Intersection Types for the \(\lambda\mu\)-Calculus

Steffen van Bakel, Franco Barbanera, and Ugo de’Liguoro

Department of Computing, Imperial College London, 180 Queen’s Gate, London SW7 2BZ, UK
Dipartimento di Matematica e Informatica, Università degli Studi di Catania, Viale A. Doria 6, 95125 Catania, Italy
Dipartimento di Informatica, Università degli Studi di Torino, Corso Svizzera 185, 10149 Torino, Italy
s.vanbakel@imperial.ac.uk, barba@dmi.unict.it, ugo.deliguoro@unito.it

Abstract

We introduce an intersection type system for the pure \(\lambda\mu\)-calculus, which is invariant under subject reduction and expansion. The system is obtained by describing Streicher and Reus’s denotational model of continuations in the category of omega-algebraic lattices via Abramsky’s domain logic approach. This provides at the same time an interpretation of the type system and a proof of the completeness of the system with respect to the continuation models by means of a filter model construction.

We then define a restriction of ours system, such that a lambda-mu term is typeable if and only if it is strongly normalising. We also show that Parigot’s typing of lambda-mu terms with classically valid propositional formulas can be translated into the restricted system, which then provides an alternative proof of strong normalisability for the typed lambda-mu calculus.

keywords: \(\lambda\mu\)-calculus, intersection types, filter semantics, strong normalisation.

Introduction

The \(\lambda\mu\)-calculus is a pure, type-free calculus introduced by Parigot [43] to denote classical proofs and to compute with them. It is an extension of the proofs-as-programs paradigm where types can be understood as classical formulas and (closed) terms inhabiting a type as the respective proofs in a variant of Gentzen’s natural deduction calculus for classical logic [29]. The study of the syntactic properties of the \(\lambda\mu\)-calculus has been challenging, which led to the introduction of variants of term syntax, reduction rules, and typing as, for example, in de Groote’s variant of the \(\lambda\mu\)-calculus [32]. These changes have an impact on the deep nature of the calculus which emerges both in the typed and in the untyped setting [22, 48].

Types are of great help in understanding the computational properties of terms in an abstract way. Although in [14] Barendregt treats the theory of the pure \(\lambda\)-calculus without a reference to types, most of the fundamental results of the theory can be exposed in a quite elegant way by using the Coppo-Dezani intersection type system [18]. This is used by Krivine [36], where the treatment of the pure \(\lambda\)-calculus relies on intersection typing systems called \(\mathcal{D}\) and \(\mathcal{D}\Omega\).

The quest for more expressive notions of typing for \(\lambda\mu\) is part of an ongoing investigation into calculi for classical logic. In order to come to a characterisation of strong normalisation for Curien and Herbelin’s (untyped) sequent calculus \(\overline{\lambda\mu\overline{\mu}}\) [21], Dougherty, Ghielen and Lescanne presented System \(\mathcal{M}^{\cap\cup}\) [25, 26], that defines a notion of intersection and union typing for that calculus. However, in [7] van Bakel showed that this system is not closed.
under conversion, an essential property of Coppo-Dezani systems; in fact, it is shown that it is impossible to define a notion of typing for \( \lambda \mu \) that satisfies that property.

In [8] van Bakel brought intersection (and union) types to the context of the (untyped) \( \lambda \mu \)-calculus, and showed that for \( \lambda \mu \)-conversion it is possible to prove type preservation under conversion. However, union types are no longer dual to intersection types and play only a marginal role, as was also the intention of [26]. In particular, the normal \(( \cup I)\) and \(( \cup E)\) rules as used in [13], which are known to create a soundness problem in the context of the \( \lambda \)-calculus, are not allowed. Moreover, although one can link intersection types with the logical connector and, the union types used in [8] bear no relation with or; one could argue that therefore union might perhaps not be the right name to use for this type constructor. In the view of the above mentioned failure noted in [7], the result of [8] came as a surprise, and led automatically to the question we answer here: does a filter semantics for \( \lambda \mu \) exist?

The idea of building a \( \lambda \)-model out of a suitable type assignment system appeared first in [15]. In that system types are an extension of simple types with the binary operator \( \land \) for intersection, and are preordered by an axiomatisable (actually decidable) relation \( \leq \); if types are interpreted by subsets of the domain \( D \) (an applicative structure satisfying certain conditions), one can see \( \land \) as set theoretic intersection and \( \leq \) as containment. The discovery of [15] is that, if we take a proper relation \( \leq \), then the set \( F_D \) of filters of types (namely upward sets of types, closed under type intersection) is a new structure that is a \( \lambda \)-model, where (closed) terms can be interpreted by the set of types that can be assigned to them in the type system. This is what we call a filter semantics.

It emerged in [19] that models constructed as set of filters of intersection types are exactly the \( \omega \)-algebraic lattices, a category of complete lattices, but with Scott continuous maps as morphisms. \( \omega \)-algebraic lattices are posets whose structure is fully determined by a countable subset of elements called “compact points” for topological reasons; now the crucial fact is that given an \( \omega \)-algebraic lattice \( D \) the set \( K(D) \) of its compact points can be described by putting its elements into a one-to-one correspondence with a suitable set of intersection types, in such a way that the order over \( K(D) \) is reflected by the inverse of the \( \leq \) pre-order over types. Then one can show that the filter structure \( F_D \) obtained from the type pre-order is isomorphic with the original \( D \). In fact Abramsky proved that this is not true only of \( \omega \)-algebraic lattices, but of quite larger categories of domains, like 2/3-SFP domains, that can be finitely described by a generalization of intersection type theories, called the logics of the respective domains in [1].

Now, instead of guessing a suitable type system for \( \lambda \mu \), and then trying to prove that it actually induces a filter model, we follow the opposite route. We start from a model of the \( \lambda \mu \)-calculus in \( \omega \text{-AlgL} \), the category of \( \omega \)-algebraic lattices. Then we distill the type syntax and the corresponding type theory out of the construction of the model, and recover the typing rules from the clauses that define term interpretation in the given model \( F_D \), that is by construction isomorphic to the given \( D \).

However things are slightly more complex than this. First we need a domain theoretic model of \( \lambda \mu \), and we use for that purpose Streicher and Reus’s models of continuations. Building on Lafont’s ideas and the papers [37, 41], in [49] Streicher and Reus proposed a model of both typed and untyped \( \lambda \)-calculus embodying a concept of continuation, including Felleisen’s \( \lambda C \)-calculus [28, 27] and a version of Parigot’s pure \( \lambda \mu \). The model is based on the solution of the domain equations \( D = C \rightarrow R \) and \( C = D \times C \), where \( R \) is an arbitrary domain of “results”. The domain \( C \) is set of what are called “continuations” in [49], which are infinite tuples of elements in \( D \). \( D \) is the domain of continuous functions from \( C \) to \( R \) and it is the set of “denotations” of terms. We call the triple \(( R, D, C) \) a \( \lambda \mu \)-model, that exists in \( \omega \text{-AlgL} \) by the inverse limit technique, provided that \( R \in \omega \text{-AlgL} \).

The next step is to find type languages \( L_D \) and \( L_C \) and type theories axiomatizing the
respective pre-orders \( \leq_D \) and \( \leq_C \), such that \( D \) and \( C \) are isomorphic to \( F_D \) and \( F_C \) respectively. To this aim we may suppose that logical description of \( R \) is given via a language of types \( L_R \) and a pre-order \( \leq_R \), but then we need a detailed analysis of \( K(D) \) and \( K(C) \), keeping into account that \( D \) and \( C \) are both co-limits of certain chains of domains, and that their compact points are into one-to-one correspondence with the union of the compact points of the domains approximating \( D \) and \( C \). This leads us to a mutually inductive definition of \( \mathcal{L}_D \) and \( \mathcal{L}_C \) and of \( \leq_D \) and \( \leq_C \). In this way, we obtain an extension of the type theory used in \([15]\) which is a natural equated intersection type theory in terms of \([2]\) and hence isomorphic to the inverse limit construction of a \( D_\infty \) \( \lambda \)-model (as an aside, we observe that this matches perfectly with Theorem 3.1 in \([49]\)).

Once the filter domains \( F_D \) and \( F_C \) have been constructed, we can consider the interpretation of terms and of commands (“unnamed” and “named terms” respectively in Parigot’s terminology). Following \([49]\) we can in fact define the interpretation of expressions of Parigot’s \( \lambda\mu \)-calculus inductively via a set of equations. Guided by this equations in the particular case of \( F_D \) and \( F_C \), and considering the correspondence we have established among types and compact points, we are able to reconstruct the inference rules of a type assignment system that is the main product of our work.

The study of the properties of the system produces a series of results that confirm the validity of the construction. First we prove that in the filter model the meaning of \( M \) in the environment \( e \), denoted by \( \llbracket M \rrbracket_e \), coincides with the filter of all types \( \delta \in \mathcal{L}_D \) such that \( \Gamma \vdash M : \delta \mid \Delta \) is derivable in the system, for \( \Gamma \) and \( \Delta \) such that \( e \) satisfies both \( \Gamma \) and \( \Delta \), and similarly for \( \llbracket C \rrbracket_e \), where \( C \) is a command. It follows that if two terms or commands are convertible, then they must have the same types. As a matter of fact we prove this result twice: first abstractly, making essential use of the filter model construction; then concretely, by studying in depth the structure of the derivations in our system, and establishing that types are preserved under subject reduction and expansion.

We then face the problem of characterizing strong normalization in the case of \( \lambda\mu \). Indeed it is a characteristic property of intersection types, spotted the first time by Pottinger \([46]\) for ordinary \( \lambda \)-calculus, that strongly normalizing terms can be captured by certain “restricted” type systems, ruling out the universal type \( \omega \). As it will be apparent in the technical treatment, we cannot simply restrict our system by throwing away \( \omega \); however the characterization can be obtained by distinguishing certain “good” occurrences of \( \omega \) that cannot be eliminated, and the “bad” ones that must be forbidden. This is still guided by the semantics and by the proof theoretic study of the system, and we can establish that there exists a sub-system of ours which is determined just by a restriction on type syntax, plus the elimination of the rule \( \omega \) from our type system.

A last question we answer is about the relation between our type system and the original one proposed by Parigot \([43]\) on the basis a of the Curry-Howard correspondence between types and formulas and \( \lambda\mu \)-terms and proofs of classical logic. We show that there exists an interpretation of Parigot’s first order types into intersection types such that the structure of derivations is preserved; moreover the translations are all restricted intersection types, hence we obtain a new proof that all proof-terms in \( \lambda\mu \), namely typeable in Parigot’s system (even extended with negation) are strongly normalising.

Outline of this paper

The paper is organised as follows. After recalling the \( \lambda\mu \)-calculus in § 1, we study the domain theoretic models in § 2. In § 3 we introduce intersection types and type theories and we illustrate the filter model construction. The main part of the paper is § 4, where we introduce the type assignment system. We study type invariance under reduction and expansion in
§ 5. The § 6 is devoted to the characterisation of strongly normalising terms by means of a subsystem of ours obtained by suitably restricting the type syntax. Then, in § 7, we compare our system with Parigot’s, and show that Parigot’s types are translatable into our restricted types while preserving type derivability (in the two systems). We finish by citing some related works in § 8 and by drawing our conclusions.

This paper is the full version of [10], extended with a revised version of [11].

1 The untyped \(\lambda\mu\)-calculus

The \(\lambda\mu\)-calculus, as introduced in [43], is an extension of the untyped \(\lambda\)-calculus, obtained by adding names and a name-abstraction operator, \(\mu\). It was conceived in the spirit of Felleisen’s \(\lambda C\)-calculus, that Griffin showed to be typeable with classical propositional logic in [31].

In the same way, the \(\lambda\mu\)-calculus is type free, and uses names and name abstraction operator to model a form of functional programming with control. Here we briefly revise the basic notions of the \(\lambda\mu\)-calculus, though by slightly changing the notation and terminology, and defer the presentation of the typed \(\lambda\mu\)-calculus to § 7.

Definition 1.1 (Term Syntax [43]) The sets \(\text{Term}\) of terms (ranged over by \(M, N, L\)) and \(\text{Cmd}\) of commands (ranged over by \(C\)) are defined inductively by the following grammar, where \(x \in \text{Var}\), the set of term variables (ranged over by \(x, y, z\)) and \(\alpha \in \text{Name}\), the set of names (ranged over by \(\alpha, \beta, \gamma\)), both denumerable:

\[
\begin{align*}
M, N & ::= x \mid \lambda x. M \mid MN \mid \mu \alpha. C \\
C & ::= [\alpha] M
\end{align*}
\]

We let \(T\) range over \(\text{Term} \cup \text{Cmd}\).

As usual, \(\lambda x. M\) binds \(x\) in \(M\), and \(\mu \alpha. C\) binds \(\alpha\) in \(C\). A variable or a name occurrence is free in a term if it occurs and is not bound: we denote the free variables and the free names occurring in \(T\) by \(\text{fv}(T)\) and \(\text{fn}(T)\), respectively.

We identify terms and commands obtained by renaming of free variables or names, and we adopt Barendregt’s convention that free and bound variables and names are distinct, assuming silent \(\alpha\)-conversion during reduction to avoid capture. Note that we could have defined the set of pure \(\lambda\mu\)-terms via the grammar

\[
\begin{align*}
M, N & ::= x \mid \lambda x. M \mid MN \mid \mu \alpha.[\beta]M
\end{align*}
\]

but it is convenient to have commands as part of the syntax as well.

In [43] terms and commands are called ‘terms’ and ‘named terms’, respectively; also, names there are called \(\mu\)-variables, while they might be better understood as ‘continuation variables’ (see [49]); since this would imply a strong commitment to a particular interpretation, we prefer a more neutral terminology.

In the \(\lambda\mu\)-calculus, substitution takes the following three forms:

\[
\begin{align*}
term\ substitution: & \quad T[N/x] \quad (N\ is\ substituted\ for\ x\ in\ T,\ avoiding\ capture) \\
\text{renaming:} & \quad T[\alpha/\beta] \quad (every\ free\ occurrence\ of\ \beta\ in\ T\ is\ replaced\ by\ \alpha) \\
\text{structural substitution:} & \quad T[\alpha \leftarrow L] \quad (every\ subterm\ [\alpha]N\ of\ T\ is\ replaced\ by\ [\alpha]|NL)
\end{align*}
\]

In particular, structural substitution is defined by induction over terms and commands as follows:
Definition 1.2 (Structural Substitution) The base case for the structural substitution is defined as:

\[(\alpha M) [\alpha \leftarrow L] \triangleq [\alpha](M[\alpha \leftarrow L])L\]

The other cases are defined as:

\[x [\alpha \leftarrow L] \triangleq x\]
\[(\lambda x. M) [\alpha \leftarrow L] \triangleq \lambda x.M[\alpha \leftarrow L]\]
\[(MN) [\alpha \leftarrow L] \triangleq (M[\alpha \leftarrow L])(N[\alpha \leftarrow L])\]
\[(\mu \beta. C) [\alpha \leftarrow L] \triangleq \mu \beta. C[\alpha \leftarrow L]\]
\[(\beta M) [\alpha \leftarrow L] \triangleq [\beta]M'[\alpha \leftarrow L] \quad \text{if } \alpha \neq \beta\]

Notice that the base case places the argument of the substitution to the right of the term \(M\), but within the scope of the name \(\alpha\), and propagates the substitution towards subterms of \(M\) that are named \(\alpha\) as well.

The reduction relation for the \(\lambda \mu\)-calculus is defined as follows.

Definition 1.3 (Reduction \(\rightarrow_{\beta \mu}\) [43]) The reduction relation \(\rightarrow_{\beta \mu}\) is the compatible closure of the following rules:

\[(\beta) : (\lambda x. M)N \rightarrow M[N/x]\]
\[(\mu) : (\mu a. C)N \rightarrow \mu a. C[\alpha \leftarrow N]\]
\[(\text{ren}) : [\alpha] \mu \beta. C \rightarrow C[\alpha / \beta].\]

Note that Rule \((\beta)\) is the normal \(\beta\)-reduction rule of the \(\lambda\)-calculus. Rule \((\text{ren})\), after ‘renaming’, is called ‘structural reduction’ in [43] and rule \((\rho)\) in [47]; it is an auxiliary notion of reduction, aimed at simplifying proof terms. Rule \((\mu)\) is characteristic for \(\lambda \mu\); the intuition behind this rule has been explained by de Groote in [32], by arguing on the ground of the intended typing of the \(\mu\)-terms: ‘in a \(\lambda \mu\)-term \(\mu a. M\) of type \(A \rightarrow B\), only the subterms named by \(\alpha\) are really of type \(A \rightarrow B\) \((\ldots)\); hence, when such a \(\mu\)-abstraction is applied to an argument, this argument must be passed over to the sub-terms named by \(\alpha\).’

Py [47] has shown that the reduction relation \(\rightarrow_{\beta \mu}\) is confluent. Therefore the convertibility relation \(\equiv_{\beta \mu}\) determined by \(\rightarrow_{\beta \mu}\) is consistent in the usual sense that distinct normal forms are not equated.

Definition 1.4 (The theory \(\lambda \mu\)) The theory \(\lambda \mu\) is the equational theory determined by the compatible closure of the axioms:

\[(\beta) : (\lambda x. M)N = M[N/x]\]
\[(\mu) : (\mu a. M)N = \mu a. M[\alpha \leftarrow N]\]
\[(\text{ren}) : [\alpha] \mu \beta. C = C[\alpha / \beta]\]

We write \(M \equiv_{\beta \mu} N\) if the equation \(M = N\) is derivable in \(\lambda \mu\).

2 \(\lambda \mu\)-models and term interpretation

As for the \(\lambda\)-calculus, in order to provide a semantics to the untyped \(\lambda \mu\)-calculus we need to look for a domain \(D\) and a mapping \([\cdot]D\) such that \([M]D e \in D\) for each term \(M\), where \(e\) maps variables to terms and names to continuations. Since the interpretation of terms depends on the interpretation of names and commands, we need an auxiliary domain \(C\) and a mapping \([\cdot]C\) such that \(e \alpha \in C\) for any name \(\alpha\), and \([C]C e \in C\). The term interpretation is a model of the theory \(\lambda \mu\) if \([M]D = [N]D\) whenever \(M \equiv_{\beta \mu} N\).

The semantics we consider here is due to Streicher and Reus [49]. The idea is to work in the category \(N_R\) of ‘negated’ domains of the shape \(A \rightarrow R\), where \(R\) is a parameter for the
domain of results. In such a category, continuations are directly modeled and treated as the fundamental concept, providing a semantics both to Felleisen’s $\lambda C$-calculus and to a variant of $\lambda \mu$ that has, next to the two sorts of term we consider here (terms and commands) also continuation terms.

