{\mathcal{D}} (cPQ)^{\cap} : (\sigma \wedge \tau)^{\cap}$.

(2) The only cases that remain to be proved for $\lambda_c$ are $(\vee E)$ and $(\vee I)$.

(vi) Let

$$\frac{\Gamma, x : \varphi \vdash_{\lambda_c} M : \rho \quad \Gamma, x : \psi \vdash_{\lambda_c} P : \rho \ \Gamma \vdash_{\lambda_c} N : \varphi \vee \psi}{\Gamma \vdash_{\lambda_c} dxMPN : \rho} \quad (\vee E)$$

and $(dxMPN)^{\cap} \neq\uparrow$. Then by Definition 3.3.4. (vi) $M^{\cap} = P^{\cap} \neq\uparrow$ and $(dxMPN)^{\cap} = M^{\cap}[N^{\cap}/x]$. Then by the induction hypothesis

$$\Gamma, x : \varphi^{\cap} \vdash_{\mathcal{D}\cup} M^{\cap} : \rho^{\cap} \quad \Gamma, x : \psi^{\cap} \vdash_{\mathcal{D}\cup} P^{\cap} : \rho^{\cap} \quad \Gamma \vdash_{\mathcal{D}\cup} N^{\cap} : \varphi^{\cap} \cup \psi^{\cap}.$$

and so by $(\cup E)$ we obtain $\Gamma \vdash_{\mathcal{D}\cup} M^{\cap}[N^{\cap}/x] : \rho^{\cap}$.

(vii) Let

$$\frac{\Gamma \vdash_{\lambda_c} M : \sigma}{\Gamma \vdash_{\lambda_c} d_1 M : \sigma \vee \tau} \ (\vee I)$$

and $(d_1 M)^{\cap} \neq\uparrow$. Then by Definition 3.3.4. (vii) $(d_1 M)^{\cap} = M^{\cap}$ and $M^{\cap} \neq\uparrow$. Then by the induction hypothesis

$$\Gamma \vdash_{\mathcal{D}\cup} M^{\cap} : \sigma^{\cap}$$

and $\Gamma \vdash_{\mathcal{D}\cup} M^{\cap} : \sigma^{\cap} \cup \tau^{\cap}$ by $(\cup I)$, where $\sigma^{\cap} \cup \tau^{\cap} = (\sigma \vee \tau)^{\cap}$.

Similarly for the other disjunction introduction rule. □

The following property of the introduced partial mapping is obvious.

**Lemma 3.3.7.** *If $M \in \Lambda$, then $M^{\cap} = M$.*

In order to obtain the converse of Lemma 3.3.6 we define inductively for each derivation $d$ of $\Gamma \vdash M : \varphi$ in $\mathcal{D}$ and $\mathcal{D}\cup$ a term $M^d \in \Lambda_c$.

**Definition 3.3.8.**

*(i) If* $\dfrac{(x:\ \sigma) \in \Gamma}{\Gamma \vdash_{\mathcal{D}} x :\ \sigma}$, *then* $x^d = x$.

*(ii) If*

$$\frac{\Gamma \vdash_{\mathcal{D}} P : \sigma \to \tau \quad \Gamma \vdash_{\mathcal{D}} N :\ \sigma}{\Gamma \vdash_{\mathcal{D}} PN :\ \tau} \quad \textit{and } P^{d_1}, N^{d_2} \in \Lambda_c$$

*correspond to the derivations of the premises, then* $(PN)^d = P^{d_1} N^{d_2}$.

*(iii) If*

$$\frac{\Gamma, x :\ \sigma \quad \vdash_{\mathcal{D}} N :\ \tau}{\Gamma \vdash_{\mathcal{D}} (\lambda x.N) :\ \sigma \to \tau} \quad \textit{and } N^{d_1} \in \Lambda_c$$

*corresponds to the derivation of the premise, then* $(\lambda x.N)^d = \lambda x.N^{d_1}$.

*(iv) If*

$$\frac{\Gamma \vdash_{\mathcal{D}} N :\ \sigma \cap \tau}{\Gamma \vdash_{\mathcal{D}} N :\ \sigma (N :\ \tau)} \quad \textit{and } N^{d_1} \in \Lambda_c$$

*corresponds to the derivation of the premise, then* $N^d = c_1 N^{d_1}$ $(N^d = c_2\ N^{d_1})$.

*(v) If*

$$\frac{\Gamma \vdash_{\mathcal{D}} M :\ \sigma \quad \Gamma \vdash_{\mathcal{D}} M : \tau}{\Gamma \vdash_{\mathcal{D}} M :\ \sigma \cap \tau} \quad \textit{and } M^{d_1}, M^{d_2} \in \Lambda_c$$

*correspond to the derivations of the premises, then* $M^d = cM^{d_1} M^{d_2}$.

*(vi) If*

$$\frac{\Gamma, x : \sigma \vdash_{\mathcal{D}\cup} M : \rho \quad \Gamma, x : \tau \vdash_{\mathcal{D}\cup} M : \rho \quad \Gamma \vdash_{\mathcal{D}\cup} N : \sigma \cup \tau}{\Gamma \vdash_{\mathcal{D}\cup} M[N/x] : \rho} \ (\cup E)$$

*and* $M^{d_1}, M^{d_2}, N^{d_3} \in \Lambda_c$ *correspond to the derivations of premises, then* $(M[N/x])^d = dxM^{d_1} M^{d_2} N^{d_3}$.

*(vii) If*

$$\frac{\Gamma \vdash_{\mathcal{D}\cup} M : \sigma \ \ (M : \tau)}{\Gamma \vdash_{\mathcal{D}\cup} M : \sigma \cup \tau} \ (\cup I), \text{ and } M^{d_1}(M^{d_2}) \in \Lambda_c$$

*corresponds to the derivation of the premise, then* $M^e = d_1 M^{d_1}$ $(M^e = d_2 M^{d_2})$.

**Lemma 3.3.9.**
*(1) If* $d$ *is a derivation of* $\Gamma \vdash_{\mathcal{D}} M : \varphi$*, then* $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^d : \varphi^\wedge$.
*(2) If* $d$ *is a derivation of* $\Gamma \vdash_{\mathcal{D}\cup} M : \varphi$*, then* $\Gamma^\wedge \vdash_{\lambda_c} M^d : \varphi^\wedge$.

**Proof.** (1) By induction on the derivation in $\mathcal{D}$.

(i) If $\dfrac{(x : \sigma) \in \Gamma}{\Gamma \vdash_{\mathcal{D}} x : \sigma}$, then by the start rule of $\lambda_{\bar{\wedge}}$we have$\dfrac{(x : \sigma^\wedge) \in \Gamma^\wedge}{\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} x : \sigma^\wedge}$

(ii) If the last applied rule is $(\rightarrow E)$, i.e., $\dfrac{\Gamma \vdash_{\mathcal{D}} M : \sigma \rightarrow \tau \ \ \Gamma \vdash_{\mathcal{D}} N : \sigma}{\Gamma \vdash_{\mathcal{D}} MN : \tau} \ (\rightarrow E)$,
then by the induction hypothesis $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^{d_1} : (\sigma \rightarrow \tau)^\wedge$ and $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} N^{d_2} : \sigma^\wedge$. By Definition 3.2.6 $(\sigma \rightarrow \tau)^\wedge = \sigma^\wedge \rightarrow \tau^\wedge$, and so by $(\rightarrow E)$
$\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^{d_1} N^{d_2} : \tau^\wedge$. Thus by Definition 3.3.8 $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} (MN)^d : \tau^\wedge$.

(iii) If the last step in the derivation is $(\rightarrow I)$, i.e., $\dfrac{\Gamma, x : \sigma \vdash_{\mathcal{D}} M : \tau}{\Gamma \vdash_{\mathcal{D}} (\lambda x.M) : \sigma \rightarrow \tau} \ (\rightarrow I)$,
then by the induction hypothesis $(\Gamma, x : \sigma)^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^d : \tau^\wedge$. By Definition 3.2.6 $(\Gamma, x : \sigma)^\wedge = \Gamma^\wedge, x : \sigma^\wedge$ and so by $(\rightarrow I)$
$\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} (\lambda x.M^d) : \sigma^\wedge \rightarrow \tau^\wedge$. Thus by Definition 3.3.8 $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} (\lambda x.M)^d : (\sigma \rightarrow \tau)^\wedge$.

(iv) If the last applied rule is $(\cap E)$, i.e., $\dfrac{\Gamma \vdash_{\mathcal{D}} M : \sigma \cap \tau}{\Gamma \vdash_{\mathcal{D}} M : \sigma} \ (\cap E)$, then by the induction hypothesis $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^d : (\sigma \cap \tau)^\wedge$. By Definition 3.2.6
$(\sigma \cap \tau)^\wedge = \sigma^\wedge \wedge \tau^\wedge$, and so by $(\wedge E)$ $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} c_1 M^{d_1} : \sigma^\wedge$. Thus by Definition 3.3.8 $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^d : \sigma^\wedge$.

(v) If the last applied rule is $(\cap I)$, i.e., $\dfrac{\Gamma \vdash_{\mathcal{D}} M : \sigma \ \ \Gamma \vdash_{\mathcal{D}} M : \tau}{\Gamma \vdash_{\mathcal{D}} M : \sigma \cap \tau} \ (\cap I)$,
then by the induction hypothesis $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^{d_1} : \sigma^\wedge$ and $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^{d_2}$ :

$\tau^\wedge$. Hence by $(\wedge I)$ and by Definition 3.2.6 $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} cM^{d_1}M^{d_2} : (\sigma \cap \tau)^\wedge$. Then by Definition 3.3.8 $\Gamma^\wedge \vdash_{\lambda_{\bar{\wedge}}} M^d : (\sigma \cap \tau)^\wedge$.

(2) The only cases that remain to be proved for $\mathcal{D}\cup$ are $(\cup E)$ and $(\cup I)$.

(vi) If the last step is $(\cup E)$, i.e.,

$$\frac{\Gamma, x : \sigma \vdash_{\mathcal{D}\cup} M : \rho \quad \Gamma, x : \tau \vdash_{\mathcal{D}\cup} M : \rho \quad \Gamma \vdash_{\mathcal{D}\cup} N : \sigma \cup \tau}{\Gamma \vdash_{\mathcal{D}\cup} M[N/x] : \rho} \ (\cup E),$$

then by the induction hypothesis $\Gamma^\wedge, x : \sigma^\wedge \vdash_{\lambda_c} M^{d_1} : \rho^\wedge$. $\Gamma^\wedge, x : \tau^\wedge \vdash_{\lambda_c} M^{d_2} : \rho^\wedge$ and $\Gamma^\wedge \vdash_{\lambda_c} N^{d_3} : \sigma^\wedge \vee \tau^\wedge$. By $(\vee E)$ and by Definition 3.2.6 $\Gamma^\wedge \vdash_{\lambda_c} dxM^{d_1}M^{d_2}N^{d_3} : \rho^\wedge$. Thus by Definition 3.3.8 $\Gamma^\wedge \vdash_{\lambda_c} (M[N/x])^d : \rho^\wedge$.

(vii) If the last step is $(\cup I)$, i.e.,

$$\frac{\Gamma \vdash_{\mathcal{D}\cup} M : \sigma \ \ (M : \tau)}{\Gamma \vdash_{\mathcal{D}\cup} M : \sigma \cup \tau} \ (\cup I),$$

then by the induction hypothesis $\Gamma^\wedge \vdash_{\lambda_c} M^{d_1} : \sigma^\wedge$ and then by $(\vee I)$ $\Gamma^\wedge \vdash_{\lambda_c} d_1M^{d_1} : \sigma^\wedge \vee \rho$, where $\rho \equiv \tau^\wedge$ for a suitable union type $\tau$. Then by Definition 3.2.6 $\sigma^\wedge \vee \tau^\wedge = (\sigma \cup \tau)^\wedge$. So, by Definition 3.3.8 $\Gamma^\wedge \vdash_{\lambda_c} M^d : (\sigma \cup \tau)^\wedge$. □

**Lemma 3.3.10.**
*(1) If $d$ is a derivation of $\Gamma \vdash_{\mathcal{D}} M : \varphi$, then $(M^d)^\cap = M$.*

*(2) If $d$ is a derivation of $\Gamma \vdash_{\mathcal{D}\cup} M : \varphi$, then $(M^d)^\cap \doteq M$.*

**Proof.** (1) By induction on the derivation in $\mathcal{D}$.

(i) If we have $\dfrac{(x : \sigma) \in \Gamma}{\Gamma \vdash_{\mathcal{D}} x : \sigma}$, then by Definition 3.3.8. (i) and 3.3.4. (i) $(x^d)^\cap = x^\cap = x$.