In this section we adapt that semantics to Parigot’s original $\lambda \mu$. We rephrase the model definition in the setting of the normal categories of domains, obtaining something similar to Hindley-Longo ‘syntactical models’. Our models are essentially a particular case of the definitions in [42, 34].

**Definition 2.1** ($\lambda \mu$-Model) A triple $\mathcal{M} = (R, D, C)$ is a $\lambda \mu$-model in a category of domains $\mathcal{D}$ if $R \in \mathcal{D}$ is a fixed domain of results and $D$ and $C$ (called domains of denotations and of continuations, respectively) are solutions in $\mathcal{D}$ of the equations:

\[
\begin{align*}
D &= C \rightarrow R \\
C &= D \times C
\end{align*}
\]

In the terminology of [49], elements of $D$ are denotations, while those of $C$ are continuations. We refer to the above equations as the continuation domain equations. We let $k$ range over $C$, and $d$ over $D$.

**Remark 2.2** If $(R, D, C)$ is a $\lambda \mu$-model then $C$ is (isomorphic to) the infinite product $D \times D \times D \times \cdots$. On the other hand, we also have:

\[
D \simeq C \rightarrow D \simeq (D \times C) \rightarrow D \simeq D \rightarrow (C \rightarrow D) \simeq D \rightarrow D.
\]

since categories of domains are cartesian closed. Therefore, a $\lambda \mu$-model as defined in Definition 2.1 is an extensional $\lambda$-model.

**Definition 2.3** (Term Interpretation) Let $\mathcal{M} = (R, D, C)$ be a $\lambda \mu$-model.

i) Let $\text{Env} = (\text{Var} \rightarrow D) \uplus (\text{Name} \rightarrow C)$, where $\uplus$ is disjoint union; we let $e$ range over $\text{Env}$ and call elements of $\text{Env}$ environments.

ii) We define an environment update as:

\[
e[x \mapsto d]y = \begin{cases} 
  d & (x = y) \\
  e & \text{(otherwise)}
\end{cases}
\]

\[
e[\alpha \mapsto k]\beta = \begin{cases} 
  k & (\alpha = \beta) \\
  e\beta & \text{(otherwise)}
\end{cases}
\]

iii) The interpretation mappings $\mathcal{D} : \text{Trm} \rightarrow \text{Env} \rightarrow D$ and $\mathcal{C} : \text{Cmd} \rightarrow \text{Env} \rightarrow C$, written $\mathcal{D}$ and $\mathcal{C}$ when $\mathcal{M}$ is understood, are mutually defined by the equations:

\[
\begin{align*}
[x] &\mathcal{D} e k = e \times k \\
[\lambda x.M] &\mathcal{D} e k = [M] \mathcal{D} e[x \mapsto d] k' \quad (k = \langle d, k' \rangle) \\
[MN] &\mathcal{D} e k = [M] \mathcal{D} e [\llbracket N \rrbracket \mathcal{D} e, k] \\
[\mu \alpha.C] &\mathcal{D} e k = d k' \quad (\langle d, k' \rangle = [C] \mathcal{C} e[\alpha \mapsto k]) \\
[\alpha] &\mathcal{C} e = [\llbracket M \rrbracket \mathcal{D} e, e\alpha]
\end{align*}
\]

This definition has a strong similarity with Bierman’s interpretation of $\lambda \mu$ [16]; however, he considers a typed version.

In the second equation of the definition of $\mathcal{D}$, the assumption $k = \langle d, k' \rangle$ is not restrictive: in particular, if $k = \bot_C = \langle \bot_D, \bot_C \rangle$, then $d = \bot_D$ and $k' = k = \bot_C$. Below, we fix a $\lambda \mu$-model $\mathcal{M}$, and we shall write $\mathcal{D}$ or simply $\mathcal{D}$ by omitting the superscripts $C$ and $D$ whenever clear from the context.
Lemma 2.4 For all $e \in \text{ENV}$:

i) If $x \not\in \text{fo}(M)$, then $\llbracket M \rrbracket e = \llbracket M \rrbracket e[x \mapsto d]$, for all $d \in D$.

ii) If $\alpha \not\in \text{fn}(M)$, then $\llbracket M \rrbracket e = \llbracket M \rrbracket e[\alpha \mapsto k]$, for all $k \in C$.

Proof: Easy.

We establish the following relation between the various kinds of substitution and the interpretation.

Lemma 2.5 For any $M, N \in \text{TrM}$, $e \in \text{ENV}$ and $k \in C$:

\[
\llbracket M[N/x] \rrbracket e k = \llbracket M \rrbracket e[x \mapsto \llbracket N \rrbracket e] k.
\]

\[
\llbracket M[\alpha/\beta] \rrbracket e k = \llbracket M \rrbracket e[\beta \mapsto e \alpha] k.
\]

\[
\llbracket M[\alpha \leftarrow N] \rrbracket e k = \llbracket M \rrbracket e[\alpha \mapsto \langle \llbracket N \rrbracket e, e \alpha \rangle] k.
\]

Proof: The first two parts follow by straightforward induction on the definition of substitution; the third is shown by induction on the definition of (structural) substitution. The only non-trivial case is when $M \equiv \mu \beta. [\alpha] L$ with $\beta \neq \alpha$, so that $(\mu \beta. [\alpha] L)[\alpha \leftarrow N] \equiv \mu \beta. [\alpha] L[\alpha \leftarrow N]$.

By unravelling definitions we have:

\[
\llbracket \mu \beta. [\alpha] L[\alpha \leftarrow N] \rrbracket e k = d\, k'
\]

where

\[
\langle d', k' \rangle = \langle \llbracket \alpha \rrbracket L[\alpha \leftarrow N] \rangle e[\beta \mapsto k] = \langle \llbracket L[\alpha \leftarrow N] \rrbracket e[\beta \mapsto k], e[\beta \mapsto k] \alpha \rangle,
\]

observing that $e[\beta \mapsto k] \alpha = e \alpha$, since $\beta \neq \alpha$. Then:

\[
\llbracket \mu \beta. [\alpha] L[\alpha \leftarrow N] \rrbracket e k = \llbracket L[\alpha \leftarrow N] \rrbracket e[\beta \mapsto k] (e \alpha)
\]

\[
= \llbracket L[\alpha \leftarrow N] \rrbracket e[\beta \mapsto k] \langle \llbracket N \rrbracket e[\beta \mapsto k], e \alpha \rangle
\]

\[
= \llbracket L \rrbracket e[\beta \mapsto k, \alpha \mapsto \langle \llbracket N \rrbracket e, e \alpha \rangle \langle \llbracket N \rrbracket e, e \alpha \rangle]
\]

where the last equation follows by induction and the fact that we can assume that $\beta \not\in \text{fo}(N)$, so that $\llbracket N \rrbracket e[\beta \mapsto k] = \llbracket N \rrbracket e$. Let $e' = e[\alpha \mapsto \langle \llbracket N \rrbracket e, e \alpha \rangle]$, then:

\[
\llbracket L \rrbracket e[\beta \mapsto k, \alpha \mapsto \langle \llbracket N \rrbracket e, e \alpha \rangle \langle \llbracket N \rrbracket e, e \alpha \rangle] \langle \llbracket N \rrbracket e, e \alpha \rangle = \llbracket L \rrbracket e'[\beta \mapsto k] \langle \llbracket N \rrbracket e, e \alpha \rangle
\]

and

\[
\langle \llbracket L \rrbracket e'[\beta \mapsto k], \langle \llbracket N \rrbracket e, e \alpha \rangle \rangle = \llbracket L \rrbracket e'[\beta \mapsto k], e' \alpha \rangle
\]

\[
= \llbracket L \rrbracket e'[\beta \mapsto k], e'[\beta \mapsto k] \alpha
\]

where $\mu \beta. [\alpha] L[\alpha \leftarrow N] e' k = \llbracket L \rrbracket e'[\beta \mapsto k] \langle \llbracket N \rrbracket e, e \alpha \rangle = \llbracket \mu \beta. [\alpha] L[\alpha \leftarrow N] \rrbracket e k$. Since Definition 2.3 does not coincide exactly with the one of Streicher and Reus, we have to check that it actually models $\lambda \mu$ convertibility. We begin by stating the key fact about the semantics, i.e. that it satisfies the following ‘swapping continuations’ equation:

Lemma 2.6 $\llbracket \mu \alpha. [\beta] M \rrbracket e k = \llbracket M \rrbracket e[\alpha \mapsto k] (e[\alpha \mapsto k] \beta)$.

Proof: Since $\llbracket \mu \alpha. [\beta] M \rrbracket e k = dk'$ where $\langle d, k' \rangle = \langle \llbracket \beta \rrbracket M \rangle e[\alpha \mapsto k] = \langle \llbracket M \rrbracket e[\alpha \mapsto k], e[\alpha \mapsto k] \beta \rangle$.

We are now in place to establish the soundness of the interpretation.

\[1\] The equation is from [49], where it is actually: $\llbracket \mu \alpha. [\beta] M \rrbracket e k = \llbracket M \rrbracket e[\alpha \mapsto k] (e \beta)$, but this is certainly just a typo.
Theorem 2.7 (Soundness with respect to $\lambda\mu$) Let $M,N \in \text{Trm}$. If $M =_{\beta\mu} N$ then $\llbracket M \rrbracket = \llbracket N \rrbracket$.

Proof: By induction on the definition of $=_{\beta\mu}$ it suffices to check the axioms ($\beta$), ($\mu$), and (ren):

$$(\lambda x.M)N = M[N/x] :: \llbracket (\lambda x.M)N \rrbracket e k \triangleq \llbracket \lambda x.M \rrbracket e (\llbracket N \rrbracket e, k)$$
$$= \llbracket \lambda x.M \rrbracket e[x\mapsto \llbracket N \rrbracket e] k \quad \text{(Lemma 2.5)}$$
$$= \llbracket M[N/x] \rrbracket e k$$

$$(\mu\alpha.[\beta]M)N = \mu\alpha.([\beta]M)[\alpha \leftarrow N] :: \text{notice that, by Barendregt's convention, we can assume that } \alpha \not\in \text{fn}(N); \text{ let } e' = e[\alpha \mapsto (\llbracket N \rrbracket e, k)], \text{ then:}$$
$$\llbracket (\mu\alpha.[\beta]M)N \rrbracket e k \triangleq \llbracket \mu\alpha.[\beta]M \rrbracket e (\llbracket N \rrbracket e, k)$$
$$= \llbracket M \rrbracket e' (e'\beta) \quad \text{(Lemma 2.6)}$$
$$= \llbracket M[\alpha \leftarrow N] \rrbracket e[\alpha \mapsto k] (e'\beta) \quad \text{(Lemma 2.5)}$$

Now if $\beta = \alpha$ we have:

$$\llbracket M[\alpha \leftarrow N] \rrbracket e[\alpha \mapsto k] (e'\beta) = \llbracket M[\alpha \leftarrow N] \rrbracket e[\alpha \mapsto k] (\llbracket N \rrbracket e, k) \beta = \alpha$$
$$= \llbracket M[\alpha \leftarrow N] \rrbracket e[\alpha \mapsto k] (\llbracket N \rrbracket e[\alpha \mapsto k], k) \quad \alpha \not\in \text{fn}(N)$$
$$= \llbracket M[\alpha \leftarrow N] \rrbracket e[\alpha \mapsto k] (e[\alpha \mapsto k] \alpha)$$
$$= \llbracket \mu\alpha.[\beta]M[\alpha \leftarrow N] \rrbracket e k \quad \text{(Lemma 2.6)}$$
$$\triangleq \llbracket \mu\alpha.(\alpha[\beta]M)[\alpha \leftarrow N] \rrbracket e k$$

Otherwise, if $\beta \neq \alpha$ we have:

$$\llbracket M[\alpha \leftarrow N] \rrbracket e[\alpha \mapsto k] (e'\beta) = \llbracket M[\alpha \leftarrow N] \rrbracket e[\alpha \mapsto k] (e[\alpha \mapsto k] \beta)$$
$$= \llbracket \mu\alpha.[\beta]M[\alpha \leftarrow N] \rrbracket e k \quad \text{(Lemma 2.6)}$$
$$\triangleq \llbracket \mu\alpha.([\beta]M)[\alpha \leftarrow N] \rrbracket e k$$

$$(\mu\psi.[\alpha]\mu\beta.[\gamma]M = \mu\psi.([\gamma]M)[\alpha/\beta]) ::$$

$$\llbracket \mu\psi.[\alpha]\mu\beta.[\gamma]M \rrbracket e k = \llbracket \mu\beta.[\gamma]M \rrbracket e[\psi \mapsto k] e[\psi \mapsto k] \alpha$$
$$= \llbracket M \rrbracket e[\psi \mapsto k] [\beta \mapsto e[\psi \mapsto k] \alpha] (e[\psi \mapsto k] [\beta \mapsto e[\psi \mapsto k] \alpha] \gamma)$$
$$= \llbracket M[\alpha/\beta] \rrbracket e[\psi \mapsto k] (e[\psi \mapsto k] \gamma)$$
$$\triangleq \llbracket \mu\psi.[\gamma]M[\alpha/\beta] \rrbracket e k$$

where, in case $\gamma = \beta$, the second equation holds because $e[\psi \mapsto k] [\beta \mapsto (e[\psi \mapsto k] \alpha)] \beta = e[\psi \mapsto k] \alpha$.

3 The filter domain

In this section we build a $\lambda\mu$-model in the category of $\omega$-algebraic lattices. The model is first obtained in §3.1 by means of standard domain theoretic techniques and following the construction in [49]; then we exploit the fact that compact points of any $\omega$-algebraic lattice can be described by means of a suitable intersection type theory, which is recalled in §3.2, to get a description of the model as a filter-model in §3.3. This provides us with a semantically justified definition of intersection types, actually of three kinds to describe the domains $R,D$ and $C$ respectively that form the model, and of their preorders that we shall use in section 4 for the type assignment system to $\lambda\mu$ terms and commands.

The treatment of §3.1 is introductory and can be skipped by readers who are familiar with domain theory, but for propositions 3.1 and 3.2, which are referred to in the subsequent parts of the paper. A fuller treatment of these topics can be found e.g. in [3], chapters 1-3 and 7. The developments in §3.2 and §3.3 are inspired to [15, 19] and [1]; in particular we have used [24] in §3.3; we borrow the terminology of “intersection type theory” from [33], where intersection type systems and filter models are treated in full detail in part III.
3.1 A domain theoretic solution of continuation domain equations

Complete lattices are partial orders \((X, \sqsubseteq)\) closed under meet \(\sqcap Z\) (greatest lower bound) and join \(\sqcup Z\) (smallest upper bound) of arbitrary subsets \(Z \subseteq X\). Noting that \(\sqcap Z = \bigsqcap\{x \in X \mid \forall z \in Z, x \sqsubseteq z\}\) and \(\sqcup Z = \bigsqcup\{x \in X \mid \forall z \in Z, z \sqsubseteq x\}\), we have that if \(X\) is closed under arbitrary meets (joins) it is such under arbitrary joins (meets). Further in \(X\) there exist \(\bot = \bigsqcup\emptyset\) and \(\top = \bigsqcap\emptyset\), which are the top and bottom elements of \(X\) respectively.

A subset \(Z \subseteq X\) is directed if for any finite subset \(V \subseteq Z\) there exists \(z \in Z\) which is an upper bound of \(V\). In particular directed subsets are always non-empty. An element \(e \in X\) is compact if, whenever \(e \subseteq \sqcup Z\) for some directed \(Z \subseteq X\), there exists \(z \in Z\) such that \(e \sqsubseteq z\); we write \(K(X)\) for the set of compact elements of \(X\). For \(x \in X\) let us write \(\mathcal{K}(x) = \{e \in K(X) \mid e \sqsubseteq x\}\); since directed sets are non-empty, \(\bot \in \mathcal{K}(X)\) and hence \(\bot \in \mathcal{K}(x)\) for all \(x \in X\). A complete lattice \(X\) is algebraic if any \(x \in X\) is such that \(\mathcal{K}(x)\) is directed and \(x = \sqcup \mathcal{K}(x)\); \(X\) is \(\omega\)-algebraic if it is algebraic and the subset \(\mathcal{K}(X)\) is countable.

A function \(f : X \to Y\) of \(\omega\)-algebraic lattices is Scott-continuous if and only if it preserves directed sups, namely \(f(\sqcup Z) = \sqcup_{z \in Z} f(z)\) whenever \(Z \subseteq X\) is directed. By algebraicity any continuous function \(f\) with domain \(X\) is fully determined by its restriction to \(\mathcal{K}(X)\), that is, given a monotonic function \(g : \mathcal{K}(X) \to Y\) there exists a unique continuous function \(\hat{g} : X \to Y\) that coincides with \(g\) over \(\mathcal{K}(X)\) namely \(\hat{g}(x) = \sqcup g(\mathcal{K}(x))\); \(\hat{g}\) is called the continuous extension of \(g\). The category \(\omega\text{-Alg}\) has \(\omega\)-algebraic lattices as objects and Scott-continuous maps as morphisms. As such \(\omega\text{-Alg}\) is a full subcategory of the category of domains, but not of the category of (complete) lattices, since morphisms do not preserve arbitrary joins. In this paper we use the word domain as synonym of \(\omega\)-algebraic lattice.

If \(X, Y\) are domains, then the cartesian product \(X \times Y\) ordered component-wise and the set \([X \to Y]\) of Scott-continuous functions from \(X\) to \(Y\) ordered point-wise, are both domains; in particular if \(f, g \in [X \to Y]\) then the function \((f \sqcup g)(x) = f(x) \sqcup g(x)\) is the join of \(f\) and \(g\), that are always compatible, namely they have an upper bound. If \(Z\) is an \(\omega\)-algebraic lattice then \([X \times Y \to Z] \simeq [X \to [Y \to Z]]\) is a natural isomorphism, therefore the category \(\omega\text{-Alg}\) is cartesian closed. The set compact points of \(X \times Y\) is \(\mathcal{K}(X \times Y) = \mathcal{K}(X) \times \mathcal{K}(Y)\); the set of compact points of \([X \to Y]\) namely \(\mathcal{K}[X \to Y]\), is the set of finite joins of step functions \((a \Rightarrow b)\) where \(a \in \mathcal{K}(X), b \in \mathcal{K}(Y)\) and

\[
(a \Rightarrow b)(x) = \begin{cases} b & \text{if } a \sqsubseteq x, \\ \bot & \text{otherwise.} \end{cases}
\]

An infinite sequence \((X_n)_{n \in \mathbb{N}}\) of domains is projective if for all \(n\) there exist the continuous functions \(e_n : X_n \to X_{n+1}\) and \(p_n : X_{n+1} \to X_n\), called embedding-projection pairs, such that \(p_n \circ e_n = \text{id}_X\) and \(e_n \circ p_n \leq \text{id}_X\), where \(\leq\) is point wise ordering. The inverse limit of the projective chain \((X_n)_{n \in \mathbb{N}}\) is the set \(X_\infty = \lim_\leftarrow X_n\) which is defined as the set of all vectors \(\vec{x} \in \Pi_n X_n\) such that \(x_i = p_i(x_{i+1})\) for all \(i\), ordered component wise. Moreover for all \(n\) there exists an embedding-projection pair \(e_{n,\infty} : X_n \to X_\infty\) and \(p_{n,\infty} : X_\infty \to X_n\) such that \(p_{n,\infty}(\vec{x}) = x_n\) for all \(\vec{x} \in X_\infty\); for the details see e.g. [3] chapter 7.