(ii) If we have $\dfrac{\Gamma \vdash_{\mathcal{D}} M : \sigma \to \tau \quad \Gamma \vdash_{\mathcal{D}} N : \sigma}{\Gamma \vdash_{\mathcal{D}} MN : \tau} (\to E)$, then by the induction hypothesis $(M^{d_1})^{\cap} = M$ and $(N^{d_2})^{\cap} = N$. Thus

$$\begin{aligned} ((MN)^d)^{\cap} &= (M^{d_1}N^{d_2})^{\cap} \quad \text{by Definition 3.3.8. (ii)} \\ &= (M^{d_1})^{\cap}(N^{d_2})^{\cap} \quad \text{by Definition 3.3.4. (ii)} \\ &= MN \qquad \text{by the induction hypothesis.} \end{aligned}$$

(iii) If we have $\dfrac{\Gamma, x : \sigma \vdash_{\mathcal{D}} M : \tau}{\Gamma \vdash_{\mathcal{D}} (\lambda x.M) : \sigma \to \tau} (\to I)$, then by the induction hypothesis $(M^{d_1})^{\cap} = M$. So

$$\begin{aligned} ((\lambda x.M)^d)^{\cap} &= (\lambda x.M^{d_1})^{\cap} \quad \text{by Definition 3.3.8. (iii)} \\ &= \lambda x.(M^{d_1})^{\cap} \quad \text{by Definition 3.3.4. (iii)} \\ &= \lambda x.M \qquad \text{by the induction hypothesis.} \end{aligned}$$

(iv) If we have $\dfrac{\Gamma \vdash_{\mathcal{D}} M : \sigma \cap \tau}{\Gamma \vdash_{\mathcal{D}} M : \sigma} (\cap E)$, then by the induction hypothesis $(M^{d_1})^{\cap} = M$. Thus

$$\begin{aligned} (M^d)^{\cap} &= (c_1 M^{d_1})^{\cap} \quad \text{by Definition 3.3.8. (iv)} \\ &= (M^{d_1})^{\cap} \quad \text{by Definition 3.3.4. (iv)} \\ &= M \quad \text{by the induction hypothesis.} \end{aligned}$$

Similarly for the other projection.

(v) If $\dfrac{\Gamma \vdash_{\mathcal{D}} M : \sigma \quad \Gamma \vdash_{\mathcal{D}} M : \tau}{\Gamma \vdash_{\mathcal{D}} M : \sigma \cap \tau} (\cap I)$, then by the induction hypothesis $(M^d)^{\cap} = M = (M^{d_2})^{\cap}$. Thus

$$\begin{aligned} (M^d)^{\cap} &= (cM^{d_1}M^{d_2})^{\cap} \quad \text{by Definition 3.3.8. (v)} \\ &= (M^{d_1})^{\cap} \quad \text{by Definition 3.3.4. (v).} \\ &= M \quad \text{by the induction hypothesis.} \end{aligned}$$

(2) Again, $(\cup E)$ and $(\cup I)$ are additional cases for $\mathcal{D}\cup$.

(vi) If we have

$$\frac{\Gamma, x:\sigma \vdash_{\mathcal{D}\cup} M:\rho \quad \Gamma, x:\tau \vdash_{\mathcal{D}\cup} M:\rho \quad \Gamma \vdash_{\mathcal{D}\cup} N:\sigma\cup\tau}{\Gamma \vdash_{\mathcal{D}\cup} M[N/x]:\rho} (\cup E).$$

then by the induction hypothesis $(N^{d_3})^{\cap} = N$, $(M^{d_1})^{\cap} = M$ and $(M^{d_2})^{\cap} = M$, where these $M^{d_1}$ and $M^{d_2}$ are not necessarily equal because they are derivation dependent, as mentioned above. Therefore

$$\begin{aligned}((M[N/x])^d)^{\cap} &= (dxM^{d_1}M^{d_2}N^{d_3})^{\cap} \text{ by Definition 3.3.8. (vi)}\\ &= (M^{d_1})^{\cap}[(N^{d_3})^{\cap}/x] \text{ by Definition 3.3.4. (vi)}\\ &= M[N/x] \text{ by the induction hypothesis.}\end{aligned}$$

(vii) If we have $\dfrac{\Gamma \vdash_{\mathcal{D}\cup} M:\sigma\ (M:\tau)}{\Gamma \vdash_{\mathcal{D}\cup} M:\sigma\cup\tau}$ $(\cup I)$, then by the induction hypothesis $(M^{d_1})^{\cap} = M$ $((M^{d_2})^{\cap} = M)$. Thus

$$\begin{aligned}(M^d)^{\cap} &= (d_1M^{d_1})^{\cap}\ ((d_2M^{d_2})^{\cap}) \text{ by Definition 3.3.8. (vii)}\\ &= (M^{d_1})^{\cap} \text{ by Definition 3.3.4. (vii)}\\ &= M \text{ by the induction hypothesis.} \quad \square\end{aligned}$$

Now, it is possible to prove the converse of Lemma 3.3.6.

**Lemma 3.3.11.** *(1) If $\Gamma \vdash_{\mathcal{D}} M:\varphi$, then there exists a term $N \in \Lambda_c$ such that $N^{\cap} = M$ and $\Gamma^{\wedge} \vdash_{\lambda_{\bar{\wedge}}} N:\varphi^{\wedge}$.*

*(2) If $\Gamma \vdash_{\mathcal{D}\cup} M:\varphi$, then there exists a term $N \in \Lambda_c$ such that $N^{\cap} = M$ and $\Gamma^{\wedge} \vdash_{\lambda_c} N:\varphi^{\wedge}$.*

**Proof.** (1) If $\Gamma \vdash_{\mathcal{D}} M:\varphi$, then by Lemma 3.3.8 $\Gamma^{\wedge} \vdash_{\lambda_{\bar{\wedge}}} M^d:\varphi^{\wedge}$ and by Lemma 3.3.10 $(M^d)^{\cap} = M$.

(2) Similar. $\square$

Together, Lemma 3.3.6 and 3.3.11 state necessary and sufficient conditions for an intersection type to be inhabited in $\mathcal{D}$ and $\mathcal{D}\cup$.

**Theorem 3.3.12.**

*(1)* $\Gamma \vdash_D M : \varphi$ *iff there exists a term* $N \in \Lambda_c$ *such that* $N^\cap = M$ *and* $\Gamma^\wedge \vdash_{\Lambda_{\bar{\wedge}}} N : \varphi^\wedge$.

*(2)* $\Gamma \vdash_{D\cup} M : \varphi$ *iff there exists a term* $N \in \Lambda_c$ *such that* $N^\cap = M$ *and* $\Gamma^\wedge \vdash_{\Lambda_c} N : \varphi^\wedge$.

If we deal with closed terms only, then the following statement is a consequence of Theorem 3.3.12.

**Corollary 3.3.13.** *(1)* $\vdash_D M : \varphi$ *iff there exists a closed term* $N \in \Lambda_c$ *such that* $N^\cap = M$ *and* $\vdash_{\Lambda_{\bar{\wedge}}} N : \varphi^\wedge$.
*(2)* $\vdash_{D\cup} M : \varphi$ *iff there exists a closed term* $N \in \Lambda_c$ *such that* $N^\cap = M$ *and* $\vdash_{\Lambda_c} : \varphi^\wedge$.

# 4 Type assignment systems and untyped lambda calculus

It is known that:

- All untyped lambda terms are typable in the intersection type assignemet system $\lambda\cap$, since every lambda term is trivially typable by $\omega$.

  Solvable terms are the only terms typable in $\lambda\cap$ by nontrivial types.

- Terms having normal form are the only terms typable by types not containing $\omega$ in the same system $\lambda\cap$.

- All strongly normalizing terms are exactly all the terms typable in $\lambda\cap_{-\omega}$ (Corollary 2.3.8).

The four properties mentioned above are given in Theorem 4.2.1. These powerful characterizations of various kinds of lambda terms are a good starting point in proving some properties of the untyped lambda calculus using the properties of the intersection type assignment systems.

In this Chapter we use the characterizations mentioned above in order to prove Finitness of Developments, Continuity Theorem and its consequence Genericity Lemma. A new topology on untyped lambda terms is introduced using typability in $\lambda\cap$.

In Section 4.1 we shall present a proof of the finitness of developments property using the simply typed lambda calculus $\lambda\rightarrow$, suggested by R. Statman and H. Barendregt. In Section 4.2 we shall present two proofs of Genericity Lemma for $\beta\eta$–equality of lambda terms. The first is provided by semantical methods and the second by syntactical methods. We extend the proof for contexts as given in Barendregt, 1984, as well. In Section 4.3 we shall proceed in the application of intersection type assignment systems in the untyped lambda calculus. A new topology on untyped lambda terms is introduced using typability in the intersection type assignment system $\lambda\cap$.

Application appears to be continuous with respect to this new topology, as it is continuous with respect to the Böhm tree topology (see Barendregt, 1984). We shall show that the introduced type topology and the filter topology which is introduced in Barendregt et al., 1983, are the same.

## 4.1 Finitness of developments

In this Section we present a proof of the finitness of developments property (Theorem 4.1.2) using the simply typed lambda calculus $\lambda \rightarrow$, suggested by R. Statman and H. Barendregt.

The theorem of the finitness of developments (see Barendregt, 1984, and de Vrijer, 1987) is proved in Krivine, 1990, by using $\mathcal{D}$ extended with a special constant. Every extended lambda term is typable in $\mathcal{D}$ in every basis, since the new constant provides the possibility for the application of any two terms. The key point of this proof of the finitness of developments property of the untyped lambda calculus is the fact that every term typable in $\mathcal{D}$ is strongly normalizing.

Instead of one constant given in Krivine, 1990, we shall introduce two constants in order to associate a simply typed lambda term to each untyped lambda term we are dealing with. Then, again the same key point can be used, since $\lambda \rightarrow$ is strongly normalizing.

Let us recall the notions we need in order to state the Finitness of Developments. If $M \in \Lambda$ is an untyped lambda term a *redex* of $M$ is defined by Definition 1.1.6. If $M \in \Lambda$ is an untyped lambda term then a *marked lambda term* $M'$ is obtained from $M$ by marking all redexes of $M$. For example

$$\Omega' \equiv (\underline{\lambda x}.xx)(\lambda x.xx),$$

$$\mathsf{K}' \equiv \lambda xy.x \equiv \mathsf{K},$$

$$\mathsf{Y}' \equiv \lambda f.(\underline{\lambda x}.f(xx))(\lambda x.f(xx)).$$

The set of *marked lambda terms* $\Lambda'$ is defined in the following way:

**Definition 4.1.1.**

*(i) The alphabet consists of*

- $V = \{x, y, z, x_1, \ldots\}$ *variables;*
- $\lambda, \lambda_0, \lambda_1, \ldots$ *lambdas;*
- $), ($ *parentheses;*
- $\cdot$ *application.*

*(ii) $\Lambda'$ is the set of words over the given alphabet defined inductively as follows:*

- $x \in V \;\Rightarrow\; x \in \Lambda'$;
- $M \in \Lambda', x \in V \;\Rightarrow\; (\lambda x.M) \in \Lambda'$;
- $M, N \in \Lambda' \;\Rightarrow\; (MN) \in \Lambda'$;
- $M, N \in \Lambda' \;\Rightarrow\; ((\lambda_i x.M)N) \in \Lambda'$, *for all* $i \in N$.