**Property 3.1** The inverse limit \(X_\infty = \lim_\leftarrow X_n\) of a sequence \((X_n)_{n \in \mathbb{N}}\) of domains is itself a domain such that

\[
\mathcal{K}(X_\infty) = \bigcup_n \{e_{n,\infty}(x) \mid x \in \mathcal{K}(X_n)\}.
\]

**Proof:** That \(X_\infty\) is a domain follows by the fact that \(X_n\) is such for all \(n\), and that for all \(x \in X_n\) there exists \(\vec{x} = e_{n,\infty}(x)\) such that \(x_n = p_{n,\infty}(\vec{x}) = x\).

Let us write \(\bigcup_n e_{n,\infty}\mathcal{K}(X_n)\) for the righthand side of the above equation, and consider a directed subset \(Z \subseteq X_\infty\). For any \(n\), If \(x \in \mathcal{K}(X_n)\) then \(e_{n,\infty}(x) \sqsubseteq \sqcup Z\) implies

\[
x = p_{n,\infty} \circ e_{n,\infty}(x) \sqsubseteq p_{n,\infty}(\sqcup Z) = \sqcup p_{n,\infty}(Z)
\]
by the fact that \( (e_{n,\infty}, p_{n,\infty}) \) are an embedding-projection pair and the continuity of \( p_{n,\infty} \). By assumption there exists \( z \in Z \) s.t. \( x \subseteq p_{n,\infty}(z) \), and therefore \( e_{n,\infty}(x) \subseteq (e_{n,\infty} \circ p_{n,\infty})(z) = z \in Z \), hence \( e_{n,\infty}(x) \in K(X_{\infty}) \), by the arbitrary choice of \( Z \). This proves \( K(X_{\infty}) \supseteq \bigcup_n e_{n,\infty}K(X_n) \).

To see the converse inclusion let \( \vec{x} \in K(X_{\infty}) \). For any \( n \) we claim that \( x_n = p_{n,\infty}(\vec{x}) \in K(X_n) \).

Indeed if \( U \subseteq X_n \) is directed and such that \( x_n \subseteq \square U \) then consider the set \( V = \{ \vec{y} \in X_\infty \mid \forall m \neq n. \ y_m = x_m \ & \ \exists u \in U. \ y_u = u \} \); it follows that \( V \) is directed and \( \vec{x} \subseteq \square V \). By the hypothesis \( \vec{x} \in K(X_{\infty}) \) there exists \( \vec{y} \in V \) such that \( \vec{x} \subseteq \vec{y} \), and so there exists \( u = y_n \in U \) s.t. \( x_n \subseteq u \), establishing the claim. From this and the first part of this proof it follows that \( \{ e_{n,\infty}(p_{n,\infty}(\vec{x})) \}_{n \in N} = \{ e_{n,\infty}(x_n) \}_{n \in N} \subseteq \bigcup_n e_{n,\infty}K(X_n) \) is a chain of elements in \( K(X_{\infty}) \), and by construction \( \vec{x} = \bigcup_n e_{n,\infty}(x_n) \); since \( \vec{x} \in K(X_{\infty}) \) we conclude that \( \vec{x} = e_{n,\infty}(x_n) \) for some \( n_0 \), so that \( K(X_{\infty}) \subseteq \bigcup_n e_{n,\infty}K(X_n) \) as desired.

We now consider the construction in [49] in the particular case of \( \omega \cdot \text{AlgL} \). Let \( R \) be some fixed domain, dubbed the domain of \textit{results} (for the sake of solving the continuation domain equations in a non trivial way it suffices to take \( R = \{ \bot \cup \top \} \), the two points lattice). Now define the following sequences of domains:

\[
C_0 = \{ \bot \},
D_n = [C_n \rightarrow R],
C_{n+1} = D_n \times C_n
\]

where \( \{ \bot \} \) is the trivial lattice such that \( \bot = \top \). Observe that \( D_0 = [C_0 \rightarrow R] \simeq R \) and \( D_0 \simeq D_0 \times \{ \bot \} = C_1 \) and so \( D_1 = [C_1 \rightarrow R] \simeq [R \rightarrow R] \). By unravelling the definition of \( C_n \) and \( D_n \) we obtain:

\[
C_n = [C_{n-1} \rightarrow R] \times [C_{n-2} \rightarrow R] \times \cdots \times [C_0 \rightarrow R] \times C_0.
\]

In [49], Theorem 3.1, it is proved that these sequences are projective, so that \( D = \lim_\leftarrow D_n \) and \( C = \lim_\leftarrow C_n \) are the initial/final solution of the continuation domain equations such that \( R \simeq D_0 \). By Proposition 3.1 we know that, up to the embeddings of each \( D_0 \) into \( D \) and of each \( C_n \) into \( C \), the compact points of \( D \) and \( C \) are the union of the compacts of the \( D_n \) and \( C_n \) respectively:

\[
\mathcal{K}(D) = \bigcup_n \mathcal{K}(D_n), \quad \mathcal{K}(C) = \bigcup_n \mathcal{K}(C_n).
\] (1)

In particular \( \mathcal{K}(R) = \mathcal{K}(D_0) \subseteq \mathcal{K}(D) \). Since \( C \simeq D \times C \), it can be seen as the infinite product \( \Pi_n D = D \times D \cdots \); however \( \mathcal{K}(C) \) is a proper subset of the product \( \Pi_n \mathcal{K}(D) = \mathcal{K}(D) \times \mathcal{K}(D) \times \cdots \).

**Property 3.2** The compact points in \( C = \lim_\leftarrow C_n \) are those infinite tuples in \( \Pi_n \mathcal{K}(D) \) whose components are all equal to \( \bot \) but for a finite number of cases:

\[
\mathcal{K}(C) = \{ \langle d_1, d_2, \ldots \rangle \in \Pi_n \mathcal{K}(D) \mid \exists i \forall j \geq i. \ d_j = \bot \}.
\]

**Proof:** Let \( K = \{ \langle d_1, d_2, \ldots \rangle \in \Pi_n \mathcal{K}(D) \mid \exists i \forall j \geq i. \ d_j = \bot \} \). Since \( C = \Pi_n D \) is ordered point-wise, we immediately know that \( K \subseteq \mathcal{K}(C) \), so that it suffices to show the inverse inclusion.

Let \( k = \langle d_1, d_2, \ldots \rangle \in \mathcal{K}(C) \); and, toward a contradiction, suppose that there are infinitely many components \( d_i \) of \( k \) that are different than \( \bot \). Set \( k_j = \langle d_1, d_2, \ldots, d_j, \bot, \ldots \rangle \in K \) as the tuple which is definitely equal to \( \bot \) after \( d_j \), while previous components are the same as in \( k \). Then the set \( \{ k_j \mid j \in \mathbb{N} \} \) is directed (actually a chain) and \( k = \bigcup_j k_j \); but \( k \not\subseteq k_j \) for any \( j \) hence \( k \notin \mathcal{K}(C) \), a contradiction.

Another way to see what is stated in the last proposition is to observe that the \( C_n \) are finite products of the shape \( [C_{n-1} \rightarrow R] \times [C_{n-2} \rightarrow R] \times \cdots \times [C_0 \rightarrow R] \times C_0 \), where \( C_0 = \{ \bot \} \), hence any tuple in \( C_n \), and therefore in \( \mathcal{K}(C_n) \), has the form \( \langle d_1, \ldots, d_{n-2}, \bot \rangle \). Now the embedding of such a tuple into \( C = \lim_\leftarrow C_n \) is the infinite tuple \( \langle d_1, \ldots, d_{n-2}, \bot, \ldots, \bot, \ldots \rangle \) that is definitely \( \bot \) after \( d_{n-2} \), and we know that the images of compact points in the \( C_n \)'s are exactly the elements of \( \mathcal{K}(C) \).
3.2 Intersection type theories and the filter construction

Intersection types form the “domain logic” of \( \omega \)-algebraic lattices in the sense of [1]. This means that for each domain \( X \) in \( \omega \text{-AlgL} \) there exists a countable language \( \mathcal{L}_X \) of intersection types together with an appropriate preorder \( \leq_X \) such that \( (\mathcal{L}_X, \leq_X) \) is the Lindenbaum algebra of the compact points \( \mathcal{K}(X) \) of \( X \), namely an axiomatic, and hence finitary presentation of the structure \( \mathcal{K}^{op}(X) = (\mathcal{K}(X), \subseteq^{op}) \) where \( \subseteq^{op} \) is just the inverse of the partial order \( \sqsubseteq \) of \( X \).

To see this one first observes that \( \mathcal{K}(X) \) is closed under binary joins. Indeed for any \( e_1, e_2 \in \mathcal{K}(X) \) if \( e_1 \sqcup e_2 = \bigcup \{ e_1, e_2 \} \subseteq \bigcup Z \) for some directed \( Z \subseteq X \) then \( e_1, e_2 \subseteq \bigcup Z \), which implies that there exist \( z_1, z_2 \in Z \) such that \( e_i \subseteq z_i \) for \( i = 1,2 \). By directness of \( Z \) there is some \( z_3 \in Z \) such that \( z_1, z_2 \subseteq z_3 \), hence \( e_1 \sqcup e_2 \subseteq z_1 \sqcup z_2 \subseteq z_3 \). It follows that the structure \( (\mathcal{K}(X), \subseteq) \) is a sup-semilattice (a poset closed under finite joins), so that its dual \( \mathcal{K}^{op}(X) \) is an inf-semilattice (a poset closed under finite meets), whose meet operator \( \sqcap^{op} \) coincides with the join \( \sqcup \) over \( \mathcal{K}(X) \).

By algebraicity \( X \) is generated by \( \mathcal{K}(X) \) in the sense that \( (X, \subseteq) \) is isomorphic to the poset \( (\text{Idl}(\mathcal{K}(X)), \subseteq) \), where \( \text{Idl}(\mathcal{K}(X)) \), the set of ideals over \( \mathcal{K}(X) \), consists of directed and downward closed subsets of \( \mathcal{K}(X) \). It turns out that the compact elements of \( \text{Idl}(\mathcal{K}(X)) \) are just the images \( \downarrow e = \mathcal{K}(e) \) of the elements \( e \in \mathcal{K}(X) \).

Dually \( (X, \subseteq) \) is isomorphic to the poset \( (\text{Filt}(\mathcal{K}^{op}(X)), \subseteq) \), where \( \text{Filt}(\mathcal{K}^{op}(X)) \) is the set of filters over \( \mathcal{K}(X) \), that are non empty subsets of \( \mathcal{K}(X) \) which are upward closed w.r.t. \( \subseteq^{op} \) and closed under \( \sqcap^{op} \). Therefore filters over \( \mathcal{K}^{op}(X) \) give rise to the algebraic lattice \( (\text{Filt}(\mathcal{K}^{op}(X)), \subseteq) \), whose compact elements are \( \uparrow^{op} e = \{ e' \in \mathcal{K}(X) \mid e \sqsubseteq^{op} e' \} \), called principal filters. In summary we have the isomorphisms in the category \( \omega \text{-AlgL} \):

\[
X \simeq \text{Idl}(\mathcal{K}(X)) \simeq \text{Filt}(\mathcal{K}^{op}(X)).
\]

The fact that \( \mathcal{K}(X) \) is a countable set allows a finitary, that is syntactic presentation of \( X \simeq \text{Filt}(\mathcal{K}^{op}(X)) \) itself by introducing a language of types denoting the elements of \( \mathcal{K}(X) \) and axioms and rules defining a pre-order over types whose intended meaning is \( \subseteq^{op} \).

An intersection type language \( \mathcal{L} \) is a set of expressions closed under the binary operation \( \land \) and including the constant \( \omega \). Over this set it is defined a pre-order \( \leq \), making \( \land \) into the meet and \( \omega \) into the top element, as formally stated in the next definition.

**Definition 3.3 (Intersection Type Language and Theory)**

i) A denumerable set of type expressions \( \mathcal{L} \) is called an intersection type language if there is a constant \( \omega \in \mathcal{L} \) and \( \mathcal{L} \) is closed under the binary operator \( \sigma \land \tau \), called type intersection.

ii) An intersection type theory \( \mathcal{T} \) over \( \mathcal{L} \) is any axiomatic presentation of a pre-order \( \leq_{\mathcal{T}} \) over intersection types in \( \mathcal{L} \) validating the following axioms and rules:

\[
\begin{align*}
\sigma \land \tau & \leq \sigma \quad \sigma \land \tau \leq \tau \\
\sigma \leq \omega \\
\rho \leq \sigma \quad \rho \leq \tau \\
\rho \leq \sigma \land \tau
\end{align*}
\]

iii) We abbreviate \( \sigma \leq_{\mathcal{T}} \tau \leq_{\mathcal{T}} \sigma \) by \( \sigma \sim_{\mathcal{T}} \tau \) and write \( [\sigma]_{\mathcal{T}} \) for the equivalence class of \( \sigma \) with respect to \( \sim_{\mathcal{T}} \). The subscript \( \mathcal{T} \) will be omitted when no ambiguity is possible.

The type \( \sigma \land \tau \) is called “intersection type” in the literature. The reason is that as a type of \( \lambda \)-terms it is interpreted as the intersection of the interpretations of \( \sigma \) and \( \tau \) in set theoretic models of the \( \lambda \)-calculus. This is unfortunate in the present section, where we shall speak of filters and of their intersections. To avoid confusion we speak of type intersections when we refer to expressions of the shape \( \sigma \land \tau \), reserving the word ‘intersection’ to the set theoretic operation.
Given an intersection type theory $\mathcal{T}$ over a language $\mathcal{L}$ axiomatising the pre-order $\leq_T$, the quotient $\mathcal{L}/\leq_T$ is an inf-semilattice. We then establish a sufficient condition for $\leq_T$ being isomorphic to $K^{\text{op}}(X)$ for some $X \in \omega\text{-Alg}\mathcal{L}$.

Lemma 3.4 Let $\mathcal{T}$ be an intersection type theory over $\mathcal{L}$ and $\leq_T$ the relative pre-order. Let $(X,\sqsubseteq)$ be a domain and $\Theta : \mathcal{L} \to \mathcal{K}(X)$ an order reversing surjective mapping, that is such that for all $\sigma, \tau \in \mathcal{L}$:

$$\sigma \leq_T \tau \Rightarrow \Theta(\tau) \sqsubseteq \Theta(\sigma).$$

Then $\mathcal{L}/\leq_T \simeq K^{\text{op}}(X)$ as inf-semilattices.

Proof: Let $\Theta' : \mathcal{L}/\leq_T \to K^{\text{op}}(X)$ be defined by $\Theta'([\sigma]) = \Theta(\sigma)$. If $[\sigma] = [\tau]$, then $\sigma \sim_T \tau$, so by assumption $\Theta(\sigma) \sqsubseteq \Theta(\tau) \sqsubseteq \Theta(\sigma)$, which implies $\Theta(\sigma) = \Theta(\tau)$. This implies that $\Theta'$ is well defined and that $\Theta$ preserves and reflects $\leq_T$ with respect to $\sqsubseteq^{\text{op}}$ and that $\Theta'$ is a bijection, since $\Theta$ is surjective. Finally, $\Theta'([\sigma \wedge \tau]) = \Theta(\sigma \wedge \tau) = \Theta(\sigma) \sqcup \Theta(\tau) = \Theta(\sigma) \sqcap \Theta(\tau)$. In particular $\Theta'([\omega]) = \Theta(\omega) = \bot$.

Under the hypotheses of the last lemma we have that $X \simeq \text{Filt}(K^{\text{op}}(X)) \simeq \text{Filt}(\mathcal{L}/\leq_T)$; nonetheless we consider the more handy isomorphism of $X$ with the set of filters over the pre-order $(\mathcal{L},\leq_T)$, which we call formal filters.

Definition 3.5 (Formal Filters)  

i) A formal filter with respect to an intersection type theory $\mathcal{T}$ over $\mathcal{L}$ is a subset $f \subseteq \mathcal{L}$ such that:

$$\omega \in f \quad \sigma \in f \quad \sigma \leq_T \tau \quad \tau \in f$$

ii) We write $\mathcal{F}(\mathcal{T})$ for the set of formal filters induced by the theory.

iii) The filter $\uparrow_T \sigma = \{ \tau \in \mathcal{L} \mid \sigma \leq_T \tau \}$ is called principal and we write $\mathcal{F}_p(\mathcal{T})$ for the set of principal filters.

Before going on we recall some properties of formal filters and of the poset $(\mathcal{F}(\mathcal{T}),\subseteq)$. Since these are easily established or well known from the literature, we just state them or provide short arguments. The following is a list of some useful facts that follow immediately by definition.

Fact 3.6 Let $\sigma, \tau \in \mathcal{L}$ and $f \in \mathcal{F}(\mathcal{T})$, then:

i) $\sigma \leq_T \tau$ if and only if $\uparrow_T \tau \subseteq \uparrow_T \sigma$.

ii) $\uparrow_T \sigma \sqcup \uparrow_T \tau = \uparrow_T \sigma \wedge \tau$.

iii) If $\sigma \in f$, then $\uparrow_T \sigma \subseteq f$.

iv) $f = \bigcup_{\sigma \in f} \uparrow_T \sigma$.

v) For any $\mathcal{G} \subseteq \mathcal{F}(\mathcal{T})$, $\bigcap \mathcal{G} \in \mathcal{F}(\mathcal{T})$.