Terms of the form $(\lambda_i x.M).N$ are called *marked redexes. Substitution* in $\Lambda'$ is defined in the usual way. Reduction on marked terms, called $\beta_0$-*reduction*, is defined by

$$(\lambda_i x.M)N \;\rightarrow_{\beta_0}\; M[N/x].$$

**Theorem 4.1.2. (Finitness of developments)** *(Barendregt, 1984, de Vrijer, 1987.)*

*All marked lambda terms are strongly normalizing, i.e.,*

$$SN(\beta_0).$$

We shall prove the finitness of developments via the simply typed lambda calculus. This time we need the Church version of the simply typed lambda calculus (Remark in Section 1.3) with a ground type 0. That is all the types are built up in the usual way from a unique type 0. Let $\Lambda_0$ denote the set of *simply typed lambda terms* with the ground type 0. Since the only typed lambda terms used in this Section are actually the simply typed ones, there is no place for confusion in calling them just typed lambda terms.

Let us fix two constants $f$ and $g$ of type

$$f : 0 \to (0 \to 0)$$

$$g : (0 \to 0) \to 0.$$

These two constants can be thought to act as a retraction pair for a Scott domain. Now it is possible to define a mapping from marked lambda terms into typed lambda terms, i.e.

$$[\ ] : \Lambda' \to \Lambda_0.$$

in the following way:

**Definition 4.1.3.**

*(i)* $[x] = x : 0 \;\; for \;\; each \;\; x \in V.$

*(ii)* $[\lambda x.M] = g(\lambda x : 0.[M]).$

*(iii)* $[MN] = \begin{cases} (\lambda x : 0.[P])[N], & if\ MN\ is\ a\ marked\ redex\ (\lambda_i x.P)N \\ f[M][N], & otherwise. \end{cases}$

An example of the translation will make it more familiar. A marked $\Omega$ is

$$\overline{\Omega} = (\lambda_1 x.xx)(\lambda x.xx).$$

its translation is

$$\begin{aligned} [\overline{\Omega}] &=_{df} (\lambda x : 0.fxx)[\lambda x.xx] \\ &=_{df} (\lambda x : 0.fxx)g(\lambda x : 0.fxx) \\ &=_{\beta} f(g(\lambda x : 0.fxx))(g(\lambda x : 0.fxx). \end{aligned}$$

Let us denote the set of images $[\Lambda'] \subseteq \Lambda_0$ by $\Lambda'_0$. Hence, we can prove some properties of the map [ ] and of the set $\Lambda'_0$.

**Lemma 4.1.4.** *For each $P, Q \in \Lambda'$*

$$[P][[Q]/x] = [P[Q/x]].$$

**Proof.** By induction on the construction of $P \in \Lambda'$.
If $P \equiv x$, then $[x][[Q]/x] = x[[Q]/x] = [Q] = [x[Q/x]]$.
If $P \equiv y \neq x$, then $[y][[Q]/x] = y = [y[Q/x]]$.
If $P \equiv \lambda y.M, x \neq y$, then

$$\begin{aligned}
[\lambda y.M][[Q]/x] &= g(\lambda y:0.[M])[[Q]/x] \text{ by Definition 4.1.3}\\
&= g(\lambda y:0.[M][[Q]/x]) \text{ by substitution}\\
&= g(\lambda y:0.[M[Q/x]] \text{ by the induction hypothesis}\\
&= [(\lambda y.M)[Q/x]] \text{ by Definition 4.1.3 and substitution.}
\end{aligned}$$

If $P \equiv MN$, then

$$\begin{aligned}
[MN][[Q]/x] &= (f[M][N])[[Q]/x] \text{ by Definition 4.1.3}\\
&= f([M][[Q]/x])([N][[Q]/x]) \text{ by substitution}\\
&= f([M[Q/x]])([N[Q/x]]) \text{ by the induction hypothesis}\\
&= [(MN)[Q/x]] \text{ by Definition 4.1.3 and substitution.}
\end{aligned}$$

If $P \equiv (\lambda_i y.M)N, x \neq y$, then

$$\begin{aligned}
[(\lambda_i y.M)N][[Q]/x] &= (\lambda y:0.[M])[N][[Q]/x] \text{ by Definition 4.1.3}\\
&= (\lambda y:0.[M][[Q]/x])([N][[Q]/x]) \text{ by substitution}\\
&= (\lambda y:0.[M[Q/x]])([N[Q/x]]) \text{ by the induction hypothesis}\\
&= [(\lambda_i y.M[Q/x])(N[Q/x])] \text{ by Definition 4.1.3}\\
&= [((\lambda_i y.M)N)[Q/x]] \text{ by substitution. } \square
\end{aligned}$$

It is easy to show by Definition 4.1.3 that all elements of $\Lambda_0'$ are of type 0, that is

$$[M] : 0 \text{ for each } M \in \Lambda'.$$

We shall show that a $\beta$-reduction performed on a term from $\Lambda_0'$ yields also a term from $\Lambda_0'$, i.e., an image of a term from $\Lambda'$.

**Lemma 4.1.5.** *$\Lambda'_0$ is closed under $\beta$-reduction.*

**Proof.** We shall show that if $[M] \rightarrow_\beta N'$, then there is a term $N \in \Lambda'$, such that $M \rightarrow_\beta N$ and $[N] = N'$, by induction on the construction of $M$. Notice that $[M]$ cannot be a variable, since the $\beta$-reduction applicable is trivial in that case.

**Case 1.** $M \equiv \lambda x.P$ and $[M] = g(\lambda x : 0.[P]) \rightarrow_\beta g(\lambda x : 0.Q')$. Then by the induction hypothesis there is a $Q \in \Lambda'$ such that $[Q] = Q'$. thus

$$[M] \rightarrow_\beta g(\lambda x : 0.[Q]) = [\lambda x.Q].$$

**Case 2.** $M \equiv PQ$ and $[M] = f[P][Q] \rightarrow_\beta fS'[Q]$, similarly if we take $[M] \rightarrow_\beta f[P]S'$. Then by the induction hypothesis there is a term $S \in \Lambda'$ such that $[S] = S'$, therefore

$$[M] \rightarrow_\beta f[S][Q] = [SQ].$$

**Case 3.** $M \equiv (\lambda_i x.P)Q$ and $[M] = (\lambda x : 0.[P])[Q] \rightarrow_\beta [P][[Q]/x]$. Take $N \equiv P[Q/x]$ by Lemma 4.1.4 $[M] \rightarrow_\beta [P[Q/x]]$. Thus one has $[N] \equiv [P[Q/x]]$. □

**Lemma 4.1.6.** *Let $M \in \Lambda'$. Then*

$$M \rightarrow_{\beta_0} N \text{ if and only if } [M] \rightarrow_\beta [N].$$

**Proof.** ($\Rightarrow$) By induction on the formation of $M$.

**Case 1.** If $M \equiv \lambda x.P$, then the $\beta_0$-reduction is on $P$, so $\lambda x.P \rightarrow_{\beta_0} \lambda x.Q$ because $P \rightarrow_{\beta_0} Q$. Then

$$\begin{aligned}[\lambda x.P] = g(\lambda x : 0.[P]) &\rightarrow_\beta g(\lambda x : 0.[Q]) \text{ by the induction hypothesis}\\ &= [\lambda x.Q] \text{ by Definition 4.1.3.}\end{aligned}$$

**Case 2.** If $M \equiv PQ$ and $PQ \rightarrow_{\beta_0} SQ$ holds because $P \rightarrow_{\beta_0} S$, then

$$\begin{aligned}[PQ] = f[P][Q] &\rightarrow_\beta f[S][Q] \text{ by the induction hypothesis}\\ &= [SQ] \text{ by Definition 4.1.3.}\end{aligned}$$

**Case 3.** If $M \equiv (\lambda_i x.P)Q \rightarrow_{\beta_0} P[Q/x]$, then

$$\begin{array}{rcl} [(\lambda_i x.P)Q] = (\lambda x:0.[P])[Q] & \rightarrow_\beta & [P][[Q]/x] \\ & = & [P[Q/x]] \text{ by Lemma 4.1.4.} \end{array}$$

($\Leftarrow$) Again, by induction on the formation of $M$.

**Case 1.** If $M = \lambda x.P$, then

$$\begin{array}{rcl} [\lambda x.P] = g(\lambda x:0.[P]) & \rightarrow_\beta & g(\lambda x:0.Q') \text{ and } Q' \in \Lambda'_0 \text{ by Lemma 4.1.5} \\ & = & g(\lambda x:0.[Q]) \text{ for some } Q \in \Lambda' \\ & = & [\lambda x.Q]. \end{array}$$

and we know that $[P] \rightarrow_\beta Q' = [Q]$. By the induction hypothesis for $[P] \rightarrow_\beta [Q]$ follows $P \rightarrow_{\beta_0} Q$, thus

$$\lambda x.P \rightarrow_{\beta_0} \lambda x.Q.$$

**Case 2.** $M = PQ$ similarly.

**Case 3.** $M = (\lambda_i x.P)Q$

$$\begin{array}{rcl} [(\lambda_i x.P)Q] = (\lambda x:0.[P])[Q] & \rightarrow_\beta & [P][[Q]/x] \\ & = & [P[Q/x]] \end{array}$$

Then $(\lambda_i x.P)Q \rightarrow_{\beta_0} P[Q/x]$. □

**Corollary 4.1.7.** *$N$ is a $\beta_0$-normal form if and only if $[N]$ is a $fg\beta$-normal form.*

It is easy to prove now that $SN(\beta_0)$, using the property that the simply typed lambda terms are strongly normalizing.

**Proof of Theorem 4.1.2.** Let $M \in \Lambda'$, then$[M] \in \Lambda'_0$. Since $\Lambda'_0 \subseteq \Lambda_0$ and $\Lambda_0$ is strongly normalizing, each $\beta$-reduction path of $[M]$ is finite. Thus all reductions of $[M]$ are of the form

$$[M] \equiv N_0 \rightarrow_\beta N_1 \rightarrow_\beta \ldots \rightarrow_\beta N_k$$

for some $k \in N$ and where $N_k$ is a $\beta$-normal form. By Lemma 4.1.5 each $N_i \in \Lambda'_0$ for $0 \leq i \leq k$. Thus there are $M_i \in \Lambda'$ such that $|M_i| = N_i$ for all $0 \leq i \leq k$. Then by Lemma 4.1.6 for each $\beta$-reduction path in $\Lambda'_0$ mentioned there is a finite $\beta_0$-reduction path of $M$ in $\Lambda'$

$$M \equiv M_0 \rightarrow_{\beta_0} M_1 \rightarrow_{\beta_0} \dots \rightarrow_{\beta_0} M_k.$$

Again by Lemma 4.1.6 these are the only possible $\beta$-reductions of $M$, so all of them are finite. $\square$

Now, we shall reconstruct the proof of the finitness of developments ($FD$) via the intersection type assignment system $\mathcal{D}$ given in Krivine, 1990, in order to be able to compare it with the given proof in the simply typed lambda calculus. Let us extend the set of untyped lambda terms $\Lambda$ by a *new constant* $f$. The extended set $\Lambda(f)$ is defined as the smallest set satisfying the following:

**Definition 4.1.8.**

*(i)* $x \in V \ \Rightarrow\ x \in \Lambda(f)$;

*(ii)* $M \in \Lambda(f), x \in V \ \Rightarrow\ \lambda x.M \in \Lambda(f)$;

*(iii)* $M, N \in \Lambda(f) \ \Rightarrow\ fMN \in \Lambda(f)$;

*(iv)* $M, N \in \Lambda(f) \ \Rightarrow\ (\lambda x.M)N \in \Lambda(f)$.

Then the following properties of $\Lambda(f)$ can be proved.

**Lemma 4.1.9.** *If* $M, N \in \Lambda(f)$, *then* $M[N/x] \in \Lambda(f)$.

**Proof.** By induction on the construction of $M \in \Lambda(f)$. $\square$

**Lemma 4.1.10.** $\Lambda(f)$ *is closed under* $\beta$*-reduction.*

**Proof.** If $M \in \Lambda(f)$ and $M \rightarrow_\beta M'$, then $M' \in \Lambda(f)$. This can be shown by induction on the generation of $\rightarrow_\beta$ . $\square$

**Lemma 4.1.11.**(Krivine, 1990.)
*Let $M \in \Lambda(f)$ and let $\Gamma$ be a basis such that all free variables of $M$ are in $\Gamma$, and $f$ is not in $\Gamma$. Then there exist intersection types $\varphi, \psi \in T$ such that*

$$\Gamma, f : \varphi \vdash_{\mathcal{D}} M : \psi.$$

**Proof.** By induction on the construction of $M$.

**Case 1.** $M \equiv x$, obvious.

**Case 2.** Let $M \equiv \lambda x.P$ and let $\Gamma$ be a basis with all the free variables of $M$ and suppose $x \notin \Gamma$, this is not a constraint since if $x \in \Gamma$, then we can rename the bound variable of M, i.e., $M = \lambda y.P[y/x]$ for some $y \notin \Gamma$. Then $\Gamma$ is a basis with all the free variables of $P$ as well, except $x$. By the induction hypothesis there are $\varphi, \rho \in T$ such that

$$\Gamma, x : \sigma, f : \varphi \vdash_{\mathcal{D}} P : \rho.$$

Thus by $(\to I)$

$$\Gamma, f : \varphi \vdash_{\mathcal{D}} (\lambda x.P) : \sigma \to \rho.$$

**Case 3.** Let $M \equiv fPQ$, for some $P, Q \in \Lambda(f)$. By the induction hypothesis

$$\Gamma, f : \varphi' \vdash_{\mathcal{D}} P : \rho \text{ and } \Gamma, f : \varphi'' \vdash_{\mathcal{D}} Q : \sigma.$$

Let $\alpha$ be a fresh type variable. Then

$$\Gamma, f : \varphi' \cap \varphi'' \cap (\rho \to (\sigma \to \alpha)) \vdash_{\mathcal{D}} fPQ : \alpha.$$

**Case 4.** Let $M = (\lambda x.P)Q$ for some $P, Q \in \Lambda(f)$ and let $\Gamma$ be a basis containing all free variables of $M$ and not containing $x$, then by the induction hypothesis there exist $\varphi'', \sigma \in T$ such that

$$\Gamma, f : \varphi'' \vdash_{\mathcal{D}} Q : \sigma.$$

Again, by the induction hypothesis there exist $\varphi', \rho \in T$ such that

$$\Gamma, x : \sigma, f : \varphi' \vdash_{\mathcal{D}} P : \rho.$$

Then

$$\Gamma, f : \varphi' \vdash_{\mathcal{D}} (\lambda x.P) : \sigma \to \rho.$$

so

$$\Gamma, f : \varphi' \cap \varphi'' \vdash_{\mathcal{D}} (\lambda x.P) : \sigma \to \rho \text{ and } \Gamma, f : \varphi' \cap \varphi'' \vdash_{\mathcal{D}} Q : \sigma.$$

Thus

$$\Gamma, f : \varphi' \cap \varphi'' \vdash_{\mathcal{D}} (\lambda x.P)Q : \rho. \quad \Box$$

This interesting and surprising property that a term from $\Lambda(f)$ is typable in every basis containing its free variables if a correct type for $f$ is chosen, is due to the $(\cap E)$ rule. We saw in Chapter 3 that the rules of intersection introduction and intersection elimination are on the one hand the cause of trouble in trying to find the logical meaning of intersection as a connective, while on the other hand the same rules help us to handle eassily the type system.

**Theorem 4.1.12.** *All terms of $\Lambda(f)$ are strongly normalizing.*

**Proof.** Immediately from the Strong normalization theorem (Theorem 2.2.7) for $\mathcal{D}$. $\Box$

Again, it is possible to define a mapping [ ] from the set of marked lambda terms into $\Lambda(f)$, i.e.,

$$[\,] : \Lambda' \to \Lambda(f)$$

in the following way:

(i) $[x] = x$ for each $x \in V$,

(ii) $[\lambda x.M] = \lambda x.[M]$,

(iii)

$$[MN] = \begin{cases} (\lambda x.[P])[N], & \text{if } MN \text{ is a marked redex}(\lambda_i x.P)N \\ f[M][N], & \text{otherwise.} \end{cases}$$

This translation of the marked $\Omega$ is obtained in the following way:

$$\begin{aligned}[\overline{\Omega}] &= [(\lambda_i x.xx)(\lambda x.xx)] \\ &= (\lambda x.[xx])[\lambda x.xx] \\ &= (\lambda x.fxx)(\lambda x.fxx) \\ &=_\beta f(\lambda x.fxx)(\lambda x.fxx).\end{aligned}$$

**Lemma 4.1.13.** *Let $M \in \Lambda'$. Then*

$$M \twoheadrightarrow_{\beta_0} N \textit{ if and only if } [M] \twoheadrightarrow_\beta [N].$$

**Proof.** Similar to the proof of Lemma 4.1.6. $\square$

Hence in this case, the property of the finitness of the developments, i.e. $SN(\beta_0)$, is a consequence of Lemma 4.1.13 and of the Strong Normalization Theorem for $\mathcal{D}$. The idea used in both of the proofs is essentialy the same and it is based on the strong normalization property of both $\lambda \rightarrow$ and $\mathcal{D}$ (Proposition 2.2.7). The realization differs in the fact that one needs on the one hand two constants in $\lambda \rightarrow$, while on the other hand one constant is sufficinet in $\mathcal{D}$. This is due to the difference in the type systems.

## 4.2 Genericity lemma

In this Section we proceed in the "typed" approach toward the untyped lambda calculus dealing with Genericity Lemma for $\beta\eta$–equality of lambda terms and more general for contexts (for the direct proof see Barendregt, 1984, and Wadsworth, 1971). We present two proofs of Genericity Lemma (Proposition 4.2.7) for lambda terms. The first is provided by semantical methods and the second by syntactical methods. We extend the proof for contexts, as given in Barendregt, 1984, in Proposition 4.2.12.

A classification of lambda terms according to their reduction properties, e.g. normal forms, strongly normalizing, normalizing, solvable and unsolvable terms, is given in Section 1.1. The characterization of lambda terms by their typability in the systems $\lambda\cap$ and $\mathcal{D}$ is given in the following statement:

**Theorem 4.2.1.**

*(i) (Barendregt et al., 1983.)*
*M has a normal form* $\Leftrightarrow \exists\Gamma,\sigma\ (\omega \notin \Gamma,\sigma)\ \Gamma \vdash M : \sigma$.

*(ii) (Barendregt et al., 1983.)*

$$M \text{ is solvable } \Leftrightarrow \exists\Gamma \not\ni \omega, \sigma \not\sim \omega\ \Gamma \vdash M : \sigma.$$

*(iii)* $M$ *is unsolvable* $\Leftrightarrow \forall\Gamma,\sigma(\Gamma \vdash M : \sigma \Rightarrow \sigma \sim \omega)$.

*(iv) (van Bakel, 1992, Krivine, 1990.)*

$$M \text{ is strongly normalizing } \Leftrightarrow \exists\Gamma,\sigma\quad \Gamma \vdash_{\mathcal{D}} M : \sigma.$$

The notation $\omega \in \Gamma, \sigma$ means that the type $\omega$ occurs in the type $\sigma$ and in some of the types, which are the predicates of the basis $\Gamma$. The relation $\sim$ is given in Definition 1.3.2.

The notion of *principal type* and *principal typing* of a term in normal form is introduced in Ronchi et al., 1984, and can be found in Krivine, 1990, for the intersection type assignment system without the pre-order $\leq$ on types. Actually, the rule $(\leq)$ is not important for these notions and hence they are the same in both systems $\mathcal{D}$ and $\lambda\cap_{-\omega}$.

Let $N$ be a normal form and let $Fv(N) \subseteq \{x_1, \ldots, x_n\}$. A *principal typing* of $N$ in the system $\lambda\cap_{-\omega}$ of the form $x_1 : \varphi_1, \ldots, x_n : \varphi_n \vdash_{-\omega} N : \pi$ is defined inductively on the normal form construction as follows:

**Definition 4.2.2.**

*(i)* $N \equiv x_i, N$ *is a term variable.*
*If* $\alpha_1, \ldots, \alpha_k$ *are distinct type variables, then*
$x_1 : \alpha_1, \ldots, x_k : \alpha_k \vdash_{-\omega} x_i : \alpha_i$ *is a principal typing of* $N$.

*(ii)* $N \equiv \lambda x.P$ *and* $P$ *is a normal form.*
*If* $x : \varphi, x_1 : \varphi_1, \ldots, x_k : \varphi_k \vdash_{-\omega} P : \psi$ *is a principal typing of* $P$,
*then* $x_1 : \varphi_1, \ldots, x_k : \varphi_k \vdash_{-\omega} \lambda x.P : \varphi \to \psi$ *is a principal typing of* $N$.

(iii) *$N \equiv xN_1 \ldots N_m$ and $N_1, \ldots, N_m$ are normal forms.*
*Let $x : \varphi_i, x_1 : \varphi_1^i, \ldots, x_k : \varphi_k^i \vdash_{-\omega} N_i : \psi_i$ be a principal typing of $N_i$, $0 \leq i \leq m$, such that for all $i \neq j$ the typings of $N_i$ and $N_j$ have no joint type variables. If $\alpha$ is a new type variable, fresh for all typings, then*

$$x : \bigcap_{i=1}^{m} \varphi_i \cap (\psi_1 \to \ldots \to \psi_m \to \alpha), x_1 : \bigcap_{i=1}^{m} \varphi_1^i, \ldots, x_k : \bigcap_{i=1}^{m} \varphi_k^i \vdash_{-\omega} xN_1 \ldots N_m : \alpha$$

*is a principal typing of $N$.*

It is easy to prove that normal forms are preserved under one-step $\eta$-reduction.

**Lemma 4.2.3.** *If $N$ is a $\beta$-normal form and $N \to_\eta N'$, then $N'$ is a $\beta$-normal form.*

Proof. By induction on the construction of $N$.

(i) If $N$ is a term variable, then, obviously, $N \equiv N'$.

(ii) If $N$ is an abstraction, then there are two possibilities:

- $N \equiv \lambda x.M$ and $M \to_\eta M'$, thus $N' \equiv \lambda x.M'$. $M$ is a normal form, so by the induction hypothesis $M'$ is a normal form, and so is $N'$.
- $N \equiv \lambda x.N'x$ with $x \notin Fv(N')$ and $N \to_\eta N'$. Since $N$ is a normal form $N'$ is a normal form as well.

(iii) If $N$ is not an abstraction, i.e., $N = xN_1 \ldots N_k$, where $N_i$ $(1 \leq i \leq k)$ are normal forms, then $N'$ is $xN_1 \ldots N_i' \ldots N_k$ for some $i$ with $1 \leq i \leq k$ where $N_i \to_\eta N_i'$. By the induction hypothesis all $N_i'(1 \leq i \leq k)$ are normal forms, thus $N'$ is a normal form. □

This one-step $\eta$-reduction can be extended to an $\eta$-reduction, and so as a corollary to Lemma 4.2.3, we obtain that normal forms are preserved under $\eta$-reduction.

In order to give the proof of Genericity Lemma we need some preparatory work. We call this proof semantical because type interpretations and term valuations are involved.

Since we are dealing with all lambda terms sometimes it will be necessary to point out for a normalizing term the reduction path which is reducing it to its normal form. The left-most $\beta$-reduction is always reducing a normalizing term to its normal form, so by the notation $M \twoheadrightarrow_\ell N$ we shall mean that the $\beta$-reduction from $M$ to $N$ is not just any one, but it is exactly the left-most $\beta$-reduction. The following commuting property of $\eta$- and left-most $\beta$-reduction will be used later.

**Lemma 4.2.4.** *Let $M$ and $N$ be lambda terms and let $x \notin Fv(M)$. If $\lambda x.Mx \twoheadrightarrow_\ell \lambda x.N$, then there is an $\eta$-reduct $Q$ of $\lambda x.N$, such that $M \twoheadrightarrow_\ell Q$. i.e.,*

$$\begin{tikzcd}
\lambda x.Mx \arrow[r, two heads, "\ell"'] \arrow[d, "\eta"] & \lambda x.N \arrow[d, "\eta"] \\
M \arrow[r, two heads, "\ell"'] & Q
\end{tikzcd}$$

**Proof.** By induction on the number of left-most $\beta$-reductions in the reduction $\lambda x.Mx \twoheadrightarrow_\ell \lambda x.N$.