From Fact 3.6((v)) it follows that the poset $(\mathcal{F}(\mathcal{T}),\subseteq)$ is a complete lattice, with set theoretic intersection as (arbitrary) meet. However if $\mathcal{G} \subseteq \mathcal{F}(\mathcal{T})$ then the join $\bigvee \mathcal{G} = \bigcap\{ f \in \mathcal{F}(\mathcal{T}) \mid \forall g \in \mathcal{G}, g \subseteq f \}$ includes $\bigvee \mathcal{G}$ but does not coincide with it in general, since $\bigvee \mathcal{G}$ is not necessarily closed under $\wedge$. However, since $\bigvee \mathcal{G}$ is upper closed w.r.t. $\leq_T$, to get an explicit characterization of $\bigvee \mathcal{G}$ it is enough to close $\bigvee \mathcal{G}$ under finite type intersections:

$$\bigvee \mathcal{G} = \{ \sigma \mid \exists n, c_1, \ldots, c_n \forall i \leq n [c_i \in \bigcup \mathcal{G}] \& c_\sim_T c_1 \wedge \cdots \wedge c_n \}.$$

On the other hand if $\mathcal{G}$ is directed w.r.t. $\subseteq$ then $\bigvee \mathcal{G} = \bigcup \mathcal{G}$. In fact if $c_1, \ldots, c_n \in \bigcup \mathcal{G}$ we have that $\uparrow_T c_i \subseteq g_i$ by Fact 3.6((iii), for certain $g_1, \ldots, g_n \in \mathcal{G}$. By directness there exists $g' \in \mathcal{G}$ such that...
such that \( g_1 \cup \cdots \cup g_n \subseteq g' \), and hence the same holds for \( \uparrow \mathcal{T} \sigma_1 \cup \cdots \cup \uparrow \mathcal{T} \sigma_n \). It follows that 
\( \sigma_1, \ldots, \sigma_n \in g' \) and so \( \sigma_1 \land \cdots \land \sigma_n \in g' \subseteq \bigcup \mathcal{G} \), since \( g' \) is a formal filter.

The next lemma is not referenced explicitly in the paper, but it is used thoroughly in the rest of this section; it is folklore in the theory of filter \( \lambda \)-models.

**Lemma 3.7** The poset \(( \mathcal{F}(\mathcal{T}), \subseteq \)\) is an \( \omega \)-algebraic lattice with top \( \uparrow \mathcal{T} \omega \) and compacts \( \mathcal{K}(\mathcal{F}(\mathcal{T})) = \mathcal{F}_p(\mathcal{T}) \).

**Proof:** From the discussion above we know that \(( \mathcal{F}(\mathcal{T}), \subseteq \)\) is a complete lattice, so it remains to see that it is \( \omega \)-algebraic.

Let \( \uparrow \mathcal{T} \sigma \subseteq \bigcup \mathcal{G} \) for some directed \( \mathcal{G} \subseteq \mathcal{F}(\mathcal{T}) \). Then \( \bigcup \mathcal{G} = \bigcup \mathcal{G} \) so that \( \sigma \in \uparrow \mathcal{T} \sigma \subseteq \bigcup \mathcal{G} \).

Therefore there exists \( g \in \mathcal{G} \) such that \( \sigma \in g \) which implies \( \uparrow \mathcal{T} \sigma \subseteq g \) by 3.6((iii)). Hence \( \mathcal{F}_p(\mathcal{T}) \subseteq \mathcal{K}(\mathcal{F}(\mathcal{T})) \).

By 3.6((iv)) we have \( f = \bigcup_{\sigma \in f} \uparrow \mathcal{T} \sigma \) for any \( f \in \mathcal{F}(\mathcal{T}) \). Let \( \uparrow \mathcal{T} \sigma_i \subseteq f \) for certain \( \sigma_1, \ldots, \sigma_n \in f \);

using 3.6((ii)) repeatedly we have that \( \uparrow \mathcal{T} \sigma_1 \cup \cdots \cup \uparrow \mathcal{T} \sigma_n = \uparrow \mathcal{T}(\sigma_1 \land \cdots \land \sigma_n) \).

On the other hand \( \sigma_1 \land \cdots \land \sigma_n \in f \) since \( f \) is a formal filter, and \( \sigma_1 \land \cdots \land \sigma_n \leq \mathcal{T} \sigma_i \) for all \( i = 1, \ldots, n \), which by 3.6((i)) implies that \( \uparrow \mathcal{T} \sigma_i \subseteq \uparrow \mathcal{T}(\sigma_1 \land \cdots \land \sigma_n) \), namely that \( \{ \uparrow \mathcal{T} \sigma \mid \sigma \in f \} \) is directed.

Now if \( f \in \mathcal{K}(\mathcal{F}(\mathcal{T})) \) then \( f \subseteq \uparrow \mathcal{T} \sigma \) for some \( \sigma \in f \); by 3.6((iii)) we conclude that \( f = \uparrow \mathcal{T} \sigma \).

Therefore \( \mathcal{K}(\mathcal{F}(\mathcal{T})) \subseteq \mathcal{F}_p(\mathcal{T}) \) and hence they are equal by the above. By this and 3.6((iv)) we finally conclude that \(( \mathcal{F}(\mathcal{T}), \subseteq \)\) is algebraic, and in fact it is \( \omega \)-algebraic because \( \mathcal{L} \) is countable and the map \( \sigma \mapsto \uparrow \mathcal{T} \sigma \) from \( \mathcal{L} \) to \( \mathcal{F}_p(\mathcal{T}) \) is obviously onto.

**Property 3.8** Let \( \mathcal{T} \) be an intersection type theory over \( \mathcal{L} \), and let \( X \) and \( \Theta : \mathcal{L} \to \mathcal{K}(\mathcal{X}) \) be a domain and a mapping satisfying the hypotheses of Lemma 3.4. Then \( \mathcal{F}(\mathcal{T}) \simeq \text{Filt}(\mathcal{K}^{\text{op}}(\mathcal{X})) \simeq X \).

**Proof:** If \( f \in \mathcal{F}(\mathcal{T}) \) then by Lemma 3.6, \( \{ \sigma \mid \sigma \in f \} \) is a filter over \( \mathcal{L}/\leq \mathcal{T} \); vice versa, if \( f \) is a filter over \( \mathcal{L}/\leq \mathcal{T} \) then \( \bigcup f = \{ \sigma \mid \sigma \in f \} \) is a formal filter; therefore \( \mathcal{F}(\mathcal{T}) \simeq \text{Filt}(\mathcal{L}/\leq \mathcal{T}) \).

On the other hand, by Lemma 3.4 and the hypothesis, we have \( \mathcal{L}/\leq \mathcal{T} \simeq \mathcal{K}^{\text{op}}(\mathcal{X}) \) via the mapping \( \Theta'([\sigma]) = \Theta(\sigma) \), so that the desired isomorphism \( \mathcal{F}(\mathcal{T}) \simeq \text{Filt}(\mathcal{K}^{\text{op}}(\mathcal{X})) \) is given by

\[
\lambda f. \bigcup \{ \Theta'([\sigma]) \mid \sigma \in f \} = \lambda f. \bigcup \{ \Theta(\sigma) \mid \sigma \in f \}.
\]

Because of Proposition 3.8 and essentially following [15], we ignore the distinction between formal filters over a pre-order and filters over the ordered quotient and we shall work with the simpler formal filters, henceforth just called filters.

### 3.3 A filter domain solution to the continuation domain equations

Given an arbitrary domain \( \mathcal{R} \) as in Definition 2.1, we fix the initial and final solution \( D, C \) of the continuation equations in the category \( \omega\text{-AlgL} \). Then, for \( \mathcal{A} = R, D, C \), we define the languages \( \mathcal{L}_\mathcal{A} \) and the theories \( \mathcal{T}_\mathcal{A} \), inducing the preorders \( \leq \mathcal{T}_\mathcal{A} \) written shortly as \( \leq \mathcal{A} \).

**Definition 3.9** Let \( \mathcal{R} \in \omega\text{-AlgL} \), ordered by \( \subseteq \mathcal{R} \), with bottom \( \bot \) and join \( \sqcup \), then:

i) The intersection type language \( \mathcal{L}_\mathcal{R} \) is defined by the grammar:

\[
\rho ::= v_a \mid \omega \mid \rho \land \rho \quad (a \in \mathcal{K}(\mathcal{R})),
\]

ii) \( \mathcal{T}_\mathcal{R} \) is the smallest intersection type theory axiomatising the preorder \( \leq \mathcal{R} \) such that, writing \( \sim \mathcal{R} \) for \( \leq \mathcal{R} \cap \leq^{\text{op}}_\mathcal{R} \):

\[
\begin{align*}
v_\bot \sim \mathcal{R} \omega & \quad v_a \cup b \sim \mathcal{R} v_a \land v_b
\end{align*}
\]
iii) The mapping \( \Theta_R : \mathcal{L}_R \rightarrow \mathcal{K}(R) \) is defined by:
\[
\Theta_R(v_a) = a \\
\Theta_R(\omega) = \bot \\
\Theta_R(\rho_1 \land \rho_2) = \Theta_R(\rho_1) \sqcup \Theta_R(\rho_2)
\]

Observe that \( \land \) is the meet with respect to \( \leq_R \).

The following property, that states the relation between \( \sqsubseteq_R \) and \( \leq_R \), holds naturally:

**Lemma 3.10**

i) \( v_a \leq_R v_b \iff b \sqsubseteq_R a \).

ii) \( \rho \leq_R \rho' \iff \Theta_R(\rho') \sqsubseteq_R \Theta_R(\rho) \).

**Proof:** In the following we remove all the subscripts \( R \) for notational simplicity.

i) If \( v_a \leq v_b \) then either \( b = \bot \), so that \( \bot \sqsubseteq a \), or \( v_a \sim v_a \land v_b \), since \( \land \) is the meet with respect to \( \leq \), which is an intersection type theory. By definition we have \( v_a \land v_b \sim v_a \land v_b \), so it follows that \( v_a \sim v_a \land v_b \), which implies \( a = a \sqcup b \), so \( b \sqsubseteq a \).

Vice versa, if \( b \sqsubseteq a \) then \( a = a \sqcup b \) and we have \( v_a = v_a \sqcup b \sim v_a \land v_b \) from which we conclude \( v_a \leq v_b \).

ii) When we, consistently with the theory \( \leq \), identify \( v_\bot \) and \( \omega \), then \( \rho = v_{a_1} \land \cdots \land v_{a_h} \), for some \( a_1, \ldots, a_h \in \mathcal{K}(R) \); notice that \( v_{a_1} \land \cdots \land v_{a_h} \sim v_{a_1 \sqcup \cdots \sqcup a_h} \). Likewise, \( \rho' = v_{b_1} \land \cdots \land v_{b_k} \), then by part (i) we have:
\[
\rho \sim v_{a_1 \sqcup \cdots \sqcup a_h} \leq v_{b_1 \sqcup \cdots \sqcup b_k} \sim \rho' \iff b_1 \sqcup \cdots \sqcup b_k \sqsubseteq a_1 \sqcup \cdots \sqcup a_h
\]

Now the result follows by observing that \( \Theta_R(\rho') = b_1 \sqcup \cdots \sqcup b_k \) and \( \Theta_R(\rho) = a_1 \sqcup \cdots \sqcup a_h \).

The following corollary is then the converse of Proposition 3.8.

**Corollary 3.11** There exists an intersection type theory \( \mathcal{T}_R \) such that \( \mathcal{F}_R \simeq R \).

**Proof:** Let \( \mathcal{T}_R \) and \( \Theta_R \) be defined as in Definition 3.9. Now \( \Theta_R \) is surjective since \( \Theta_R(v_a) = a \) for all \( a \in \mathcal{K}(R) \) and, by Lemma 3.10(ii), it satisfies the hypotheses of Lemma 3.4, so that we conclude that \( \mathcal{F}_R \simeq R \) by Proposition 3.8.

**Remark 3.12** By Proposition 3.8, the isomorphism \( \mathcal{F}_R \simeq R \) is given by the map \( r \mapsto \bigcup \{ \Theta_R(\rho) \mid \rho \in r \} \); as observed in the proof of Lemma 3.10(iii), for any \( \rho \in \mathcal{L}_R \) there exists \( a \in \mathcal{K}(R) \) such that \( \rho \sim_R v_a \), therefore the filter \( r \) is mapped isomorphically to \( \bigcup \{ \Theta_R(v_a) \mid v_a \in r \} = \bigcup \{ a \mid v_a \in r \} \) by Definition 3.9. In case \( r \in \mathcal{K}(\mathcal{F}_R) \) then by Lemma 3.7 and the previous remarks \( r = \uparrow_R \rho = \uparrow_R v_a \) for some \( \rho \) and \( a \), and its image in \( R \) is just \( a \).

**Definition 3.13** (Type theories \( \mathcal{T}_D \) and \( \mathcal{T}_C \))

i) Let \( \mathcal{L}_D \) and \( \mathcal{L}_C \) be the intersection type languages defined by the grammar:
\[
\mathcal{L}_D : \quad \delta ::= \rho \mid \kappa \rightarrow \rho \mid \omega \mid \delta \land \delta \quad (\rho \in \mathcal{L}_R) \\
\mathcal{L}_C : \quad \kappa ::= \delta \times \kappa \mid \omega \mid \kappa \land \kappa
\]

We let \( \delta \) range over \( \mathcal{L}_D \), and \( \kappa \) over \( \mathcal{L}_C \), and \( \sigma, \tau \) over \( \mathcal{L}_D \cup \mathcal{L}_C \).

ii) We define \( \land_{i \in I} \sigma_i \) through:
\[
\land_{i \in \emptyset} \sigma_i = \omega \\
\land_{i \in I} \sigma_i = \sigma_p \land \land_{i \in I \setminus \{ p \}} \sigma_i \quad (p \in I)
\]
iii) The theories $T_D$ and $T_C$ are the least intersection type theories closed under the following axioms and rules, inducing the preorders $\leq_D$ and $\leq_C$ over $L_D$ and $L_C$ respectively:

\[
\begin{align*}
\frac{\rho_1 \leq \rho_2}{\rho_1 \leq_D \rho_2} & & \frac{\omega \leq_D \omega \to \omega} & & \frac{v \leq_D \omega \to v} & & \frac{\omega \leq_D v \omega \to v} & & \frac{\omega \leq_C \omega \times \omega}{
\end{align*}
\]

\[
\begin{align*}
(\kappa \to \delta_1) \land (\kappa \to \delta_2) & \leq_D \kappa \to (\delta_1 \land \delta_2) & & (\delta_1 \times \kappa_1) \land (\delta_2 \times \kappa_2) & \leq_C (\delta_1 \land \delta_2) \times (\kappa_1 \land \kappa_2) \\
\delta_2 \leq_C \kappa_1 & & \rho_1 \leq_D \rho_2 & & \kappa_1 \to \rho_1 \leq_D \kappa_2 \to \rho_2 & & \delta_1 \times \kappa_1 \leq_C \delta_2 \times \kappa_2 & & \delta_1 \times \kappa_1 \leq_C \delta_2 \times \kappa_2
\end{align*}
\]

As usual, we define $\sigma \sim_A \tau$ if and only if $\sigma \leq_A \tau \leq_A \sigma$, for $A = C, D$.

It is straightforward to show that both $(\sigma \land \tau) \land \rho = \tau \land (\sigma \land \rho)$ and $\sigma \land \tau = \tau \land \sigma$, so the type constructor $\land$ is associative and commutative, and we will write $\sigma \land \tau \land \rho$ rather than $(\sigma \land \tau) \land \rho$.

The pre-order $\leq_D$ is the usual one on arrow types in that the arrow is contravariant in the first argument and covariant in the second one. The pre-order $\leq_C$ on product types is covariant in both arguments, and is the component-wise pre-order. As immediate consequence of Definition 3.13 we have that $\omega \sim_D \omega \to \omega$, $\omega \sim_C \omega \times \omega$, $(\kappa \to \delta_1) \land (\kappa \to \delta_2) \sim_D \kappa \to (\delta_1 \land \delta_2)$ and $(\delta_1 \times \kappa_1) \land (\delta_2 \times \kappa_2) \sim_C (\delta_1 \land \delta_2) \times (\kappa_1 \land \kappa_2)$.

The equation $\omega \sim_D \omega \to \omega$ together with $v \sim_D \omega \to v$ are typical of filter models that are extensional $\lambda$-models. The equation $\omega \sim_C \omega \times \omega$ allows for a finite representation of compact elements in $C$, that otherwise should be described by infinite expressions of the form $\delta_1 \times \cdots \times \delta_2 \times \omega \times \omega \times \cdots$ (see points (i) and (ii) of Lemma 3.14).

We have also:

\[
\omega \sim_D \omega \to \omega \leq_D \kappa \to \omega \leq_D \omega,
\]

which implies that $\kappa \to \rho \sim_D \omega$ if and only if $\rho \sim_D \omega$.

Lemma 3.14

i) $\forall \kappa \in L_C \exists \delta_1, \ldots, \delta_n \in L_D [\kappa \sim_C \delta_1 \times \cdots \times \delta_n \times \omega]$.

ii) $\delta_1 \times \cdots \times \delta_n \times \omega \leq_C \delta'_1 \times \cdots \times \delta'_n \times \omega \iff h \leq k \land \forall i \leq h [\delta_i \leq_D \delta'_i]$.

iii) $\forall \delta \in L_C \exists n > 0, \kappa_1, \ldots, \kappa_n \in L_C, \rho_1, \ldots, \rho_n \in L_R [\delta \sim_D \land \{1, \ldots, n\} (\kappa_i \to \rho_i)]$.

iv) If $I, J$ are finite and non-empty sets of indexes and $\rho_i \neq_D \omega$ for all $i \in I$ then:

\[
\land_{i \in I} (\kappa'_i \to \rho'_i) \leq_D \land_{i \in J} (\kappa_i \to \rho_i) \iff \forall i \in I \exists j \in J [J_{i} \neq \emptyset \land \kappa_i \leq_C \land_{j \in I} \kappa_j \land \land_{j \in J} \rho_j \leq_D \rho_i].
\]

Proof: By induction on the structure of types and derivations in the theories $T_C$ and $T_D$.

Note that the equivalence $v \sim_D \omega \to v$ is necessary for Lemma 3.14(iii).

Parts (ii) and (iii) of the last lemma are rather straightforward; however they should be compared with Proposition 3.2, to realize how compact points in $C$ are represented by types in $L_C$. Parts (iii) and (iv) are characteristic of extended abstract type structures (shortly EATS: see e.g. [3] §3.3); in particular the latter implies:

\[
\land_{i \in I} (\kappa'_i \to \rho'_i) \leq_D \kappa \to \rho \Rightarrow \land \{ \rho'_i \mid \kappa \leq C \ k'_i \} \leq_R \rho.
\]


The next step is to define the mappings $\Theta_D : L_D \to K(D)$ and $\Theta_C : L_C \to K(C)$ such that both satisfy the hypotheses of Lemma 3.4. In doing that we shall exploit the property (1) in section 3.1, by introducing a stratification of $L_D$ and $L_C$ extending [24]. First we define the rank of a type in $L_D$ and $L_C$ inductively as follows:

\[
\begin{align*}
\text{rk}(\rho) & = \text{rk}(\omega) = 0 \\
\text{rk}(\sigma \land \tau) & = \max \{ \text{rk}(\sigma), \text{rk}(\tau) \} \\
\text{rk}(\delta \times \kappa) & = \max \{ \text{rk}(\delta), \text{rk}(\kappa) \} + 1 \\
\text{rk}(\kappa \to \rho) & = \text{rk}(\kappa) + 1.
\end{align*}
\]
Then we define $\mathcal{L}_{An} = \{ \sigma \in \mathcal{L}_A \mid rk(\sigma) \leq n \}$ for $A=D,C$. By this we have that if $n \leq m$ then $\mathcal{L}_{An} \subseteq \mathcal{L}_{Am}$ and that $\mathcal{L}_A = \bigcup_n \mathcal{L}_{An}$.