**Case 1.** If $\lambda x.Mx \equiv \lambda x.N$, then $\lambda x.N \equiv \lambda x.Mx \to_\eta M$, so $Q \equiv M$.

**Case 2.** If $M \equiv \lambda y.P$, then

$$\lambda x.Mx \equiv \lambda x.(\lambda y.P)x \twoheadrightarrow_\ell \lambda x.P[x/y] \twoheadrightarrow_\ell \lambda x.N.$$

Since $\quad \lambda y.P = \lambda x.P[x/y] \quad$ we have that $\quad M \twoheadrightarrow_\ell \lambda x.N.$

**Case 3.** If $M$ is not an abstraction, then the first left-most $\beta$-reduction is somewhere inside $M$, say $M \twoheadrightarrow_\ell M'$, so

$$\lambda x.Mx \twoheadrightarrow_\ell \lambda x.M'x \twoheadrightarrow_\ell \lambda x.N.$$

By the induction hypothesis there is an $\eta$-reduct $Q$ of $\lambda x.N$ such that $M' \twoheadrightarrow_\ell Q$. Therefore $\quad M \twoheadrightarrow_\ell M' \to_\ell Q.$ □

In order to prove the relation between a normal form and a term typable by its principal type various type interpretations $\| \|_{\upsilon} : T \to \mathcal{P}(T)$ (Definition 3.2.1). are used in Krivine, 1990.

**Proposition 4.2.5.** *Let $N$ be a normal form, and let $\Gamma \vdash_{-\omega} N : \pi$ be a principal typing of $N$. There is a type interpretation $\upsilon$ such that:*

*(i)* $\upsilon \models \Gamma$.

*(ii) if $M \in \Lambda$ is such that $Fv(M) \subseteq \Gamma$ and $M \in \|\pi\|_{\upsilon}$, then there is an $\eta$-reduct $Q$ of $N$ such that $M \twoheadrightarrow_{\ell} Q$.*

**Proof.** By induction on the formation of a principal typing of a normal form $N$. □

**Proposition 4.2.6.**

*(i) Let $N$ be a normal form, and let $\Gamma \vdash_{-\omega} N : \pi$ be a principal typing of $N$. Then, if $\Gamma \vdash M : \pi$, then there is an $\eta$-reduct $Q$ of $N$ such that $M \twoheadrightarrow_{\ell} Q$.*

*(ii) Let $N$ be a $\beta\eta$-normal form, and let $\Gamma \vdash_{-\omega} N : \pi$ be a principal typing of $N$. Then, if $\Gamma \vdash M : \pi$, then $M \twoheadrightarrow_{\ell} N$.*

**Proof.** (i)

$$\begin{aligned} \Gamma \vdash M : \pi \;&\Rightarrow\; \Gamma \models M : \pi \\ &\Leftrightarrow\; \forall \rho, \upsilon\ (\rho, \upsilon \models \Gamma \Rightarrow \|M\|_{\rho} \in \|\pi\|_{\upsilon}). \end{aligned}$$

On the other hand by Proposition 4.2.5(i) for the term valuation $\rho^{*}(x) = x$ there is a type interpretation $\upsilon^{*}$ such that $\rho^{*}, \upsilon^{*} \models \Gamma$. It follows that

$$\|M\|_{\rho^{*}} \in \|\pi\|_{\upsilon^{*}}.$$

Thus

$$M \in \|\pi\|_{\upsilon^{*}}$$

and hence

$$\exists Q\ (N \twoheadrightarrow_{\eta} Q \text{ and } M \twoheadrightarrow_{\ell} Q)$$

by Proposition 4.2.5(ii).

(ii) Straightforward by (i). □

And now follows the proof of the Genericity Lemma for lambda terms and for $\beta\eta$-equality.

**Proposition 4.2.7. (Genericity Lemma)**
*Let $M$ and $N$ be lambda terms such that $M$ is unsolvable and $N$ has a normal form. Then, for all lambda terms $F$,*

$$FM = N \Rightarrow (\forall L \in \Lambda)\ FL =_{\beta\eta} N.$$

**Proof 1.** Let $N_{nf}$ be the normal form of $N, N \twoheadrightarrow_{\ell} N_{nf}$. Then there is a principal typing of $N_{nf}$, say, $\Gamma \vdash_{-\omega} N_{nf} : \pi$ ($\Gamma$ and $\pi$ do not contain $\omega$). Since $\lambda\cap$ is closed under $\beta$-conversion, we have

$$\Gamma \vdash FM : \pi.$$

Then there is a type $\tau$ such that

$$\Gamma \vdash F : \tau \rightarrow \pi \text{ and } \Gamma \vdash M : \tau,$$

by the structural property of $\lambda\cap$ which is given in Proposition 1.3.5.(i). Since $M$ is unsolvable $\tau \sim \omega$ by Theorem 4.2.1 (iii), and therefore by the application of the rule ($\leq$) in $\lambda\cap$ we obtain

$$\Gamma \vdash F : \omega \rightarrow \pi. \qquad (1)$$

On the other hand, if we take any lambda term $L$ by the rule ($\omega$)

$$\Gamma \vdash L : \omega, \qquad (2)$$

and so we get

$$\Gamma \vdash FL : \pi$$

by applying $(\to E)$ on (1) and (2). Now, by Proposition 4.2.6 there is an $\eta$-reduct $Q$ of $N$ such that $FL \twoheadrightarrow_{\varepsilon} Q$, and so $FL =_{\beta\eta} N$. $\Box$

In order to link the Genericity Lemma for lambda terms and the Genericity Lemma for contexts, as given in Barendregt, 1984, let us recall the notion of context. A context $C[\ ]$ is a lambda term with some holes in it. The notion of a *context*, $C[\ ]$, is introduced in Barendregt, 1984, in the following way:

**Definition 4.2.8.**

*(i)* *$x$ is a context.*

*(ii)* *$[\ ]$ is a context.*

*(iii)* *If $C_1[\ ]$ and $C_2[\ ]$ are contexts, then $C_1[\ ]C_2[\ ]$ is a context.*

*(iv)* *If $C[\ ]$ is a context, then $\lambda x.C[\ ]$ is a context.*

*(v)* *If $C[\ ]$ is a context and $M \in \Lambda$, then $C[M]$ is a lambda term obtained by placing $M$ in the holes of $C[\ ]$. The free variables of $M$ are allowed to become bound in $C[M]$.*

In the substitution of lambda terms, $N[M/x]$, the free variables of $M$ have to remain free in $N[M/x]$, while the replacement $C[M]$ can have the binding effect. For example in the context $C[\ ] \equiv \lambda x.[\ ]$ we can place the term $M \equiv xx$ and obtain the term $\lambda x.xx$, in which the free variables of $M$ become bound. This cannot happen with the substitution of lambda terms, i.e., the term $\lambda x.xx$ cannot be obtained by any substitution.

The set of free variables of a context, $Fv(C)$ is defined similarly as the set of free variables of a lambda term (Definition 1.1.3) with the addition for the hole $Fv([\ ]) = \emptyset$. The connection between contexts and lambda terms is given in the following lemma. Let us recall that $\Lambda^0(\vec{y}) = \{M \in \Lambda | Fv(M) = \{\vec{y}\}\}$, where $\vec{y}$ denotes a sequence of variables.

**Lemma 4.2.9.** *Let $C[\ ]$ be a context. There is a set of variables $\{\vec{x}\}$ such that for all $\{\vec{y}\} \supseteq \{\vec{x}\} \cup Fv(C)$ there exists a term $F \in \Lambda^0(\vec{y}\backslash\vec{x})$ such that*

$$(\forall M \in \Lambda)\ C[M] = F(\lambda\vec{y}.M).$$

**Proof.**

(i) If $C[\ ] \equiv x$, then $\{\vec{x}\} = \{x\}$. For every $\{\vec{y}\} \supseteq \{x\}$ one has that $F \equiv \lambda z.(\lambda\vec{y}.x)\vec{y} \in \Lambda^0(\vec{y})$ since $C[M] \equiv x =_\beta (\lambda z.(\lambda\vec{y}.x)\vec{y})(\lambda\vec{y}.M)$.

(ii) If $C[\ ] \equiv [\ ]$, then $\{\vec{x}\} = \emptyset$. For every $\{\vec{y}\}$ we have that $F \equiv \lambda x.x\vec{y} \in \Lambda^0(\vec{y})$ since $C[M] = M =_\beta (\lambda x.x\vec{y})(\lambda\vec{y}.M)$.

(iii) If $C[\ ] = C_1[\ ]C_2[\ ]$, then by the induction hypothesis there are sets $\{\vec{x}_1\}$ and $\{\vec{x}_2\}$ corresponding to contexts $C_1[\ ]$ and $C_2[\ ]$, respectively. Take $\{\vec{x}\} = \{\vec{x}_1\} \cup \{\vec{x}_2\}$. Hence, by the induction hypothesis for every $\{\vec{y}\} \supseteq \{\vec{x}_1\} \cup \{\vec{x}_2\} \cup (Fv(C_1) \cap Fv(C_2))$ there are $F_i \in \Lambda^0(\vec{y}\backslash\vec{x}_i)$ $i = 1, 2$, such that

$$C[M] \equiv C_1[M]C_2[M] = F_1(\lambda\vec{y}.M)(F_2(\lambda\vec{y}.M)) =_\beta (\lambda z.F_1 z(F_2 z))(\lambda\vec{y}.M).$$

Thus $F \equiv \lambda z.F_1 z(F_2 z) \in \Lambda^0(\vec{y}\backslash\vec{x}_1) \cap \Lambda^0(\vec{y}\backslash\vec{x}_2) = \Lambda^0(\vec{y}\backslash(\vec{x}_1 \cup \vec{x}_2))$.

(iv) If $C[\ ] \equiv \lambda z.C_1[\ ]$, then by the induction hypothesis there exists a set $\{\vec{x}_1\}$ which corresponds to the context $C_1[\ ]$. Now, there are two possibilities:

$$z \in \{\vec{x}_1\} \text{ or } z \notin \{\vec{x}_1\}.$$

**Subcase 1.** If $z \in \{\vec{x}_1\}$, then by the induction hypothesis for every set $\{\vec{y}\} \supseteq \{\vec{x}_1\} \cup Fv(C_1)$ there is a term $F_1 \in \Lambda^0(\vec{y}\backslash\vec{x}_1)$ such that

$$C_1[M] = F_1(\lambda\vec{y}.M).$$

Let us take $\{\vec{x}\} = \{\vec{x}_1\}$. Thus

$$C[M] \equiv \lambda z.C_1[M] = \lambda z.F_1(\lambda\vec{y}.M) =_\beta (\lambda pz.F_1 p)(\lambda\vec{y}.M)\ ,\ p \notin Fv(F_1).$$

Hence $F \equiv \lambda pz.F_1 p \in \Lambda^0(\vec{y}\backslash\vec{x})$.