**Definition 3.15** The mappings $\Theta_{C_n} : \mathcal{L}_{C_n} \to \mathcal{K}(C_n)$ and $\Theta_{D_n} : \mathcal{L}_{D_n} \to \mathcal{K}(D_n)$ are defined by mutual induction:

$$
\begin{align*}
\Theta_{C_n}(\kappa) &= \bot \\
\Theta_{D_n}(v) &= (\bot \Rightarrow \Theta_R(v)) = \lambda \Rightarrow \Theta_R(v) \\
\Theta_{D_n}(\kappa \rightarrow \rho) &= (\Theta_{C_n}(\kappa) \Rightarrow \Theta_R(\rho)) \\
\Theta_{C_{n+1}}(\delta \times \kappa) &= (\Theta_{D_n}(\delta), \Theta_{C_n}(\kappa))
\end{align*}
$$

And, for $A_n = C_n, D_n$:

$$
\begin{align*}
\Theta_{A_n}(\omega) &= \bot \\
\Theta_{A_n}(\sigma \land \tau) &= \Theta_{A_n}(\sigma) \cup \Theta_{A_n}(\tau)
\end{align*}
$$

In the following lemma we prove that the mappings $\Theta_{C_n}$ and $\Theta_{D_n}$ are well defined, which is necessary since $\mathcal{L}_{C_n} \subseteq \mathcal{L}_{C_m}$ and $\mathcal{L}_{D_n} \subseteq \mathcal{L}_{D_m}$ when $n \leq m$.

**Lemma 3.16** For all $\kappa \in \mathcal{L}_C$ and $\delta \in \mathcal{L}_D$, if $rk(\kappa) \leq m$ and $rk(\delta) \leq n$ then

$$\Theta_{C_m}(\kappa) = \Theta_{C_{rk(\kappa)}}(\kappa) \quad \text{and} \quad \Theta_{D_n}(\delta) = \Theta_{D_{rk(\delta)}}(\delta).$$

**Proof:** By an easy induction over $m-rk(\kappa)$ and $n-rk(\delta)$ respectively. In fact let us consider the case of $\delta \times \kappa \rightarrow \rho$; then we have that $rk(\delta \times \kappa \rightarrow \rho) = p + 2$ where $p = \max\{rk(\delta), rk(\kappa)\}$ and

$$\Theta_{D_n}(\delta \times \kappa \rightarrow \rho) = (\Theta_{C_m}(\delta \times \kappa) \Rightarrow \Theta_R(\rho)) = (\Theta_{D_{n-1}}(\delta), \Theta_{C_{n-1}}(\kappa)) \Rightarrow \Theta_R(\rho)).$$

If $p + 2 \leq n$ then $rk(\delta) \leq p < p + 1 \leq n - 1$ and $rk(\kappa) \leq p < p + 1 \leq n - 1$, so that by induction:

$$\Theta_{D_{n-1}}(\delta) = \Theta_{D_{rk(\delta)}}(\delta) \quad \text{and} \quad \Theta_{C_{n-1}}(\kappa) = \Theta_{C_{rk(\kappa)}}(\kappa).$$

It follows that:

$$((\Theta_{D_{n-1}}(\delta), \Theta_{C_{n-1}}(\kappa)) \Rightarrow \Theta_R(\rho)) = ((\Theta_{D_{rk(\delta)}}(\delta), \Theta_{C_{rk(\kappa)}}(\kappa)) \Rightarrow \Theta_R(\rho)) = \Theta_{D_{p+2}}(\delta \times \kappa \rightarrow \rho).$$

For $A = D, C$, let $\leq_{An}$ be the preorder $\leq_A$ restricted to $\mathcal{L}_{An}$.

**Lemma 3.17** For every $n$, the mappings $\Theta_{C_n}$ and $\Theta_{D_n}$ are surjective and order reversing w.r.t. $\leq_{C_n}$ and $\leq_{D_n}$ respectively, i.e. they satisfy the hypotheses of Lemma 3.4.

**Proof:** By induction on the definition of $\Theta_{C_n}$ and $\Theta_{D_n}$. The language $\mathcal{L}_{C_0}$ is generated by the constant $\omega$ and the connectives $\times$ and $\land$, so all types in $\mathcal{L}_{C_0}$ are equated by $\sim_{C_0}$. Then the thesis holds for $\Theta_{C_0}$ since $C_0 = \{ \bot \}$. On the other hand, since the only way for a type in $\mathcal{L}_D$ to be of rank greater than 0 is to include an arrow, $\mathcal{L}_{D_0}$ is generated by the constants $\omega$ and $\nu_a$ for $a \in \mathcal{K}(R)$ and the connective $\land$ so that $\mathcal{L}_{D_0} = \mathcal{L}_R$. Besides, the isomorphism $D_0 \simeq R$ is given by the mapping $\lambda \Rightarrow a \mapsto a$ from $\mathcal{K}(D_0)$ to $\mathcal{K}(R)$. Then $\Theta_{D_0}(a) = (\bot \Rightarrow a) \mapsto a = \Theta_R(\nu_a)$, where the last mapping is an isomorphism of ordered sets, hence order preserving and respecting. We conclude that $\Theta_{D_0} = \Theta_R$ up to the isomorphism $\mathcal{K}(D_0) \simeq \mathcal{K}(R)$, hence it satisfies the hypotheses of Lemma 3.4 by Lemma 3.11.(ii).

Let $n > 0$. If $\kappa \in \mathcal{L}_{C_n}$ then:

$$\Theta_{C_n}(\kappa) = \begin{cases} 
(\Theta_{D_{n-1}}(\delta), \Theta_{C_{n-1}}(\kappa)) & \text{if } \kappa = \delta \times \kappa' \\
\Theta_{C_n}(\kappa_1) \cup \Theta_{C_n}(\kappa_2) & \text{if } \kappa = \kappa_1 \land \kappa_2 \\
\bot & \text{if } \kappa = \omega.
\end{cases}$$

For $\kappa = \delta \times \kappa'$ we have that $rk(\kappa) \leq n$ implies $rk(\delta'), rk(\kappa') \leq n-1$ by definition of $rk$; hence $\delta' \in \mathcal{L}_{D_{n-1}}$ and $\kappa' \in \mathcal{L}_{C_{n-1}}$. By induction, $\Theta_{D_{n-1}} : \mathcal{L}_{D_{n-1}} \to \mathcal{K}(D_{n-1})$ and $\Theta_{C_{n-1}} : \mathcal{L}_{C_{n-1}} \to \mathcal{K}(C_{n-1})$
are onto and order reversing. Since $\mathcal{K}(C_n) = \mathcal{K}(D_{n-1}) \times \mathcal{K}(C_{n-1})$, by induction $\Theta_{C_n}$ is onto and order reversing. If $\kappa = \kappa_1 \land \kappa_2$ then for any $\kappa_3 \in \mathcal{L}_{C_n}$:

$$\Theta_{C_n}(\kappa_1) \sqcup \Theta_{C_n}(\kappa_2) \sqsubseteq \Theta_{C_n}(\kappa_3) \iff \Theta_{C_n}(\kappa_1) \sqsubseteq \Theta_{C_n}(\kappa_3) \quad (i = 1, 2)$$

$$\iff \kappa_3 \leq_{C_n} \kappa_i \quad \text{(by a subordinate induction on } \kappa)$$

$$\iff \kappa_3 \leq_{C_n} \kappa_1 \land \kappa_2.$$

Finally, the case $\kappa = \omega$ is obvious as $\bot = (\bot, \bot)$ is the bottom in $\mathcal{K}(C_n)$, while $\omega \sim_{C_n} \omega \times \omega$ is the top in $(\mathcal{L}_{C_n}, \leq_{C_n})$.

If $\delta \in \mathcal{L}_{D_n}$ then:

$$\Theta_{D_n}(\delta) = \begin{cases} 
(\bot \Rightarrow \Theta_{R}(\rho)) & \text{if } \delta = \rho \in \mathcal{L}_{R} \\
(\Theta_{C_n}(\kappa) \Rightarrow \Theta_{R}(\rho)) & \text{if } \delta = \kappa \rightarrow \rho \\
\Theta_{D_n}(\delta_1) \sqcup \Theta_{D_n}(\delta_2) & \text{if } \delta = \delta_1 \land \delta_2 \\
\bot & \text{if } \delta = \omega.
\end{cases}$$

By construction $\mathcal{K}(D_n) = \mathcal{K}(\mathcal{C}_n \rightarrow R) = \{ \bigcup_{i \in I} (k_i \Rightarrow r_i) \mid I \text{ is finite} \land \forall i \in I. \ k_i \in \mathcal{K}(C_n) \land r_i \in \mathcal{K}(R) \}$. We know from the above that $\Theta_{C_n}$ is surjective (since both $\Theta_{D_{n-1}}$ and $\Theta_{C_{n-1}}$ are), while $\Theta_{R}$ is surjective by definition. Let $k_i = \Theta_{C_n}(k_i)$ and $r_i = \Theta_{R}(\rho_i)$, then $\bigcup_{i \in I} (k_i \Rightarrow r_i) = \Theta_{D_n}(\bigwedge_{i \in I} k_i \rightarrow \rho_i)$; since also $\Theta_{D_n}(\omega) = \bot = (\bot \Rightarrow \bot) = \Theta_{D_n}(\omega \rightarrow \omega)$, we conclude that $\Theta_{D_n}$ is surjective.

To see that $\Theta_{D_n}$ is order reversing, note that $(k \Rightarrow r) \subseteq f$ for $f \in [C_n \rightarrow R]$ if and only if $r \sqsubseteq f(k)$, which is trivially the case if $r = \bot$. Since $\bot = \Theta_{R}(\omega)$ and also $\bot = (k \Rightarrow \bot) = \Theta_{D_n}(\kappa \rightarrow \omega)$ for any $k$ and $\kappa$, while $\kappa \rightarrow \omega \sim_{D_n} \omega \leq_{D_n} \delta$ for any $\delta \in \mathcal{L}_{D_n}$, the thesis trivially holds if $r = \bot$.

Suppose that $r \neq \bot$. Since

$$(\bigcup_{i \in I} (k_i \Rightarrow r_i))(x) = \bigcup_{j \in I} r_j$$

for $J = \{ j \in I \mid k_j \sqsubseteq x \}$, we have

$$(k \Rightarrow r) \subseteq \bigcup_{i \in I} (k_i \Rightarrow r_i) \iff r \sqsubseteq \bigcup_{i \in I} (k_i \Rightarrow r_i)(k)$$

$$\iff \exists J \subseteq I [r \sqsubseteq \bigcup_{j \in J} r_j \land \bigcup_{j \in J} k_j \sqsubseteq k]$$

By subjectivity of $\Theta_{C_n}$ and $\Theta_{R}$ we know that there exist $\kappa, \rho$, such that $\Theta_{C_n}(\kappa) = k$ and $\Theta_{C_n}(\kappa_i) = k_i$, and $\kappa_i, \rho_i$ such that $\Theta_{R}(\rho) = r, \Theta_{R}(\rho_i) = r_i$, for every $i \in I$. Therefore,

$$\Theta_{D_n}(\kappa \rightarrow \rho) = (\Theta_{C_n}(\kappa) \Rightarrow \Theta_{R}(\rho)) \sqsubseteq \bigcup_{i \in I} (\Theta_{C_n}(\kappa_i) \Rightarrow \Theta_{R}(\rho_i))$$

$$\iff \exists J \subseteq I [\Theta_{R}(\rho) \sqsubseteq \bigcup_{j \in J} \Theta_{R}(\rho_j) \land \bigcup_{j \in J} \Theta_{C_n}(\kappa_j) \sqsubseteq \Theta_{C_n}(\kappa)]$$

$$\iff \exists J \subseteq I [\Theta_{R}(\rho) \sqsubseteq \Theta_{R}(\bigwedge_{j \in J} \rho_j) \land \Theta_{C_n}(\bigwedge_{j \in J} \kappa_j) \sqsubseteq \Theta_{C_n}(\kappa)]$$

$$\iff \exists J \subseteq I [\bigwedge_{j \in J} \rho_j \leq \rho \land \kappa \leq_{C_n} \bigwedge_{j \in J} \kappa_j]$$

$$\iff \land_{i \in I} (k_i \rightarrow \rho_i) \leq_{D_n} \kappa \rightarrow \rho \quad \text{(by 3.14.(iv))}$$

where we use that both $\Theta_{C_n}$ (as proved above) and $\Theta_{R}$ (by Lemma 3.10.(ii)) are order reversing, and that 3.14.(iv) applies because $\Theta_{R}(\rho) \neq \bot$ if and only if $\rho \neq R \omega$ by 3.10.(ii). The general case

$$\Theta_{D_n}(\bigwedge_{i \in I} \kappa_i \rightarrow \rho_i) = \bigcup_{i \in I} \Theta_{D_n}(\kappa_i \rightarrow \rho_i) \subseteq \Theta_{D_n}(\bigwedge_{j \in J} \kappa_j' \rightarrow \rho_j')$$

now follows, since this is equivalent to

$$\Theta_{D_n}(\kappa_i \rightarrow \rho_i) = (\Theta_{C_n}(\kappa_i) \Rightarrow \Theta_{R}(\rho_i)) \sqsubseteq \Theta_{D_n}(\bigwedge_{J \in I} \kappa_j' \rightarrow \rho_j') = \bigcup_{j \in I} (\Theta_{C_n}(\kappa_j') \Rightarrow \Theta_{R}(\rho_j'))$$

for all $i \in I$.  

17
Definition 3.18  The mappings $\Theta_D : \mathcal{L}_D \rightarrow \mathcal{K}(D)$ and $\Theta_C : \mathcal{L}_C \rightarrow \mathcal{K}(C)$ are defined by

$$\Theta_D(\delta) = \Theta_{Drk(\delta)}(\delta), \quad \Theta_C(\kappa) = \Theta_{Crk(\kappa)}(\kappa).$$

Remark 3.19 Because of Lemma 3.16 and of the definition of $rk$, we have that: $\Theta_D(\kappa \rightarrow \rho) = (\Theta_C(\kappa) \Rightarrow \Theta_R(\rho))$ and similarly that $\Theta_C(\delta \times \kappa) = (\Theta_D(\delta), \Theta_C(\kappa))$. In general all the equations in Definition 3.15 concerning the mappings $\Theta_{A_n}$ do hold for the respective maps $\Theta_A$.

Lemma 3.20  The mappings $\Theta_D$ and $\Theta_C$ are surjective and order reversing, namely satisfy the hypotheses of Lemma 3.4.

Proof: First let us observe that if $n \geq rk(\delta)$ then $\Theta_D(\delta) = \Theta_{Drk(\delta)}(\delta) = \Theta_{D_n}(\delta)$ by Lemma 3.16, and similarly for $\Theta_C$. Now let $d \in \mathcal{K}(D)$ then by the equations (1) we have $\mathcal{K}(D) = \bigcup_n \mathcal{K}(D_n)$, so that there exists $n$ such that $d \in \mathcal{K}(D_n)$. By Lemma 3.17 $\Theta_{D_n}$ is surjective, hence there exists $\delta \in \mathcal{L}_{D_n}$ s.t. $\Theta_{D_n}(\delta) = d$. It follows that $rk(\delta) \leq n$ and $\Theta_D(\delta) = \Theta_{D_n}(\delta) = d$, by the above remark. Hence $\Theta_D$ is surjective.

On the other hand if $\delta_1 \subseteq \delta_2$ then $\delta_1 \subseteq D_n \delta_2$ for any $n \geq \max\{rk(\delta_1), rk(\delta_2)\}$; by Lemma 3.17 and the above remark we conclude that $\Theta_D(\delta_1) = \Theta_{D_n}(\delta_1) \supseteq \Theta_{D_n}(\delta_2) = \Theta_D(\delta_1)$, which establishes that $\Theta_D$ is order reversing. The proof concerning $\Theta_C$ is the same.

Theorem 3.21  The filter domains $\mathcal{F}_R, \mathcal{F}_D$ and $\mathcal{F}_C$ are isomorphic to $R, D$ and $C$ respectively.

Proof: That $\mathcal{F}_R \simeq R$ is stated in Corollary 3.11. By Lemma 3.20 $\Theta_D$ and $\Theta_C$ satisfy Lemma 3.4, hence we conclude by Proposition 3.8.

Theorem 3.21 implies that $(\mathcal{F}_R, \mathcal{F}_D, \mathcal{F}_C)$ is a $\lambda \mu$-model. However it is a rather implicit description of the model on which we base the construction of the intersection type assignment system in the next section. To get a better picture relating term and type interpretation, we show how functional application and the operation of adding an element of $\mathcal{F}_D$ in front of a continuation in $\mathcal{F}_C$ are defined in this model; this provides us with a more explicit description of the isomorphisms relating $\mathcal{F}_D$ and $\mathcal{F}_C$.

In the following, we let $d$ and $k$ range over filters in $\mathcal{F}_D$ and $\mathcal{F}_C$, respectively; notice that above they were used for elements of $C$ and $D$. Since no confusion is possible, and there is a clear link between these concepts, we permit ourselves a little overloading in notation.

Definition 3.22  For $d \in \mathcal{F}_D$ and $k \in \mathcal{F}_C$ we define:

$$d \cdot k = \uparrow_D \{ \rho \in \mathcal{L}_R \mid \exists \kappa \rightarrow \rho \in d[\kappa \in k] \}$$

$$d :: k = \uparrow_C \{ \land_{i \in I} \delta_i \times \kappa_i \in \mathcal{L}_C \mid \forall i \in I [\delta_i \in d \& \kappa_i \in k] \}$$

The upward closure $\uparrow_D$ in the definition of $d \cdot k$ is redundant. We have added it in order to simplify the proofs; in fact any set of types $\uparrow A$ is clearly closed under $\sim$. A similar remark holds for $\uparrow_C$ in the definition of $d :: k$, where we have to include $\omega$. Alternatively one could stipulate the usual convention that $\land_{i \in I} \delta_i \times \kappa_i$ is syntactically the same as $\omega$ when $I = \emptyset$.

Lemma 3.23  $d \cdot k \in \mathcal{F}_R$ and $d :: k \in \mathcal{F}_C$, for any $d \in \mathcal{F}_D$ and $k \in \mathcal{F}_C$. Moreover, the mappings $\_ \cdot \_$ and $\_ :: \_ $ are continuous in both their arguments.