**Subcase 2.** If $z \notin \{\vec{x}_1\}$, then let us take $\{\vec{x}\} = \{\vec{x}_1\} \cup \{z\}$. Let us consider $\{\vec{y}\} \supseteq \{\vec{x}\} \cup Fv(C_1) \supset \{\vec{x}_1\} \cup Fv(C_1)$. On the one hand $\{\vec{y}\}\backslash\{\vec{x}\} = \{\vec{y}_1\}\backslash\{\vec{x}_1\}$, where $\{\vec{y}_1\} = \{\vec{y}\}\backslash\{z\}$ and $\{\vec{y}_1\} \supseteq \{\vec{x}_1\}$. On the other hand by the induction hypothesis there is a term $F_1 \in \Lambda^0(\vec{y}_1\backslash\vec{x}_1)$ such that

$$C_1[M] = F_1(\lambda\vec{y}_1.M).$$

Therefore

$$\begin{aligned} C[M] &\equiv \lambda z.C_1[M] = \lambda z.F_1(\lambda\vec{y}_1.M) = \lambda z.F_1((\lambda z\vec{y}_1.M)z) \\ &= (\lambda pz.F_1(pz))(\lambda zy_1.M), p \notin Fv(F_1) \end{aligned}$$

and hence $F \equiv \lambda pz.F_1(pz) \in \Lambda^0(\vec{y}\backslash\vec{x})$. $\square$

Since Lemma 4.2.9 holds for every set $\{\vec{y}\} \supseteq \{\vec{x}\} \cup Fv(C)$, it holds for $\{\vec{y}\} = \{\vec{x}\} \cup Fv(C)$. Thus a corollary to Lemma 4.2.9 is the following statement:

**Corollary 4.2.10.** *For all contexts $C[\ \ ]$ there is a set of variables $\{\vec{x}\}$ and a term $F \in \Lambda^0(Fv(C))$ such that for all $M \in \Lambda$*

$$C[M] = F(\lambda\vec{y}.M).$$

*where*

$$\{\vec{y}\} = \{\vec{x}\} \cup Fv(C).$$

Now, we shall see that unsolvable terms remain unsolvable after any closure.

**Lemma 4.2.11.** *$M \in \Lambda$ is unsolvable if and only if $\lambda\vec{x}.M$ is unsolvable for any set of variables $\{\vec{x}\}$.*

**Proof.** ($\Leftarrow$) Let $\lambda\vec{x}.M$ be unsolvable. Suppose $M$ is solvable, thus

$$M =_\beta \lambda\vec{y}.z\vec{N}.$$

But then

$$\lambda\vec{x}.M =_\beta \lambda\vec{x}\vec{y}.z.\vec{N},$$

which means that $\lambda\vec{x}.M$ is solvable. Contradiction.

($\Rightarrow$) Let $M$ be unsolvable and suppose that $\lambda\vec{x}.M$ is solvable. Then $\lambda\vec{x}.M$ has a head normal form and by the Church–Rosser property

$$\lambda\vec{x}.M =_\beta \lambda\vec{x}\vec{y}.z.\vec{N},$$

i.e., $M$ is solvable. Contradiction. □

Now, the Genericity Lemma for contexts and $\beta\eta$-equality is a consequence of the Genericity Lemma for lambda terms (Proposition 4.2.7).

**Proposition 4.2.12.** *Let $M, N \in \Lambda$ with $M$ unsolvable and $N$ having normal form. Then for all contexts $C[\ ]$*

$$C[M] = N \Rightarrow \forall L \in \Lambda \ \ C[L] = N.$$

**Proof.** Let $C[M] = N$ with $M$ unsolvable and $N$ having normal form. By Corollary 4.2.10 there is a set of variables $\{\vec{x}\}$ and a term $F \in \Lambda^0(Fv(C))$ such that

$$C[M] = F(\lambda\vec{y}.M) = N,$$

where $\{\vec{y}\} = \{\vec{x}\} \cup Fv(C)$. Now, by Lemma 4.2.11 $\lambda\vec{y}.M$ is unsolvable. Therefore by Proposition 4.2.7

$$\forall P \in \Lambda \quad FP = N.$$

Hence, for each $L \in \Lambda$ one has $F(\lambda\vec{y}.L) = N$. Again by Corollary 4.2.10

$$F(\lambda\vec{y}.L) = C[L] = N. \quad □$$

If $\Gamma$ and $\Delta$ are two basis with disjoint term variables, then the direct sum $\Gamma + \Delta$ is acctually their union $\Gamma \cup \Delta$. It is possible to establish a connection (suggested by M. Coppo) between two normal forms typable with a principal type of one of them in the following way:

**Proposition 4.2.13.** *Let $M$ and $N$ be normal forms and let $\Gamma \vdash N : \pi$ be a principal typing of $N$. Then*

$\Gamma + \{y_i : \vec{\delta_i} \to \beta_i\} \vdash M : \pi$ *where* $\beta_i \notin \Gamma \cup \{\pi\}$ *and* $y_i \notin \Gamma \Rightarrow \Gamma \vdash M : \pi$ *and* $N \twoheadrightarrow_\eta M$.

**Proof.** By induction on the argument (formation of a principal typing of $N$, construction of $M$) ordered lexicographically.

**Case 1.** $N \equiv x_i$. Then $x_1 : \alpha_1, \ldots, x_n : \alpha_n \vdash x_i : \alpha_i$ is a principal typing of $N$. Let

$$x_1 : \alpha_1, \ldots, x_n : \alpha_n + \{y : \vec{\delta_i} \to \beta_i\} \vdash M : \alpha_i.$$

Then $M$ is not an abstraction, but it cannot be an application either, because the types of all of its free variables are actually type variables. Thus $M \equiv x_i$ and, obviously, $x_1 : \alpha_1, \ldots, x_n : \alpha_n \vdash M : \alpha_i$.

**Case 2.** $N \equiv \lambda x.N'$ and the principal typing
$x_1 : \varphi_1, \ldots, x_n : \varphi_n \vdash \lambda x.N' \varphi \rightarrow \psi$ of $N$ is obtained from the principal typing $x : \varphi, x : \varphi_1, \ldots, x_n : \varphi_n \vdash N' : \psi$ of $N'$. Let

$$x_1 : \varphi_1, \ldots, x_n : \varphi_n + \{y_i : \vec{\delta_i} \rightarrow \beta_i\} \vdash M : \varphi \rightarrow \psi.$$

**Subcase 1.** $M \equiv x_i$, then $x_i : \varphi \rightarrow \psi$ is in the basis. So we have

$$x : \varphi, x_1 : \varphi_1, \ldots, x_i : \varphi \rightarrow \psi, \ldots, x_n : \varphi_n + \{y_i : \vec{\delta_i} \rightarrow \beta_i\} \vdash x_i x : \psi.$$

The length of $x_i x$ is greater than the length of $M$, but since the induction argument is ordered lexicographically by the induction hypothesis
$N' \twoheadrightarrow_\eta x_i x$ so

$$N \equiv \lambda x.N' \twoheadrightarrow_\eta \lambda x.x_i x \rightarrow_\eta x_i \equiv M.$$

**Subcase 2.** $M \equiv \lambda y.M'$. Then from

$$x_1 : \varphi_1, \ldots, x_n : \varphi_n + \{y_i : \vec{\delta_i} \rightarrow \beta_i\} \vdash \lambda x.M' : \varphi \rightarrow \psi$$

by $(\rightarrow E)$ we have

$$x : \varphi, x_1 : \varphi_1, \ldots, x_n : \varphi_n + \{y_i : \vec{\delta_i} \rightarrow \beta_i\} \vdash M' : \psi.$$

Then by the induction hypothesis $x : \varphi, x_1 : \varphi_1, \ldots, x_n : \varphi_n \vdash M' : \psi$ and $N' \twoheadrightarrow_\eta M'$, so $N \equiv \lambda x.N' \twoheadrightarrow_\eta \lambda x.M' \equiv M$.

**Subcase 3.** $M \equiv yM_1 \ldots M_k$. Then $z \equiv x_j$ for some $j, 0 \leq j \leq n$, and from

$$x_1 : \varphi_1, \ldots, x_n : \varphi_n \vdash yM_1 \ldots M_k : \varphi \rightarrow \psi$$

by $(\rightarrow E)$ we obtain

$$x : \varphi, x_1 : \varphi_1, \ldots, x_n : \varphi_n + \{y_i : \vec{\delta_i} \rightarrow \beta_i\} \vdash yM_1 \ldots M_k x : \psi,$$

where $x$ is a fresh term variable, i.e., $x \notin \{x_1, \ldots, x_n\}$ and so $x \notin Fv(M)$. Again, because of the lexicographical order of the induction argument we have that $N' \twoheadrightarrow_\eta yM_1 \dots M_k x$ and $x : \varphi, x_1 : \varphi_1, \dots, x_n : \varphi_n \vdash yM_1 \dots M_k x : \alpha$. Thus $N \equiv \lambda x.N' \twoheadrightarrow_\eta \lambda x.yM_1 \dots M_k x \rightarrow_\eta yM_1 \dots M_k \equiv M$.

**Case 3.** $N \equiv xN_1 \dots N_l$ and the principal typing

$$x : \bigcap_{i=1}^{l} \varphi_i \cap (\psi_1 \rightarrow \dots \rightarrow \psi_l \rightarrow \alpha), x_1 : \bigcap_{i=1}^{l} \varphi_1^i, \dots, x_n : \bigcap_{i=1}^{l} \varphi_n^i \vdash xN_1 \dots N_l : \alpha$$

of $N$ is obtained from the principal typings of $N_i$, $1 \leq i \leq l$, which are $x : \varphi_i, x_1 : \varphi_1^i, \dots, x_n : \varphi_n^i \vdash N_i : \psi_i$. Let us denote the basis of the principal typing of $N$ with $\Gamma$. Suppose

$$\Gamma + \{y_j : \vec{\delta}_j \rightarrow \beta_j\} \vdash M : \alpha.$$

Since $\alpha$ is a fresh type variable not occuring in the principal typings of $N_i$, $1 \leq i \leq l$ it follows that

- $M$ cannot be a variable;
- $M$ cannot be an abstraction.

So the only possible subcase is $M = yM_1 \dots M_k$. From

$$\Gamma + \{y_j : \vec{\delta}_j \rightarrow \beta_j\} \vdash yM_1 \dots M_k : \alpha$$

we have that $y \equiv x$, since in the basis $\Gamma$ the type $\alpha$ appears only in the type of $x$, and that $l = k$. Then $\Gamma + \{y_j : \vec{\delta}_j \rightarrow \beta_j\} \vdash M_i : \psi_i$ for all $1 \leq i \leq l$, because of the way of constructing the principal typing. By the induction hypothesis since $\alpha \notin \{\varphi_i, \varphi_1^i, \dots, \varphi_n^i\}$ for all $1 \leq i \leq l$

$$x : \varphi_i, x_1 : \varphi_1^i, \dots, x_n : \varphi_n^i \vdash M_i : \psi_i$$

and $N_i \twoheadrightarrow_\eta M_i$. So $N \equiv xN_1 \dots N_l \twoheadrightarrow_\eta xM_1 \dots M_l \equiv M$. $\Box$

Now, the proof of the Genericity Lemma is a straightforward application of Proposition 4.2.13.

**Proof 2. of Proposition 4.2.7.** If we proceed as in Proof 1 we obtain that

$$\Gamma \vdash FL : \pi$$

Since $\Gamma$ and $\pi$ do not contain $\omega$ by Theorem 4.2.1 (i), $FL$ has a normal form, say $Q$. Thus

$$FL =_\beta Q \quad \text{and} \quad \Gamma \vdash Q : \pi$$

Then by Proposition 4.2.13 $Q \twoheadrightarrow_\eta N$ and so $FL =_{\beta\eta} N$. □

## 4.3 Type topology

In this Section we shall deal with a new topology on the set of closed lambda terms $\Lambda^0$ introduced via typability of lambda terms in $\lambda\cap$. We shall call it *type topology*. The filter topology on $\Lambda^0$, given in Barendregt et al., 1983, and the introduced type topology appear to be the same. This is proved in Proposition 4.3.16. This topology appears to be a more simple description of the filter topology. The main difference between the filter topology and the type topology is that the former is a topology introduced on types and then traced on terms by the inverse map, while the later is introduced directly on terms. An open set, say $\mathcal{V}_\sigma$, in the type topology consists of all terms typable by $\sigma$, i.e., $\mathcal{V}_\sigma = \{M \in \Lambda^0 | \vdash M : \sigma\}$.

In terms of the type topology we shall consider Continuity Theorem (Theorem 4.3.4) of the untyped lambda calculus expressed in Barendregt, 1984, using the Böhm tree topology (Chapter 10, 14, 20). In Proposition 4.3.7 we prove that unsolvable terms are compactification points and $\beta\eta$–normal forms are isolated points in the type topology.