Proof: The proof that $d \cdot k$ is well defined and continuous is essentially the same as that with EATS: see e.g. [3] §3.3. The set $d :: k$ is a filter by definition. By definition unfolding we have that

$$d :: k = \bigcup_{\delta \in d} (\bigcup_{\kappa \in k} \uparrow_C \delta) = \bigcup_{\delta \in d, \kappa \in k} \uparrow_D \delta \cdot \uparrow_C \kappa,$$

hence $\_ :: \_ $ is continuous.
We have reported in section 3.1 the definition of step functions. In the particular case of \([F_C \rightarrow F_R]\), step functions take the form \((↑C \kappa \Rightarrow ↑R \rho)\). Indeed for \(k \in F_C\) we have that \(↑C \kappa \subseteq k\) if and only if \(\kappa \in k\), so that we have:

\[
(↑C \kappa \Rightarrow ↑R \rho)(k) = \begin{cases} ↑R \rho & \text{if } \kappa \in k \\ ↑R \omega & \text{else} \end{cases} = ↑D(κ→ρ) \cdot k
\]

Thus arrow types represent step functions. Similarly, the product of domains \(X \times Y\) ordered component-wise is a domain such that \(K(X \times Y) = K(X) \times K(Y)\). In case of \(F_D \times F_C\) compact points are of the shape \((↑D \delta, ↑C \kappa)\), which corresponds to the filter \(↑C \delta \times \kappa \in F_C\). This justifies the following definition:

**Definition 3.24** We define the following maps:

\[
\begin{align*}
F : F_D & \rightarrow [F_C \rightarrow F_R] & Fd k &= d \cdot k \\
G : [F_C \rightarrow F_R] & \rightarrow F_D & G f &= ↑D \{ \land_{i \in I} ρ_i \in L_D \mid ∀i \in I[ρ_i \in f(↑κ_i)] \} \\
H : F_C & \rightarrow (F_D \times F_C) & Hk &= \{ \{ δ \in L_D \mid δ \times \kappa \in k \}, \{ \kappa \in L_C \mid δ \times \kappa \in k \} \} \\
K : (F_D \times F_C) & \rightarrow F_C & K(d, k) &= d \cdot k
\end{align*}
\]

**Remark 3.25** As expected form the claim that step functions in \([F_C \rightarrow F_R]\) are represented by arrow types in \(L_D\), for any \(κ \rightarrow ρ \in L_D\) we have \(G(↑C \kappa \Rightarrow ↑R ρ) = ↑D(κ→ρ)\). Indeed \(κ→ρ \leq_D κ'→ρ'\) if and only if \(κ' \leq_C κ\) and \(ρ \leq_R ρ'\), that is \(↑C \kappa \subseteq ↑C κ'\) and \(↑R ρ' \subseteq ↑R ρ\), namely if and only if \(↑D \delta \times ↑C \kappa = ↑D \delta \times ↑C κ'\).

Similarly types \(δ \times κ \in L_C\) represent pairs in \(F_D \times F_C\) via \(K\), that is \(K(↑D \delta, ↑C κ) = ↑D δ \cdot ↑C κ = ↑C (δ \times κ)\).

We will write \(↑ρ\) for \(↑R ρ\) when no ambiguity is possible, and similarly for \(↑D\) and \(↑C\).

**Lemma 3.26** The functions \(F, G\) and \(H, K\) are well defined and monotonic w.r.t. subset inclusion.

**Proof:** By Lemma 3.23, \(F\) and \(K\) are well defined and continuous, hence they are monotonic. We need to check that \(G\) is well defined and continuous, hence monotonic. For all \(f \in [F_C \rightarrow F_R]\) the set \(G f\) is a filter over \((L_D, \subseteq_D)\) by definition; we check that \(G\) is monotonic. Observe that \(\land_{i \in I} ρ_i \in G f\) if and only if \(\lor_{i \in I}(↑κ_i \Rightarrow ↑ρ_i) \subseteq f\); on the other hand, if \(f \subseteq g\) then \(\lor_{i \in I}(↑κ_i \Rightarrow ↑ρ_i) \subseteq g\) implies \(\land_{i \in I}(↑κ_i \Rightarrow ↑ρ_i) \subseteq f\), so \(\land_{i \in I} ρ_i \in G f\) implies \(\land_{i \in I} ↑κ_i \Rightarrow ↑ρ_i \in G g\).

The function \(H\) is evidently monotonic w.r.t. \(\subseteq\). We check that it is well defined, i.e. that both \(d' = \{ δ \in L_D \mid δ \times κ \in k \}\) and \(k' = \{ κ \in L_C \mid δ \times κ \in k \}\) are filters, whenever \(k\) is one. Let \(δ_1, δ_2 \in d'\), then there exist \(κ_1, κ_2\) such that \(δ_1 \times κ_1, δ_2 \times κ_2 \in k\) (and hence \(κ_1, κ_2 \in k'\)). Since \(k\) is a filter, we have \(δ_1 \times κ_1 \land δ_2 \times κ_2 \in k\); also, \(δ_1 \times κ_1 \land δ_2 \times κ_2 \sim_C (δ_1 \land δ_2) \times (κ_1 \land κ_2)\) implies \((δ_1 \land δ_2) \times (κ_1 \land κ_2)\) is a filter, closed under meets and \(\sim_C\). We conclude that \(δ_1 \land δ_2 \in d'\) (similarly we know that \(κ_1 \land κ_2 \in k'\)). The same reasoning shows that both \(d'\) and \(k'\) are upward closed sets with respect to \(\subseteq_D\) and \(\subseteq_C\), respectively.

**Theorem 3.27** The following isomorphisms exist: \(F_D \simeq [F_C \rightarrow F_R]\) via \(F\) with inverse \(G\), and \(F_C \simeq F_D \times F_C\) via \(H\) and its inverse \(K\).

**Proof:** Since any monotonic function of posets that is invertible is an isomorphism, by Lemma 3.26 it suffices to show that \(G = F^{-1}\) and \(K = H^{-1}\).

\[i) (F \circ G) f k = F(↑\{ \land_{i \in I} ρ_i \in L_D \mid ∀i \in I[ρ_i \in f(↑κ_i)] \}) k = ↑\{ \land_{i \in I} ρ_i \in L_D \mid ∀i \in I[ρ_i \in f(↑κ_i)] \} \cdot k = ↑\{ ρ \mid \exists k \in k[ρ \in f(↑κ)] \} = \bigcup_{κ \in k} f(↑κ)
\]

(since \(↑κ \mid κ \in k\) is directed)

(by continuity of \(f\)
hence \((F \circ G)f = f\).

\[ ii) \quad (G \circ F)d = G(\langle k \in \mathcal{F}_C \mid d \cdot k \rangle) \]

\[ = \uparrow \{ \bigwedge_{i \in I} \kappa_i \to \rho_i \mid \forall i \in I [\rho_i \in d \cdot \kappa_i] \} \]

\[ = \uparrow \{ \bigwedge_{i \in I} \kappa_i \to \rho_i \mid \forall i \in I \exists \kappa_i' [\kappa_i \leq \kappa_i' \land \kappa_i' \to \rho_i \in d] \} \]

\[ = d \]

where \(\lambda\) represents semantic abstraction; in the last equation, the inclusion \(\supseteq\) is obvious, while the inclusion \(\subseteq\) follows by the fact that if \(\kappa_i \leq \kappa_i'\) then \(\kappa_i' \to \rho_i \leq D_{\kappa_i} \kappa_i \to \rho_i\), hence \(\kappa_i' \to \rho_i \in d\) implies \(\kappa_i \to \rho_i \in d\) for all \(i \in I\), which in turn implies that \(\bigwedge_{i \in I} \kappa_i \to \rho_i \in d\).

\[ iii) \quad (H \circ K)(d, k) = H(\langle d :: k \rangle) \]

\[ = \{ \delta \in \mathcal{L}_D \mid \delta \times \kappa \in d :: k \}, \{ \kappa \in \mathcal{L}_C \mid \delta \times \kappa \in d :: k \} \]

\[ = \langle d, k \rangle \]

by observing that \(\delta \times \kappa \in d :: k\) if and only if \(\delta \in d\) and \(\kappa \in k\), and that if \(\kappa' \leq \omega \leq \delta :: k\) then \(\kappa' \sim \kappa \omega \times \omega\) and obviously \(\omega \in d\) and \(\omega \in k\).

\[ iv) \quad (K \circ H)k = \{ \delta \in \mathcal{L}_D \mid \delta \times \kappa \in k \} :: \{ \kappa \in \mathcal{L}_C \mid \delta \times \kappa \in k \} \]

\[ = \uparrow \{ \bigwedge_{i \in I} \delta_i \times \kappa_i \mid \forall i \in I \exists \delta_i', \kappa_i' [\delta_i \times \kappa_i \leq \delta_i' \times \kappa_i \in k] \} \]

\[ = k \quad \text{(since } k \in \mathcal{F}_C) \]

Remark 3.28 As observed in Remark 2.2, a \(\lambda\mu\)-model is an extensional \(\lambda\)-model. Theorem 3.1 in [49] states that the initial/final solution of the continuation domain equations is isomorphic to the domain \(R_\infty \simeq [R_\infty \rightarrow R_\infty]\), i.e. the Scott’s \(D_\infty\) \(\lambda\)-model obtained as inverse limit of a chain where \(D_0 = R\).

To see this from the point of view of the intersection type theory, consider the extension \(\mathcal{L}_\Lambda = \cdots \mid \delta \rightarrow \delta\) of \(\mathcal{L}_D\). Let \(T_\Lambda\) be the theory obtained by adding to \(T_D\) the equation \(\delta \in \kappa \rightarrow \rho = \delta \rightarrow \kappa \rightarrow \rho\). In the intersection type theory \(T_\Lambda\), the following rules are derivable:

\[
(\delta \rightarrow \delta_1) \land (\delta \rightarrow \delta_2) \leq_\Lambda \delta \rightarrow (\delta_1 \land \delta_2) \]

\[
\frac{\delta_1' \leq_\Lambda \delta_1 \quad \delta_2 \leq_\Lambda \delta_2}{\delta_1 \rightarrow \delta_2 \leq_\Lambda \delta_1' \rightarrow \delta_2'}
\]

By this, \(T_\Lambda\) is a natural equated intersection type theory in terms of [2], and hence \(\mathcal{F}_\Lambda \simeq [\mathcal{F}_\Lambda \rightarrow \mathcal{F}_\Lambda]\) where \(\mathcal{F}_\Lambda\) is the set of filters generated by the preorder \(\leq_\Lambda\) (see [2], Cor. 28(4)).

4 An intersection type system

Let \(M = (R, D, C)\) be a \(\lambda\mu\)-model, where \(D, C\) are initial solutions of the continuation domain equations (hence we say that \(M\) is initial). In this section, using the fact that \(M\) is isomorphic to the filter model \(\mathcal{F} = (\mathcal{F}_R, \mathcal{F}_D, \mathcal{F}_C)\) established by theorems 3.21 and 3.27, we will define an assignment system such that the typing \(M : \delta\) (or \(C : \kappa\)) is derivable, under appropriate assumptions about the variables and names in it, if and only if \(\text{⟦}M\text{⟧}^\mathcal{F} e \in \Lambda \delta\) (or \(\text{⟦}C\text{⟧}^\mathcal{F} e \in \Lambda \kappa\)) for all environments \(e\) respecting the assumptions.

Because of this an interpretation of types can be defined such that \(\text{⟦}\sigma\text{⟧}^\mathcal{F} = \uparrow_\Lambda \sigma\) for \(\Lambda = D, C\). Since filters are upward closed sets of types, we have that \(\text{⟦}T\text{⟧}^\mathcal{F} e \in \Lambda \sigma\) holds if and only if \(\sigma \in \text{⟦}T\text{⟧}^\mathcal{F} e\), and we obtain that the denotation of a term/command is just the set of types that can be inferred for it in the assignment system.

4.1 Type assignment

We now give some preliminary definitions for our type system.

Definition 4.1 (Bases, Name Contexts and Judgements)
i) A basis is a finite mapping from term variables to types in $T_D$, written as a finite set $\Gamma = \{x_1: \delta_1, \ldots, x_n: \delta_n\}$ where the term variables $x_i$ are pairwise distinct.

ii) A name context (or context for short) is a finite mapping from names to types in $T_C$, written as a finite set $\Delta = \{\alpha_1: \kappa_1, \ldots, \alpha_m: \kappa_m\}$ where the continuation variables $\alpha_i$ are pairwise distinct.

iii) We write $\Gamma, x: \delta$ for the basis $\Gamma \cup \{x: \delta\}$, so assuming that either $x$ does not occur in $\Gamma$ or $x: \delta \in \Gamma$, and similarly for $\alpha: \kappa, \Delta$.

iv) Let $\Gamma$ be a basis and $\Delta$ a name context. We define

$$
\text{dom}(\Gamma) \overset{\Delta}{=} \{x \mid \exists \delta [x: \delta \in \Gamma]\}
$$

$$
\text{dom}(\Delta) \overset{\Delta}{=} \{\alpha \mid \exists \kappa [\alpha: \kappa \in \Delta]\}
$$

Judgements are in appearance very similar to Parigot’s, apart from the obvious difference in the language of types; in fact, there is a relation among Parigot’s system and the one presented here, which will be treated in detail in Section 7. Being bases and contexts sets, the order in which variable and name assumptions are listed is immaterial.

Let $\Gamma$ be a basis and $\Delta$ a context; then we introduce the following abbreviations:

$$
\Gamma(x) = \begin{cases} 
\delta & \text{if } x: \delta \in \Gamma \\
\omega & \text{otherwise}
\end{cases}
$$

$$
\Delta(\alpha) = \begin{cases} 
\kappa & \text{if } \alpha: \kappa \in \Delta \\
\omega & \text{otherwise}
\end{cases}
$$

**Definition 4.2 (Intersection Type System for $\lambda\mu$)**

Type rules:

$$(Ax) : \frac{}{\Gamma, x: \delta \vdash M: \kappa \rightarrow \rho \mid \Delta}$$

$$(Abs) : \frac{\Gamma, x: \delta \vdash M: \kappa \rightarrow \rho \mid \Delta}{\Gamma \vdash \lambda x. M: \delta \times \kappa \rightarrow \rho \mid \Delta} \quad \text{($x \not\in \text{dom}(\Gamma)$)}$$

$$(App) : \frac{\Gamma \vdash M: \delta \rightarrow \rho \mid \Delta \quad \Gamma \vdash N: \delta \mid \Delta}{\Gamma \vdash MN: \kappa \rightarrow \rho \mid \Delta}$$

$$(mu) : \frac{\Gamma \vdash C: \langle \delta \rightarrow \rho \rangle \times \kappa \mid \alpha: \kappa, \Delta}{\Gamma \vdash \mu \alpha. C: \delta \rightarrow \rho \mid \Delta} \quad \text{($\alpha \not\in \text{dom}(\Delta)$)}$$

$$(Cmd) : \frac{\Gamma \vdash M: \delta \mid \Delta}{\Gamma \vdash [\alpha] M: \delta \times \kappa \mid \Delta} \quad \text{($\Delta(\alpha) = \kappa$)}$$

Logical rules:

$$(\wedge) : \frac{\Gamma \vdash T: \sigma \mid \Delta \quad \Gamma \vdash T: \tau \mid \Delta}{\Gamma \vdash T: \sigma \wedge \tau \mid \Delta}$$

$$(\omega) : \frac{\Gamma \vdash T: \omega \mid \Delta}{\Gamma \vdash T: \omega \mid \Delta}$$

$$(\leq) : \frac{\Gamma \vdash T: \sigma \mid \Delta \quad \sigma \leq \tau}{\Gamma \vdash T: \tau \mid \Delta}$$

A judgement is an expression of the form $\Gamma \vdash M: \delta \mid \Delta$ or $\Gamma \vdash C: \kappa \mid \Delta$ where $\Gamma$ is a basis and $\Delta$ is a name context; $M$ and $C$ are the subjects and $\delta$ and $\kappa$ the predicates.

We will write $\Gamma \vdash M: \delta \mid \Delta$ if there is a derivation built using the above rules that has this judgment in the bottom line, and $D : \Gamma \vdash M: \delta \mid \Delta$ if we want to name that derivation.

We extend Barendregt’s convention to judgements $\Gamma \vdash T: \delta \mid \Delta$ by seeing the variables that occur in $\Gamma$ and names in $\Delta$ as binding occurrences over $T$ as well; in particular, we can assume that no variable in $\Gamma$ and no name in $\Delta$ is bound in $T$.

To understand these rules we can think of types as properties of term denotations in the initial model $\mathcal{M} = (R, D, C)$. In particular, if $\sigma \in \mathcal{L}_A$ then $\sigma$ denotes a subset $[\sigma]^M \subseteq A$, for $A = R, D, C$. The judgement $\Gamma \vdash T: \sigma \mid \Delta$ is then interpreted as the claim that $\{T\}^M \in [\sigma]^M$ whenever $e x \in \Gamma(x)$ and $e \alpha \in \Delta(\alpha)$ for all $x$ and $\alpha$ (formal definitions are given in Section 4.2 below).

The logical rules, which are familiar from intersection type systems for the standard $\lambda$-calculus, just state that types are sets: $\omega$ is the largest set which coincides with the domain of interpretation itself, the pre-order is subset inclusion and $\sigma \wedge \tau$ is the set theoretic intersection.
Remark 4.3. Note how rules have been obtained from the equations in Definition 2.3 by representing the left-hand side of the equation in the conclusion and the right-hand side in the premises of the corresponding rule:

**Rule (Abs):** corresponds to the equation $[[\lambda x. M]]^D e \langle d, k \rangle = [[M]]^D e[x \mapsto d]k$, where $[[\cdot]]^D$ is short for $[[\cdot]]^\mathcal{M}_D$. It states that $\lambda x. M$ is a function of continuations $\langle d, k \rangle$, whose values are those of $M$ where $x$ is interpreted by $d$, and applied to continuation $k$. On the other hand, the arrow types from $\mathcal{L}_D$ represent properties of functions: a property of $\lambda x. M$ is then a type $\delta \times \kappa \to \rho$ (the conclusion of the rule) so that whenever $\delta \times \kappa$ is a property of $\langle d, k \rangle$, i.e. $d \in [[\delta]]^\mathcal{M}$ and $k \in [[\kappa]]^\mathcal{M}$, $\rho$ is a property of the result. But since the result is $[[M]]^D e[x \mapsto d]k$, it suffices to prove that $M$ has the property $\kappa \to \rho$ whenever $x$ is interpreted by $d$, which is represented by the assumption $x : \delta$ in the premise of (Abs).