Let us consider a set of all lambda terms that can be typable by the same type, say

$$\mathcal{V}_\sigma = \{M \in \Lambda^0 | \vdash M : \sigma\}.$$

**Lemma 4.3.1.**

*(i) $\sigma$ is inhabited by a closed term if and only if $\mathcal{V}_\sigma \neq \emptyset$.*

*(ii) $\mathcal{V}_\sigma \cap \mathcal{V}_\tau = \mathcal{V}_{\sigma\cap\tau}$.*

**Proof.**

(i) Obvious.

(ii) From $M \in \mathcal{V}_\sigma \cap \mathcal{V}_\tau$ follows $\vdash M : \sigma$ and $\vdash M : \tau$ and thus by $(\cap I)$ $\vdash M : \sigma \cap \tau$. Conversely, if $\vdash M : \sigma \cap \tau$ by $(\cap E)$ $M \in \mathcal{V}_\sigma$ and $M \in \mathcal{V}_\tau$. □

**Proposition 4.3.2.** *The sets $\mathcal{V}_\sigma, \sigma \in T$ form a basis for a topology on $\Lambda^0$. This topology will be called type topology.*

**Proof.**

(i) Every lambda term is typable in $\lambda\cap$; so for every closed lambda term $M \in \Lambda^0$ there is a type $\tau \in T$ such that $\vdash M : \tau$. That is $M \in \mathcal{V}_\tau$.

(ii) For every two set $\mathcal{V}_\sigma$ and $\mathcal{V}_\tau$, by Lemma 4.3.1, $\mathcal{V}_\sigma \cap \mathcal{V}_\tau = \mathcal{V}_{\sigma\cap\tau}$. □

*Open sets* in the type topology are defined in the usual way.

**Definition 4.3.3.** *A set $\mathcal{O} \subseteq \Lambda^0$ is open if for any term $M \in \mathcal{O}$ there is a set $\mathcal{V}_\rho$ such that*

$$M \in \mathcal{V}_\rho \quad \text{and} \quad \mathcal{V}_\sigma \subseteq \mathcal{O}.$$

**Theorem 4.3.4. (Continuity Theorem)**
*Given $F \in \Lambda^0$. Then the map $M \mapsto FM$ is continuous ( with respect to the type topology).*

**Proof.** We have to show that

$$(\forall \mathcal{V}_\varepsilon \ni FM)(\exists \mathcal{V}_\delta \ni M)(Q \in \mathcal{V}_\delta \Rightarrow FQ \in \mathcal{V}_\varepsilon).$$

If $FM \in \mathcal{V}_\varepsilon$, then $\vdash FM : \varepsilon$. By the structural property given in Proposition 1.3.5.(i) there is a type $\delta$ such that

$$\vdash F : \delta \to \varepsilon \text{ and } \vdash M : \delta.$$

Then if $Q \in \mathcal{V}_\delta$, i.e., $\vdash Q : \delta$ we obtain $\vdash FQ : \varepsilon$. □

Let us consider some properties of the introduced topology that are related with $\beta$- and $\eta$- reduction.

**Lemma 4.3.5.** *Let $M, N \in \Lambda^0$.*

*(i) If $M \twoheadrightarrow_\beta N$, then $\forall \sigma \in T\ (M \in \mathcal{V}_\sigma \Leftrightarrow N \in \mathcal{V}_\sigma)$.*

*(ii) If $M \twoheadrightarrow_\eta N$, then $\forall \sigma \in T\ (M \in \mathcal{V}_\sigma \Rightarrow N \in \mathcal{V}_\sigma)$.*

**Proof.**

(i) It is obvious that $M \twoheadrightarrow_\beta N$ implies that $\forall \sigma \in T\ (M \in \mathcal{V}_\sigma \Rightarrow N \in \mathcal{V}_\sigma)$. But it implies that $\forall \sigma \in T\ (N \in \mathcal{V}_\sigma \Rightarrow M \in \mathcal{V}_\sigma)$ as well, since $\lambda\cap$ is closed under $\beta$-expansion.

(ii) $M \twoheadrightarrow_\eta N$ implies that $\forall \sigma \in T\ (M \in \mathcal{V}_\sigma \Rightarrow N \in \mathcal{V}_\sigma)$ since $\lambda\cap$ is closed under $\eta$-reduction, but it does not imply the other implication because it is not closed under $\eta$-expansion.

A counterexample is $1 \equiv \lambda xy.xy \to_\eta \lambda x.x \equiv \mathsf{I}$, but there are types of $\mathsf{I}$, which are not types of $1$ such as $\alpha \to \alpha$, where $\alpha$ is a type variable, since

$$\vdash \lambda x.x : \alpha \to \alpha, \text{ but } \nvdash \lambda xy.xy : \alpha \to \alpha.$$

Hence $\mathsf{I} \in \mathcal{V}_{\alpha \to \alpha}$, but $1 \notin \mathcal{V}_{\alpha\to\alpha}$. □

**Lemma 4.3.6.** *Let $N \in \Lambda^0$ be a $\beta\eta$-normal form. Then*

$$M \twoheadrightarrow_\beta N \text{ if and only if } \forall \sigma \in T\ (M \in \mathcal{V}_\sigma \Leftrightarrow N \in \mathcal{V}_\sigma).$$

**Proof.** ($\Rightarrow$) By Lemma 4.3.5.
($\Leftarrow$) Let $\pi \in T$ be a principal type of $N$, so $\vdash N : \pi$. Hence $\vdash M : \pi$. Therefore by Proposition 4.2.6(ii) $M \twoheadrightarrow_\beta N$. □

Lambda terms that are $\beta$-equal belong to same open sets therefore we shall identify them. By "up to $\beta$-equality" we mean that if a normal form $N$ is in a set, then all the terms that are $\beta$-equal to $N$ are in the same set.

**Proposition 4.3.7.**

*(i) Unsolvable terms are compactification points.*

*(ii) If $N$ is a normal form, then there is a type $\rho \in T$ such that $\mathcal{V}_\rho = \{P|N \twoheadrightarrow_\eta P\}$ up to $=_\beta$.*

*(iii) $\beta\eta$-normal forms are isolated points up to $=_\beta$.*

**Proof.**

(i) If $M$ is an unsolvable term, then by Theorem 4.2.1 $M \in \mathcal{V}_\sigma$ such that $\sigma \sim \omega$. But then $\mathcal{V}_\sigma = \mathcal{V}_\omega = \Lambda^0$. Therefore $\Lambda^0$ is the only open set containing unsolvable terms, and hence they are compactification points.

(ii) If $N$ is a normal form, then there is a principal typing $\vdash_{-\omega} N : \pi$. For every term $M$ for which $\vdash M : \pi$, i.e., $M \in \mathcal{V}_\pi$ by Proposition 4.2.6 there is an $\eta$-reduct $P$ of $N$ such that $M \twoheadrightarrow_\beta P$. By Lemma 4.3.5 (ii) $P \in \mathcal{V}_\pi$, also. By Lemma 4.2.3 $P$ is a normal form as well. Hence $\mathcal{V}_\pi = \{P|N \twoheadrightarrow_\eta P\}$ up to $\beta$-equality, since $M =_\beta P$ .

(iii) Obvious, since if $N$ is a $\beta\eta$-normal form, then $\mathcal{V}_\pi$ from (ii) is a singleton, i.e., $\mathcal{V}_\pi = \{N\}$ up to $\beta$-equality. □

And now, finaly, the third proof of the Genericity Lemma for closed terms.

**Proof 3 of Proposition 4.2.7.** Let $N_{\beta\eta}$ be the $\beta\eta$-normal form of $N$, $N \twoheadrightarrow_{\beta\eta} N_{\beta\eta}$. If $\pi \in T$ is a principal type of $N_{\beta\eta}$, then by Proposition 4.3.7

(iii) $\mathcal{V}_\pi = \{N_{\beta\eta}\}$ is a singleton up to $=_\beta$. Again by Proposition 4.3.7 (i) $\mathcal{V}_\omega = \Lambda^0$ is the only open set containing the unsolvable term $M$. By the Continuity theorem 4.3.4

$$(\forall \mathcal{V}_\varepsilon \ni N)(\exists \mathcal{V}_\delta \ni M)(L \in \mathcal{V}_\delta \Rightarrow FL \in \mathcal{V}_\varepsilon).$$

Let us choose $\mathcal{V}_\varepsilon = \mathcal{V}_\pi = \{N_{\beta\eta}\}$. $\mathcal{V}_\delta = \mathcal{V}_\omega = \Lambda^0$, and so if $L \in \Lambda^0$, then $FL = N_{\beta\eta}$ and $\vdash FL : \varepsilon$. By Proposition 4.2.6(ii) this means that $FL \twoheadrightarrow_\beta N_{\beta\eta}$. and so $FL =_{\beta\eta} N$. □

Now, we shall compare the type topology and the filter topology on $\Lambda^0$ introduced in Barendregt et al., 1983. So, first let us recall some basic notions of the filter topology.

Let $\mathcal{F} \subseteq \mathcal{P}(T)$ be a *filter model* (see Definition 1.3.15 and 1.3.17). The *valuation of closed terms* $\|\ \|: \Lambda^0 \to \mathcal{F}$ is given by the following mapping

$$\|M\| = \{\sigma | \vdash M : \sigma\} \in \mathcal{F}.$$

A Scott topology is defined on $\mathcal{F}$ in the following way:

**Definition 4.3.8.** *A set $\mathcal{O} \subseteq \mathcal{F}$ is open if:*

*(i) $d \in \mathcal{O}$ and $d \subseteq e$, then $e \in \mathcal{O}$;*

*(ii) $\cup d_i \in \mathcal{O}$, then there is an $i_0$ such that $d_{i_0} \in \mathcal{O}$.*

This topology induces the so called filter topology on $\Lambda^0$ in the sense that open sets in $\Lambda^0$ are $\|\mathcal{O}\|^{-1} \subseteq \Lambda^0$, where $\mathcal{O} \subseteq \mathcal{F}$ is open in the given topology.

Let $\uparrow \sigma = \{\tau | \sigma \leq \tau\}$ denote the principal filter generated by the type $\sigma$ and let $d \uparrow = \{e | d \subseteq e\}$ be the upper closure of the filter $d$.

**Definition 4.3.9.** *A filter $d$ is compact if $d \uparrow$ is open.*

In order to compare the filter topology and the type topology we need some investigations on the type topology. Also, we shall recall some properties of the filter topology.

**Lemma 4.3.10.** *Let $\sigma \in T$. Then the principal filter $\uparrow \sigma = \{\tau | \sigma \leq \tau\}$ is compact.*

**Proof.** In order to show that $\uparrow \sigma$ is compact, we show that $(\uparrow \sigma) \uparrow = \{d | \uparrow \sigma \subseteq d\} \subseteq \mathcal{F}$ is open in the filter topology.

(i) Let $d \in (\uparrow \sigma) \uparrow$ and $d \subseteq e$. It is obvious that $e \in (\uparrow \sigma) \uparrow$.