**Rule (App):** dually, comes from the equation $[[M N]]^D e k = [[M]]^D e \langle [[N]]^D e, k \rangle$. For the application $M N$ to have the property $\kappa \to \rho$ (as in the conclusion) it suffices that if applied to a continuation $k \in [[\kappa]]^\mathcal{M}$ it yields a value with property $\rho$. By the equation, such a value is computed by putting $[[N]]^D e$ before $k$ in the continuation passed to $[[M]]^D e$. Therefore, for the conclusion to hold it suffices to prove that $N$ has type $\delta$ and $M$ type $\delta \times \kappa \to \rho$ (premises).

**Rule (Cmd):** is based on the equation $[[\alpha] M] e = [[M]]^D e, e \alpha$, which says that the meaning of a command $[\alpha] M$ is a continuation $\langle d, k \rangle$ where $d$ is the meaning of $M$ and $k = e \alpha$. For $\langle d, k \rangle$ to have the property $\delta \times \kappa$ (as in the conclusion) we have to check that $M$ has the property $\delta$ whenever $\alpha$ denotes the continuation $k$ with property $\kappa$. Since the assumptions about the environment are in the contexts $\Delta$ in case of names, this is represented by the side condition $\Delta(\alpha) = \kappa$ of the rule.

**Rule (\mu):** is the more involved case, which corresponds to the equation $[[\mu a. C]]^D e k = dk'$, where $\langle d, k' \rangle' = [[C]]^e [\alpha \mapsto k]$. This states that $[[\mu a. C]]^D e$ is the function that, when applied to a continuation $k$ yields the value of the application of the first component $d$ to the second component $k'$ of a different continuation $\langle d, k' \rangle$, which however depends on $k$, because it is computed by $C$ whenever $\alpha$ is sent to $k$. Now the result $dk'$ will have the property $\rho$ if for some $\kappa'$ both $k' \in [[\kappa']]^\mathcal{M}$ and $d \in [[\kappa'] \to \rho]]^\mathcal{M}$. Therefore, to type $\mu a. C$ by $\delta \to \rho$ (as in the conclusion) we have to ensure that the continuation represented by $C$ has the property $(\kappa' \to \rho) \times \kappa'$, whenever $a : \kappa$ occurs in the context (as in the premise).

**Remark 4.3** Note how rules (App) and (Abs) are actually instances of the familiar rules for application and $\lambda$-abstraction in the simply typed $\lambda$-calculus. In fact, $\delta \times \kappa \to \rho \in \mathcal{L}_D$ is equivalent to $\delta \to (\kappa \to \rho) \in \mathcal{L}_\lambda$ so that, if we admitted types of $\mathcal{L}_\lambda$, the following rules would be admissible:

$$
\frac{\Gamma \vdash M : \delta \to (\kappa \to \rho) \mid \Delta}{\Gamma \vdash MN : \kappa \to \rho \mid \Delta} \quad (\text{App'}) \quad \frac{\Gamma \vdash N : \delta \mid \Delta}{\Gamma \vdash MN : \kappa \to \rho \mid \Delta} \quad (\text{Abs'}) \quad \frac{\Gamma, x : \delta \vdash M : \kappa \to \rho \mid \Delta}{\Gamma \vdash \lambda x. M : \delta \to (\kappa \to \rho) \mid \Delta}
$$

We will also use the following variant of rule ($\wedge$):

$$
\frac{\Gamma \vdash T : \sigma_i \mid \Delta \quad (\forall i \in I)}{\Gamma \vdash T : \bigwedge_{i \in I} \sigma_i \mid \Delta}
$$

22
Remark 4.4  Rule (Cmd) is equivalent to the following two:

\[(Cmd_1) : \frac{\Gamma \vdash M : \delta \mid \Delta}{\Gamma \vdash \sigma[M : \alpha \mid \alpha : \kappa, \Delta]} \quad (Cmd_2) : \frac{\Gamma \vdash M : \delta \mid \Delta}{\Gamma \vdash \alpha[M : \delta \times \omega \mid \Delta]}\]

By definition the context \(\alpha : \kappa, \Delta\) in the conclusion of \((Cmd_1)\) is only legal when either \(\alpha \not\in \text{dom}(\Delta)\) or \(\alpha : \kappa \in \Delta\).

The need of \((Cmd_2)\) will be apparent when proving the admissibility of the strengthening rule (Lemma 4.8) and the completeness of the type assignment. For the moment we observe that with \((Cmd_1)\) the conclusion of the shape \(\Gamma \vdash \alpha[M : \delta \mid \Delta]\) would be derivable from \(\Gamma \vdash M : \delta \mid \Delta\) only if \(\alpha : \omega \in \Delta\); on the contrary \(\Delta(\alpha) = \omega\) doesn’t require that \(\alpha \in \text{dom}(\Delta)\). On the other hand \((Cmd_2)\) allows the implicit typing of \(\alpha\) by \(\omega\) even if \(\alpha \not\in \text{dom}(\Delta)\). Not having rule \((Cmd_2)\) (that is a particular case of \((Cmd_d)\)) would introduce an asymmetry w.r.t. the typing with \(\omega\) of the term variable \(x\) in a basis \(\Gamma\), since we can conclude \(\Gamma \vdash x : \omega \mid \Delta\) either by rule \((Ax)\) (in which case \(x : \omega \in \Gamma\) is required), or by rule \((\omega)\), where \(x \not\in \text{dom}(\Gamma)\) is allowed.

With the above proviso, in the proofs we shall often consider as instances of \((Cmd)\) the rules \((Cmd_1)\) and \((Cmd_2)\) without explicit mention.

The admissibility of the following rules will be useful:

Lemma 4.5 (Admissibility of Weakening and Thinning)  The following rules are admissible:

\[(W) : \frac{\Gamma \vdash T : \sigma \mid \Delta}{\Gamma' \vdash T : \sigma \mid \Delta'} \quad (T) : \frac{\Gamma \vdash M : \delta \mid \Delta}{\Gamma' \vdash M : \delta \mid \Delta'}\]

\((W') : \frac{\Gamma \vdash T : \sigma \mid \Delta}{\Gamma' \vdash T : \sigma \mid \Delta'} \quad (T') : \frac{\Gamma \vdash M : \delta \mid \Delta}{\Gamma' \vdash M : \delta \mid \Delta'}\]

\((W'') : \frac{\Gamma \vdash T : \sigma \mid \Delta}{\Gamma' \vdash T : \sigma \mid \Delta'} \quad (T'') : \frac{\Gamma \vdash M : \delta \mid \Delta}{\Gamma' \vdash M : \delta \mid \Delta'}\]

Proof:  Easy.

Notice that, by Barendregt’s convention, the variables in \(\Gamma'\) and names in \(\Delta'\) are not bound in \(\Delta\).

Remark 4.6  Notice that weakening is only sound when extending Barendregt’s convention to judgements. If we would not do so, then, for example, we could derive

\[
\frac{x : \kappa \rightarrow \rho \vdash x : \kappa \rightarrow \rho}{(Ax)} \\
\frac{\vdash \lambda x. x : (\kappa \rightarrow \rho) \times \kappa \rightarrow \rho}{(Abs)} \\
\frac{x : \delta \vdash \lambda x. x : (\kappa \rightarrow \rho) \times \kappa \rightarrow \rho}{(W)}
\]

Note that we cannot apply rule \((Abs)\) to this derivation, since that would create the term \(\lambda x. \lambda x. x\), which is not a valid term. We can solve this through \(\alpha\)-conversion, replacing \(\lambda x. x\) by \(\lambda y. y\);

\[
\frac{y : \kappa \rightarrow \rho \vdash y : \kappa \rightarrow \rho}{(Ax)} \\
\frac{\vdash \lambda y. y : (\kappa \rightarrow \rho) \times \kappa \rightarrow \rho}{(Abs)} \\
\frac{x : \delta \vdash \lambda y. y : (\kappa \rightarrow \rho) \times \kappa \rightarrow \rho}{(W)}
\]

notice that then the application of rule \((W)\) is valid according to our criteria.

In presence of subtyping (rule \((\leq)\)) we can have a further form of weakening, namely by weakening the types in the assumptions. Let us extend the operator \(\wedge\) and the pre-orders \(\leq_D\) and \(\leq_C\) to bases and contexts.
Definition 4.7

i) If $\Gamma_1, \Gamma_2$ are bases then we define the basis $\Gamma_1 \land \Gamma_2$ by:

$$\Gamma_1 \land \Gamma_2 \triangleq \{ x : \Gamma_1(x) \land \Gamma_2(x) \mid x \in \text{dom}(\Gamma_1) \cap \text{dom}(\Gamma_2) \}$$

$$\cup \{ x : \delta \in \Gamma_1 \mid x \notin \text{dom}(\Gamma_2) \}$$

$$\cup \{ x : \delta \in \Gamma_2 \mid x \notin \text{dom}(\Gamma_1) \}$$

Similarly if $\Delta_1, \Delta_2$ are contexts then we define the context $\Delta_1 \land \Delta_2$ by:

$$\Delta_1 \land \Delta_2 \triangleq \{ a : \Delta_1(a) \land \Delta_2(a) \mid a \in \text{dom}(\Delta_1) \cap \text{dom}(\Delta_2) \}$$

$$\cup \{ a : \kappa \in \Delta_1 \mid a \notin \text{dom}(\Delta_2) \}$$

$$\cup \{ a : \kappa \in \Delta_2 \mid a \notin \text{dom}(\Delta_1) \}$$

ii) We extend the relations $\leq_D$ and $\leq_C$ to bases and contexts respectively by:

$$\Gamma_1 \leq_D \Gamma_2 \iff \forall x \in \text{VAR.} \ \Gamma_1(x) \leq_D \Gamma_2(x)$$

$$\Delta_1 \leq_C \Delta_2 \iff \forall a \in \text{NAME.} \ \Delta_1(a) \leq_C \Delta_2(a)$$

Note that, if $\Gamma_1, \Gamma_2$ are well-formed bases and $\Delta_1, \Delta_2$ are well-formed contexts, then $\Gamma_1 \land \Gamma_2$ is a well-formed basis and $\Delta_1 \land \Delta_2$ is a well-formed context. Also it is clear that $\text{dom}(\Gamma_1 \land \Gamma_2) = \text{dom}(\Gamma_1) \cup \text{dom}(\Gamma_2)$ and $\text{dom}(\Delta_1 \land \Delta_2) = \text{dom}(\Delta_1) \cup \text{dom}(\Delta_2)$. Because of this our $\Gamma_1 \land \Gamma_2$ is often called “union of bases” in the literature.

By introducing the abbreviations $\Gamma(x)$ and $\Delta(a)$ we implicitly see bases and contexts as (total) functions from term variables to $\mathcal{L}_D$ and from names to $\mathcal{L}_C$ respectively. The relations $\Gamma_1 \leq_D \Gamma_2$ and $\Delta_1 \leq_C \Delta_2$ are then the pointwise extensions of the relations $\leq_D$ and $\leq_C$ over types, where note that the quantifications are not restricted to the domains of the bases nor of the contexts.

It is also an immediate consequence of the definition that $\Gamma_1 \land \Gamma_2 \leq_D \Gamma_i$ and that $\Delta_1 \land \Delta_2 \leq_C \Delta_i$, for $i = 1, 2$. However if $\Gamma_1 \leq_D \Gamma_2$ then $\text{dom}(\Gamma_1)$ and $\text{dom}(\Gamma_2)$ are unrelated in general, since we have e.g. $\{ x : \omega, y : \delta_1 \land \delta_2 \} \leq_D \{ z : \omega, y : \delta_1 \}$. It follows that $\Gamma_1 \leq_D \Gamma_2$ don’t imply that $\Gamma_1$ and $\Gamma_1 \land \Gamma_2$ are the same, as one could expect; this is however harmless since in this case $\Gamma_1 \leq_D \Gamma_1 \land \Gamma_2$, so that using the admissibility of strengthening to be shown below, one can prove that all the typings obtainable by means of either basis, can be obtained by the other one. A similar remark holds for contexts.

Now we are in place to prove the admissibility of strengthening:

**Lemma 4.8 (Admissibility of Strengthening)** The following rule is admissible:

$$\frac{\Gamma \vdash T : \sigma \mid \Delta}{\Gamma' \vdash T : \sigma \mid \Delta'} (\Gamma' \leq_D \Gamma \& \Delta' \leq_C \Delta)$$

**Proof:** The proof is by induction over the derivation of $\Gamma \vdash T : \sigma \mid \Delta$. The only interesting cases are when it ends by $(Ax), (Abs), (\mu)$ or $(Cmd)$.

$(Ax)$: in this case $T \equiv x$ and $\sigma = \Gamma(x)$, for some $x \in \text{VAR.}$ Then we replace the rule by the inference:

$$\frac{\Gamma' \vdash x : \Gamma'(x) \mid \Delta' \quad (Ax) \quad \Gamma'(x) \leq_D \Gamma(x)}{\Gamma' \vdash x : \Gamma(x) \mid \Delta'}$$

where $\Gamma'(x) \leq_D \Gamma(x)$ follows by the assumption that $\Gamma' \leq_D \Gamma$.  

24
(Abs): in this case we have that $T \equiv \lambda x. M$, $\sigma = \delta \times \kappa \rightarrow \rho$ and the inference ends by

$$
\Gamma, x : \delta \vdash M : \kappa \rightarrow \rho \mid \Delta \Rightarrow \\
\Gamma \vdash \lambda x. M : \delta \times \kappa \rightarrow \rho \mid \Delta \quad (\text{Abs})
$$

where $x \notin \text{dom}(\Gamma)$. Since $x$ is bound in $\lambda x. M$, and hence it can be replaced by any variable which is not in $\text{dom}(\Gamma')$, we assume w.l.o.g. that also $x \notin \text{dom}(\Gamma')$. It follows that $\Gamma', x: \delta$ is a well-formed basis, and clearly $\Gamma' \leq D \Rightarrow \Gamma'$ implies $\Gamma', x: \delta \leq D, \Gamma, x: \delta$. Therefore by induction we have $\Gamma', x: \delta \vdash M : \kappa \rightarrow \rho \mid \Delta'$, from which we conclude that $\Gamma' \vdash \lambda x. M : \delta \times \kappa \rightarrow \rho \mid \Delta'$ by rule (Abs).

($\mu$): this case is similar to that of (Abs), this time reasoning on the contexts $\Delta$ and $\Delta'$.

(Cmd): in this case we have that $T \equiv [a] M$ and $\sigma = \delta \times \kappa$, where $\kappa = \Delta(\alpha)$, and the derivation ends by

$$
\Gamma \vdash M : \delta \mid \Delta \Rightarrow \\
\Gamma \vdash [a] M : \delta \times \kappa \mid \Delta \quad (\text{Cmd})
$$

Since $\Delta'(\alpha) = \kappa' \leq C \kappa$ implies $\delta \times \kappa' \leq C \delta \times \kappa$, and by induction $\Gamma' \vdash M : \delta \mid \Delta'$, we can replace this rule by:

$$
\Gamma' \vdash M : \delta \mid \Delta' \Rightarrow \\
\Gamma' \vdash [a] M : \delta \times \kappa' \mid \Delta' \quad (\text{Cmd})
$$

$$
\Gamma' \vdash [a] M : \delta \times \kappa \mid \Delta' \quad (\leq)
$$

Note that in case $\alpha : \omega \in \Delta$ but $\alpha \notin \text{dom}(\Delta')$ we still would have that $\Delta' \leq C \Delta$, but the instance (Cmd$_1$) of rule (Cmd) would not be applicable.

The following lemma suffices to describe the set of types that can be assigned to a term or a command (Corollary 4.10 below).

**Lemma 4.9** If $D :: \Gamma \vdash T : \sigma \mid \Delta$, then either $\sigma \equiv \omega$ or there exist sub-derivations $D_i :: \Gamma \vdash T_i : \sigma_i \mid \Delta$ of $D$, with $1 \leq i \leq n$, such that $\bigwedge_{i=1}^{n} \sigma_i \leq A \sigma$ and the last rule of each $D_i$ is a type rule.

**Proof:** By straightforward induction over the structure of derivations.

In the following we abbreviate $\Gamma \setminus \{x : \Gamma(x)\}$ by $\Gamma \setminus x$; notice that this correspond to the function update $\Gamma[x := \omega]$. Similarly, we write $\Delta \setminus a$ for $\Delta \setminus \{a : \Delta(\alpha)\} = \Delta[a := \omega]$.

**Corollary 4.10 (Generation Lemma)** Let $\delta \neq D \omega$ and $\kappa \neq C \omega$:

i) $\Gamma \vdash x : \delta \mid \Delta \iff \Gamma(x) \leq D \delta$.

ii) $\Gamma \vdash \lambda x. M : \delta \mid \Delta \iff$

$$
\exists I, \delta_i, \kappa_i, \rho_i \forall i \in I \Gamma, x : \delta_i \vdash M : \kappa_i \rightarrow \rho_i \mid \Delta \land \bigwedge_{i \in I} \delta_i \times \kappa_i \rightarrow \rho_i \leq D \delta].
$$

iii) $\Gamma \vdash MN : \delta \mid \Delta \iff$

$$
\exists I, \delta_i, \kappa_i, \rho_i \forall i \in I \Gamma \vdash M : \delta_i \times \kappa_i \rightarrow \rho_i \mid \Delta \land \Gamma \vdash N : \delta_i \mid \Delta \land \bigwedge_{i \in I} \kappa_i \rightarrow \rho_i \leq D \delta].
$$

iv) $\Gamma \vdash \mu a.C : \delta \mid \Delta \iff$

$$
\exists I, \kappa_i, \rho_i, \kappa_i' \forall i \in I \Gamma \vdash C : (\kappa_i \rightarrow \rho_i) \times \kappa_i \mid \alpha : \kappa_i' \Delta \land \bigwedge_{i \in I} \kappa_i' \rightarrow \rho_i \leq D \delta],
$$

v) $\Gamma \vdash [a] M : \kappa \mid \Delta \iff$

$$
\exists I, \delta_i, \kappa_i \forall i \in I \Gamma \vdash M : \delta_i \mid \Delta \land \Delta(\alpha) \leq C \kappa_i \land \bigwedge_{i \in I} \delta_i \times \kappa_i \leq C \kappa.
$$

Moreover, all the judgements in the right-hand sides are conclusions of sub-derivations of the respective derivations of the left-hand sides.
Proof: The ‘only if’ parts follow by Lemma 4.9, observing that if the judgement \( \Gamma \vdash \top : \sigma \mid \Delta \) is derivable then the last type (i.e. non logical) rules in the derivation are uniquely determined by the main constructor of \( T \).