(ii) If $\bigcup_{i \in I} d_i \in (\uparrow \sigma) \uparrow$, then $\uparrow \sigma \subseteq \bigcup_{i \in I} d_i$. It means that $\sigma \in \bigcup_{i \in I} d_i$ and then there exists $i_0 \in I$ such that $\sigma \in d_{i_0}$. Thus $\uparrow \sigma \subseteq d_{i_0}$, i.e., $d_{i_0} \in (\uparrow \sigma) \uparrow$. □

**Lemma 4.3.11.** *Let $M \in \Lambda$. Then $\|M\| \in \mathcal{F}$ is not in general a principal filter.*

**Proof.** Suppose $\|M\| \in \mathcal{F}$ is a principal filter. Then there is a $\sigma \in T$ such that $\|M\| = \uparrow \sigma$. But then for all $\tau \in T$

$$\vdash M : \tau \quad \Leftrightarrow \quad \sigma \leq \tau,$$

which does not hold in general, e.g. it does not hold for I, but it holds for $\Omega$. □

**Lemma 4.3.12.** *Let $M \in \Lambda$.*

$$\|M\| \in \mathcal{F} \ \textit{ is not compact in general.}$$

**Proof.** Suppose $\|M\|$ is compact. Then $\|M\| \uparrow$ is open. It is obvious that $\|M\| \subseteq \bigcup_{\substack{\vdash M:\sigma_i \\ (\sigma_i \in \|M\|)}} \uparrow \sigma_i$, thus $\bigcup_{\sigma \in \|M\|} \uparrow \sigma_i \in \|M\| \uparrow$. But then since $\|M\| \uparrow$ is open, there is a $\sigma_{i_0} \in \|M\|$ sucht that $\uparrow \sigma_{i_0} \in \|M\| \uparrow$. Thus $\|M\| \subseteq \uparrow \sigma_{i_0}$. On the other hand, it is easy to show that $\uparrow \sigma_{i_0} \subseteq \|M\|$. Therefore $\|M\| = \uparrow \sigma_{i_0}$, which is impossible according to Lemma 4.3.11. □

**Lemma 4.3.13.** *Let $\sigma \in T$. Then the set $\|(\uparrow \sigma) \uparrow \|^{-1} = \mathcal{V}_\sigma$ and hence it is open in the type topology.*

**Proof.**

$$\begin{aligned} P \in \|(\uparrow \sigma)\uparrow\|^{-1} &\Leftrightarrow \|P\| \in (\uparrow \sigma)\uparrow \\ &\Leftrightarrow \uparrow \sigma \subseteq \|P\| \\ &\Leftrightarrow \sigma \in \|P\| \\ &\Leftrightarrow \vdash P : \sigma \\ &\Leftrightarrow P \in \mathcal{V}_\sigma. \qquad \Box \end{aligned}$$

**Proposition 4.3.14.** *If $\mathcal{O} \subseteq \Lambda$ is open in the filter topology, then $\mathcal{O}$ is open in the type topology.*

**Proof.** In order to show that $\mathcal{O} \subseteq \Lambda$ is open in the type topology we shall show that for every $M \in \mathcal{O}$ there is a set $\mathcal{S}$ open in the type topology such that $M \in \mathcal{S}$ and $\mathcal{S} \subseteq \mathcal{O}$.

Let $M \in \mathcal{O}$. Then there is a set $\mathcal{O}_1 \subseteq \mathcal{F}$ open in the filter topology such that $\|\mathcal{O}_1\|^{-1} = \mathcal{O}$ and $\|M\| \in \mathcal{O}_1$. On the other hand $\|M\| \subseteq \bigcup_{\sigma_i \in \|M\|} \uparrow \sigma_i$. Since $\mathcal{O}_1$ is open, we have that $\bigcup_{\sigma_i \in \|M\|} \uparrow \sigma_i \in \mathcal{O}_i$. But then there is a $\sigma_{i_0} \in \|M\|$ such that $\uparrow \sigma_{i_0} \in \mathcal{O}_1$. Thus $(\uparrow \sigma_{i_0}) \uparrow \subseteq \mathcal{O}_1$ which means that $\|(\uparrow \sigma_{i_0}) \uparrow\|^{-1} \subseteq \mathcal{O}$. By Lemma 4.3.13 $\|(\uparrow \sigma_{i_0}) \uparrow\|^{-1} = \mathcal{V}_{\sigma_{i_0}}$ is open in the type topology. □

**Proposition 4.3.15.** *Let $\sigma \in T$. Then the set $\mathcal{V}_\sigma$ is open in the filter topology.*

**Proof.** By Lemma 4.3.13 $\mathcal{V}_\sigma = \|(\uparrow \sigma)\uparrow\|^{-1}$. On the other hand by Lemma 4.3.10, $(\uparrow \sigma) \uparrow$ is open in the filter topology. Thus by Definition $\|(\uparrow \sigma)\uparrow\|^{-1}$ is open in the filter topology. □

As shown in Propositions 4.3.14 and 4.3.15 sets open in the filter topology are open in the type topology and vice versa. Therefore the filter topology and the type topolgy are the same. The advantage of the type topology is that it is defined on terms directly, while the filter topology is actually induced by the topology defined on filters.

## References

[1] Allesi. F. and F. Barbanera. 1991. Strong conjunction and intersection types, MFCS Proceedings 520, pp. 64-73.

[2] Barbanera. F. and M. Dezani-Ciancaglini. 1991. Intersection and union types, TACS '91, Lecture Notes in Computer Science 526, Springer. Berlin. pp. 651-674.

[3] Barbanera, F., M. Dezani-Ciancaglini and U. de'Liguoro. 1993. Intersection and Union Types: Syntax and Sematics (to appear).

[4] Bakel, S. van, 1992, Complete restrictions of the intersection type discipline, Theoretical Computer Science 102. pp. 136–163.

[5] Bakel. S. van, 1993, Intersection Type Assignment in Lambda Calculus and Term Rewriting Systems, Ph. D. Thesis. University of Nijmegen.

[6] Barendregt, H.P., 1984, Lambda Calculus  Its Syntax and Semantics, North–Holland, Amsterdam.

[7] Barendregt, H.P., 1992, Lambda calculi with types, in Handbook of Logic in Computer Science, S. Abramsky, D.M. Gabbay and T.S.E. Maibaum eds, Oxford University Press, Oxford, Vol.II, pp. 117-309.

[8] Barendregt, H.P., M. Bunder and W. Dekkers. 1992, Systems of Illative Combinatory Logic complete for first-order propositional and predicate calculus (to appear).

[9] Barendregt, H.P., M. Coppo and M. Dezani–Ciancaglini, 1983, A filter model and the completeness of type assignment, The Journal of Symbolic Logic 48, pp. 931–940.

[10] Ben–Yelles, C., 1979, Type–Assignment in the Lambda Calculus: Syntax and Semantics, Ph. D. Thesis, University College of Swansea.

[11] Coppo, M., M. Dezani–Ciancaglini, and B. Venneri, 1980, Principal type schemes and $\lambda$–calculus semantics, in To H.B. Curry: Essays on Combinatory Logic, Typed Lambda Calculus and Formalism, J.R. Hindley and J.P. Seldin, eds, Academic Press, pp. 480–490.

[12] Coppo, M., M. Dezani-Ciancaglini and B. Venneri, 1981, Functional characters of solvable terms, Zeitshrift für Mathematische Logik und Grundlagen der Mathematik 27, pp. 45-58.

[13] Church, A., 1941, The Calculi of Lambda Conversion, Princeton University Press, Princeton.

[14] Curry, H.B. and R. Feys, 1958, Combinatory Logic, vol. I, North-Holland, Amsterdam.

[15] Dezani-Ciancaglini, M. and J.R. Hindley, 1992, Intersection types for combinatory logic, Theoretical Computer Science 100, pp. 303-324.

[16] Dezani-Ciancaglini, M., S. Ghilezan, and B. Venneri, 1992, The "relevance" of intersection and union types (to appear).

[17] Došen, K., 1988/1989/1990, Sequent-systems and groupoid models, Studia Logica 47, pp. 353-385/ 48, pp. 41-65./ 49, p. 614.

[18] Došen, K., 1992, A brief survey of frames for the Lambek calculus, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 38, pp. 179-187.

[19] Geuvers, J.H., 1988, The interpretation of logics in type systems, Master's thesis, Catholic University, Nijmegen.

[20] Ghilezan, S., 1992, Peirce's law and lambda calculus, Proceedings of Logic and Computer Science, Novi Sad (to appear).

[21] Ghilezan, S., 1993, Inhabitation in the intersection and union type assignment systems (to appear in The Journal of Logic and Computation).

[22] Gilezan, S., 1988, Lambda calculus and weak propositional logics, Master's tesis (in SerboCroatian), University of Belgrade.

[23] Girard, J.-Y., 1972, Interprétation fonctionelle et élimination des coupures dans l'arithmetique d'ordre supérieur, Ph. D. Thesis, Université Paris VII.

[24] Girard, J.-Y.,Y. Lafont and P. Taylor, 1990, Proofs and Types, Tracts in Theoretical Computer Science 7, Cambridge University Press, Cambridge.

[25] Hindley, J.R., 1969, The principal type scheme of an object in combinatory logic, Transactions of the American Mathematical Society 146, pp. 29-60.

[26] Hindley, J.R., 1982, The simple semantics of Coppo–Dezani Sallé types, ISP'82, Lecture Notes in Computer Science 137, Springer, Berlin, pp. 212–226.

[27] Hindley, J.R., 1984, Coppo–Dezani types do not correspond to propositional logic, Theoretical Computer Science 28, pp. 235–236.

[28] Hindley, J.R. and J.P. Seldin, 1986, Introduction to Combinators and Lambda Calculus, Cambridge University Press, Cambridge.

[29] Hindley, J.R. and J.P. Seldin, eds, 1980, To H.B. Curry: Essays on Combinatory Logic, Typed Lambda Calculus and Formalism, Academic Press, New York.

[30] Howard, W.A., (1969) 1980, The formulae-as-types notion of construction, in To H.B. Curry: Essays on Combinatory Logic, Typed Lambda Calculus and Formalism, J.R. Hindley and J.P. Seldin, eds, Academic Press, New York, pp. 479–490.

[31] Krivine, J.L., 1990, Lambda–calcul, types et modèles, Masson, Paris.

[32] Lambek, J. and P.J. Scott, 1986, Introduction to Higher Order Categorical Logic, Cambridge University Press, Cambridge.

[33] Leivant, D., 1983, Polymorphic type inference, in Proceedings $10^{th}$ ACM Symposium on Principles of Programming Languages, Austin Texas, pp. 88–98.

[34] Lopez–Escobar, E.G.K., 1985, Proof functional connectives, Lecture Notes in Mathematics 1130, Springer, Berlin, pp. 208–221.

[35] Mints, G.E., 1989, The completeness of provable realizability, Notre Dame Journal of Formal Logic 30, pp. 420–441.

[36] Montegue, R., 1979, Formal Philosophy, Selected Papers of Richard Montegue, R. Thomason, ed., Yale University Press, New Haven.

[37] Nerode, A. and P. Odifreddi, 1990, Lambda calculi and constructive logics, Technical Report, Cornell University.

[38] Penrose, R., 1989, The Emperor's New Mind, Oxford University Press, Oxford.

[39] Prawitz, D., 1965, Natural Deduction: A Proof Theoretical Study, Ålmqvist and Wiksell, Stockholm.

[40] Pottinger, G., 1980, A type assignment for the strongly normalizable $\lambda$–terms, in To H.B. Curry: Essays on Combinatory Logic, Typed Lambda Calculus and Formalism, J.R. Hindley and J.P. Seldin, eds, Academic Press, New York, pp. 561–577.

[41] Robinson, J.A., 1965, A machine oriented logic on the resolution principle, Journal of the Association for Computing Machinery 12 (1), pp. 23-41.

[42] Ronchi della Rocca, S. and B. Venneri, 1984, Principal type schemes for an extended type theory, Theoretical Computer Science 28, pp. 151-169.

[43] Schönfinkel, M., 1924, Über die Bausteine der mathematische Logik, Matematische Annalen 92, pp. 305-316.

[44] Scott, D.S., 1980, Lecture on a mathematical theory of computation, (manuscript).

[45] Stenlund, S., 1972, Combinators, Lambda Terms and Proof Theory, D. Reidel, Dordrecht.

[46] Smullyan, R., 1985, To Mock the Mocking Bird, Knop, New York.

[47] Tait, W.W., 1967, Intensional interpretation of functionals of finite type I, The Journal of Symbolic Logic 32, pp. 198–212.

[48] Tait, W.W., 1975, A realizability interpretation of the theory of species, in Logic Colloquium (Boston), R. Parikh, ed., Lecture Notes in Mathematics 453, Springer, Berlin, pp. 240–251.

[49] Venneri, B., 1992, Intersection types and logical formulae (to appear in The Journal of Logic and Computation).

[50] Vrijer, de R.C., 1987, Surjective Pairing and Strong Normalization: Two Themes in Lambda Calculus, Ph.D. Thesis, University of Amsterdam.

[51] Wadsworth, C.P., 1971, Semantics and Pragmatics of Lambda Calculus, D.Phil. Thesis, Oxford University.

[52] Yokouchi, H., 1991, Embedding second-order type systems into intersection type systems and its applications to type inference, Internal Report, IBM-Research Laboratory, Tokyo.