Vice-versa, if for example we have \( \Gamma \vdash M : \delta_i \mid \Delta \) and \( \Delta(\alpha) \leq_C \kappa_i \) for all \( i \in I \), and \( \land_{i \in I} \delta_i \times \kappa_i \leq_C \kappa \), set \( \Delta' = \Delta \setminus \alpha \); then we can derive \( \Gamma \vdash [\alpha] M : \kappa \mid \Delta \) by:

\[
\begin{align*}
\Gamma & \vdash M : \delta_i \mid \alpha : \kappa_i, \Delta' & (\text{Cmd}) \\
\Gamma & \vdash [\alpha] M : \delta_i \times \kappa_i \mid \alpha : \kappa_i, \Delta & (\text{W}) \\
\Gamma & \vdash [\alpha] M : \land_{i \in I} \delta_i \times \kappa_i \mid \Delta & (\forall i \in I) \\
\Gamma & \vdash [\alpha] M : \kappa \mid \Delta & (\leq)
\end{align*}
\]

The other cases are similar.

### 4.2 Type interpretation and soundness

In this section we will formally define the type interpretation and consequently the interpretation of typing judgements. As anticipated in the informal discussion of the system, the meaning of a type is a subset of the domain of interpretation.

In the subsequent definitions and statements we relate types to a \( \lambda \mu \)-model \( \mathcal{M} = (R, D, C) \) tacitly assuming that that the language \( \mathcal{L}_R \) includes all constants \( v_a \) for \( a \in \mathcal{K}(R) \).

**Definition 4.11 (Type Interpretation)** Let \( \mathcal{M} = (R, D, C) \) be a \( \lambda \mu \)-model. For \( A = R, D, C \) we define the interpretation \( \llbracket : \rrbracket^{\mathcal{M}} : \mathcal{L}_A \to \mathcal{P}(A) \) (written \( \llbracket : \rrbracket^A \) when \( \mathcal{M} \) is understood) as follows:

\[
\begin{align*}
\llbracket v_a \rrbracket^R & = \uparrow_R a = \{ r \in R \mid a \subseteq r \} \\
\llbracket \delta \times \kappa \rrbracket^C & = \llbracket \delta \rrbracket^D \times \llbracket \kappa \rrbracket^C \\
\llbracket \kappa \to \rho \rrbracket^D & = \{ d \in D \mid \forall k \in \llbracket \kappa \rrbracket^C [ d k \in \llbracket \rho \rrbracket^R ] \} \\
\llbracket v_a \rrbracket^D & = \llbracket \omega \to v_a \rrbracket^D = \{ d \in D \mid \forall k \in \mathcal{C} [ d k \in \llbracket v_a \rrbracket^R ] \}
\end{align*}
\]

and

\[
\begin{align*}
\llbracket \omega \rrbracket^A & = A \\
\llbracket \sigma_1 \land \sigma_2 \rrbracket^A & = \llbracket \sigma_1 \rrbracket^A \cap \llbracket \sigma_2 \rrbracket^A
\end{align*}
\]

**Remark 4.12** The last definition is a special case w.r.t. the natural adaptation of the intersection type interpretation as subsets of a \( \lambda \)-model, in that we fix the interpretation of the type constants \( v_a \). This is consistent with the approach of constructing types from the solution of the continuation domain equations, and it is the intended interpretation through all this work. In particular it implies that the language \( \mathcal{L}_R \) depends on the chosen domain of results \( R \), and that the interpretation of a type is always a principal filter of either \( R \), \( D \) or \( C \) according to its kind, as it is proved in the next lemma.

There is no limitation in doing so. If we postulate that there are denumerably many constants \( v_0, v_1, \ldots \) in \( \mathcal{L}_R \), then we can generalize straightforwardly the definition of type interpretation to \( \llbracket \sigma \rrbracket^A_\eta \) relative to the type environment \( \eta \), that is a mapping of type constants such that \( \eta(v_i) \subseteq R \) for all \( i \), then putting \( \llbracket v_i \rrbracket^R_\eta = \eta(v_i) \) in the base case of the inductive definition as usual. Then the above definition is recovered by considering an arbitrary exhaustive enumeration of the compacts \( a_0, a_1, \ldots \) (possibly with repetitions) of \( \mathcal{K}(R) \), existing since \( R \) is \( \omega \)-algebraic, and then taking the type constant interpretation \( \eta_0(v_i) = \uparrow_R a_i \).
There exists a close relation between the interpretation of types and the maps $\Theta_A$ introduced in Definition 3.18 of section 3.3, that is made explicit in the following lemma.

**Lemma 4.13** For $A = R, D, C$ and any $\sigma \in \mathcal{L}_A$, we have that $[[\sigma]]^A = \uparrow_A \Theta_A(\sigma)$.

**Proof:** By induction over $\sigma$ and by cases of $A$.

- $\sigma \equiv \omega$: we have $\Theta_A(\omega) = \bot$ and $\uparrow_A \bot = A = [[\omega]]^A$.
- $\sigma \equiv \sigma_1 \land \sigma_2$: by induction $[[\sigma_i]]^A = \uparrow_A \Theta_A(\sigma_i)$; on the other hand $\Theta_A(\sigma_1 \land \sigma_2) = \Theta_A(\sigma_1) \cup \Theta_A(\sigma_2)$ by Remark 3.19, and we have $\uparrow_A \Theta_A(\sigma_1) \cup \uparrow_A \Theta_A(\sigma_2)$, whence $[[\sigma_1 \land \sigma_2]]^A = [[\sigma_1]]^A \cap [[\sigma_2]]^A = \uparrow_A \Theta_A(\sigma_1 \land \sigma_2)$.
- $\sigma \equiv v_a$: if $A = R$ this follows immediately from the fact that $\Theta_R(v_a) = a$ and $[[v_a]]^R = \uparrow_R a$.
  
  If $A = D$ then $\Theta_D(v_a) = (\bot \Rightarrow a)$; but for any $a \in D = [C \rightarrow R]$ we know that $(\bot \Rightarrow a) \subseteq d$ if and only if $a \subseteq d$ for all $k \in C$, by definition of step functions, that is if and only if $d \in [[v_a]]^D$ for the above; it follows that $\Theta_D(v_a) = d$ if and only if $d \in [[v_a]]^D$ as desired.
- $\sigma \equiv \delta \times k$: then $\Theta_C(\delta \times k) = (\Theta_D(\delta), \Theta_C(k))$ by Remark 3.19, and for any $(d, k) \in C = D \times C$ we have:
  
  $$(\Theta_D(\delta), \Theta_C(k)) \sqsubseteq (d, k) \iff \Theta_D(\delta) \sqsubseteq d & \Theta_C(k) \sqsubseteq k$$
  
  by def. of order over $D \times C$
  
  $d \in [[\delta]]^D \land k \in [[k]]^C$ by ind.
  
  $$(d, k) \in [[\delta]]^D \times [[k]]^C = [[\delta \times k]]^C$$
  
  by Def. 4.11
- $\sigma \equiv \kappa \rightarrow \rho$: then $\Theta_D(\kappa \rightarrow \rho) = (\Theta_C(\kappa) \Rightarrow \Theta_R(\rho))$ by Remark 3.19; for any $a \in D = [C \rightarrow R]$ we have:
  
  $$(\Theta_C(\kappa) \Rightarrow \Theta_R(\rho)) \sqsubseteq d \iff \forall k \in C. \Theta_C(k) \sqsubseteq k \Rightarrow \Theta_R(\rho) \sqsubseteq d \iff \forall k \in C. k \in [[k]]^C \Rightarrow d \in [[\rho]]^R$$
  
  by def. of step functions
  
  $d \in [[\kappa \rightarrow \rho]]^D$ by ind.

**Corollary 4.14** For $A = R, D, C$, if $\sigma, \tau \in \mathcal{L}_A$ then $\sigma \leq_A \tau \iff [[\sigma]]^A \subseteq [[\tau]]^A$.

**Proof:**

$$\sigma \leq_A \tau \iff \Theta_A(\sigma) \supseteq \Theta_A(\tau)$$

by lemmas 3.10((iv)) and 3.20

$$\iff \uparrow_A \Theta_A(\sigma) \subseteq \uparrow_A \Theta_A(\tau)$$

$$\iff [[\sigma]]^A \subseteq [[\tau]]^A$$

by Lemma 4.13

We now define satisfiability typing judgments w.r.t. a $\lambda\mu$-model.

**Definition 4.15 (Satisfiability)** Let $\mathcal{M} = (R, D, C)$ be a $\lambda\mu$-model.

1. $\text{ENV}_\mathcal{M}$ is the set of environments interpreting variables and names into $\mathcal{M}$.

2. We define a notion of semantic satisfiability:

$$e \models_\mathcal{M} \Gamma \vdash \Delta \iff \forall x \in \Gamma(x) \subseteq [[\Gamma(x)]]^\mathcal{M} \land \forall a \in \Delta(a) \subseteq \Gamma(a)$$

$$\Gamma \models_\mathcal{M} M : \delta \vdash \Delta \iff e \models_\mathcal{M} \Gamma \vdash \Delta$$

$$\Gamma \models_\mathcal{M} C : \kappa \vdash \Delta \iff e \models_\mathcal{M} \Gamma \vdash \Delta$$

**Remark 4.16** Continuing the discussion in Remark 4.12, we note that we do not consider here the concept of validity, namely satisfiability w.r.t. any $\lambda\mu$-model $\mathcal{M}$, since we model both the language $\mathcal{L}_R$ and the preorder $\leq_R$ after $R$, which is the particular domain of results of $\mathcal{M}$.

As a matter of fact a definition of validity can be given as follows: first one has to fix the type theory $\mathcal{T}_R$ for the language $\mathcal{L}_R$ with denumerably many type constants $v_i$; then the
satisfiability notion should be relativized to both term and type environments \( e \) and \( \eta \), asking that the latter is a model of the theory \( T_R \), in the sense that whenever \( \rho \leq_R \rho' \), it holds that \( \eta(\rho) \subseteq \eta(\rho') \).

However we disregard such a general formulation, as it involves a useless complication of the theory.

The next result is the soundness of the typing system. Note that, although the construction of the system has been made by having an initial model in mind, the soundness theorem holds for any model.

**Theorem 4.17 (Soundness of type assignment)** Let \( \mathcal{M} \) be a \( \lambda\mu \)-model. If \( \Gamma \vdash T: \sigma \mid \Delta \), then \( \Gamma \models_{\mathcal{M}} T: \sigma \mid \Delta \).

**Proof:** By induction on the structure of derivations. (We drop the super and subscripts on the interpretation function and write \( \models \) for the symbol \( \models_{\mathcal{M}} \).

\((Ax)\): Then \( T \equiv x \) and \( \sigma = \delta \); let \( e \models \Gamma, x: \delta; \Delta \), then \( e \models x \in [\delta] \). Hence, by Definition 2.3, we get \( \llbracket x \rrbracket e \in [\delta] \).

\((Abs)\): Then \( T \equiv \lambda x. \mathcal{N} \) and there exist \( \delta, \kappa \) and \( \rho \) such that \( \sigma = \delta \times \kappa \rightarrow \rho \) and \( \Gamma, x: \delta \models \mathcal{N} : \kappa \rightarrow \rho \mid \Delta \). By definition, \( \llbracket \lambda x. \mathcal{N} \rrbracket e \models k = \llbracket \mathcal{N} \rrbracket e(x \rightarrow d) \mid \kappa, \delta \), where \( k = (d, k') \); also, \( e \models \Gamma; \Delta \) and \( d \in [\delta] \). By definition, for any \( e \models \Gamma; \Delta \) and \( d \in [\delta] \), \( e \models \mathcal{N} \mid \Delta \), so \( \llbracket \mathcal{N} \rrbracket e(x \rightarrow d) \in [\kappa, \rho] \), so \( \llbracket \mathcal{N} \rrbracket e(x \rightarrow d) \in [\rho] \) for any \( k \in [\kappa] \).

So \( \llbracket \lambda x. \mathcal{N} \rrbracket e (d, k) \in [\rho] \), so \( \llbracket \lambda x. \mathcal{N} \rrbracket e \in \langle [\delta] \times [\kappa] \rangle \), and we conclude \( \Gamma \models \lambda x. \mathcal{N} : \delta \times \kappa \rightarrow \rho \mid \Delta \).

\((App)\): Then \( T \equiv \mathcal{P} \mathcal{Q} \) and there exist \( \delta, \kappa \) and \( \rho \) such that \( \sigma = \kappa \rightarrow \rho \), \( \Gamma \vdash \mathcal{P} : \delta \times \kappa \rightarrow \rho \mid \Delta \) and \( \Gamma \vdash \mathcal{Q} : \delta \mid \Delta \). By definition, \( \llbracket \mathcal{P} \mathcal{Q} \rrbracket e \models k = \llbracket \mathcal{P} \rrbracket e \llbracket \mathcal{Q} \rrbracket e, k \). Let \( e \models \Gamma; \Delta \). By induction, \( \Gamma \vdash \mathcal{P} : \delta \times \kappa \rightarrow \rho \mid \Delta \) and \( \Gamma \vdash \mathcal{Q} : \delta \mid \Delta \), so \( \llbracket \mathcal{P} \rrbracket e \in [\delta] \times [\kappa] \rightarrow [\rho] \) and \( \llbracket \mathcal{Q} \rrbracket e \in [\delta] \); in particular, for any \( (d, k') \in [\delta] \times [\kappa] = [\delta] \times [\kappa] \), we have \( \llbracket \mathcal{P} \rrbracket e \models (d, k') \in [\rho] \), so \( \llbracket \mathcal{P} \mathcal{Q} \rrbracket e \models \kappa \rightarrow \rho \mid \Delta \), and thereby \( \Gamma \models \mathcal{P} \mathcal{Q} : \kappa \rightarrow \rho \mid \Delta \).

\((Cmd)\): Then \( T \equiv [\alpha] \mathcal{N} \) and there exist \( \delta \) and \( \kappa = \Delta(\alpha) \) such that \( \sigma = \delta \times \kappa \rightarrow \mid \Delta \). By induction we have that \( \Gamma \models \mathcal{N} : \delta \mid \Delta \), so that for any \( e \models \Gamma; \Delta \) we have that \( \llbracket \mathcal{N} \rrbracket e \in [\delta] \). But \( e \models \Gamma; \Delta \) implies that \( e \alpha \in \llbracket \Delta(\alpha) \rrbracket = [\kappa] \), hence \( \llbracket [\alpha] \mathcal{N} \rrbracket e \models \llbracket [\mathcal{N}] e, e \alpha \rrbracket \in [\delta] \times [\kappa] = [\delta] \times [\kappa] \) as desired. It follows that \( \Gamma \models [\alpha] \mathcal{N} : \delta \times \kappa \rightarrow \mid \Delta \) by the arbitrary choice of \( e \).

\((\mu)\): Then \( T \equiv \mu \alpha. \mathcal{C} \), and there exist \( \kappa, \kappa' \) and \( \rho \) such that \( \sigma = \kappa \rightarrow \rho \) and \( \Gamma \vdash \mathcal{C} : (\kappa' \rightarrow \rho) \times (\kappa \times \kappa). \) By definition, \( \llbracket \mu \alpha. \mathcal{C} \rrbracket e \models k = d \times k', \) where \( (d, k') = [\mathcal{C}] e(\alpha \rightarrow k) \). Let \( e \models \Gamma; \Delta \) and \( k \in [\kappa] \), then \( e \models [\alpha \rightarrow k] \models \Gamma; \alpha : \kappa, \Delta. \) By induction, \( \Gamma \vdash \mathcal{C} : (\kappa' \rightarrow \rho) \times \alpha : \kappa \times \kappa; \) so \( \llbracket \mathcal{C} \rrbracket e(\alpha \rightarrow k) \models [\kappa'] \times [\kappa] \), and \( \pi_1 \rho \in [\kappa'] \) and \( \pi_2 \rho \in [\kappa] \), so \( \llbracket \mu \alpha. \mathcal{C} \rrbracket e \models [\alpha \rightarrow k] \models \Gamma; \alpha : \kappa, \Delta. \)

\((\wedge)\): Easy, by induction and the interpretation of an intersection type.

\((\omega)\): Immediate by definition of interpretation of \( \omega \).


We will now show that we can interpret a term or command by the set of types that can be given to it (Theorem 4.20 below). Towards that result, we make the denotation of terms and commands and the interpretations of types in the filter model \( \mathcal{F} \) explicit.

**Lemma 4.18**

\( i) \llbracket \lambda x. \mathcal{M} \rrbracket e = \uparrow \Delta \{ \wedge_{i \in I} (\delta_i \times \kappa_i \rightarrow \rho_i) \in \mathcal{L}_D \mid \forall i \in I [\kappa_i \rightarrow \rho_i] \in \llbracket \mathcal{M} \rrbracket e(\alpha \rightarrow ([\Delta_i \delta_i])) \}. \)

\( ii) \llbracket \mathcal{M} \mathcal{N} \rrbracket e = \uparrow \Delta \{ \wedge_{i \in I} \kappa_i \rightarrow \rho_i \in \mathcal{L}_D \mid \forall i \in I \exists \delta_i \in \llbracket \mathcal{N} \rrbracket e, \kappa_i \in \llbracket \mathcal{C} \rrbracket [\delta_i \times \kappa_i \rightarrow \rho_i] \in \llbracket \mathcal{M} \rrbracket e \}. \)

\( iii) \llbracket \mu \alpha. \mathcal{C} \rrbracket e = \uparrow \Delta \{ \wedge_{i \in I} \kappa_i \rightarrow \rho_i \in \mathcal{L}_D \mid \forall i \in I \exists \kappa'_i[\kappa'_i \rightarrow \rho_i \times \kappa'_i \in \llbracket \mathcal{C} \rrbracket e(\alpha \rightarrow ([\Delta_i \kappa_i])) \}. \)
Lemma 4.19 For $A \in \mathcal{L}_C$, by part (iv) of this lemma, implies $[[\sigma]]^F = \uparrow_F (\uparrow_A \sigma) = \{ a \in \mathcal{F}_A \mid \uparrow_A \sigma \subseteq a \}$. By the remark above we have that $\{ a \in \mathcal{F}_A \mid \uparrow_A \sigma \subseteq a \} = \{ a \in \mathcal{F}_A \mid \sigma \in a \}$, and we conclude.
The next theorem, together with the Completeness Theorem 4.25 that it implies, is the main result of this section, which states that the set of types that are assigned to terms and commands by the type assignment system coincides with their interpretation in the filter model. Its proof essentially depends on Corollary 4.10 and Lemma 4.18.

**Theorem 4.20**  Let \( A = D, C \). Given an environment \( e \in \text{Envy}_F \),

\[
[[T]]^F e = \{ \sigma \in \mathcal{L}_A \mid \exists \Gamma, \Delta [e \models_F \Gamma; \Delta \& \Gamma \vdash T : \sigma \mid \Delta] \}.
\]

**Proof:** Because of the logical rules, the set \( \{ \sigma \in \mathcal{L}_A \mid \exists \Gamma, \Delta [e \models_F \Gamma; \Delta \& \Gamma \vdash T : \sigma \mid \Delta] \} \) is a filter in \( F_A \), for \( A = D, C \). To prove that this filter coincides with \( [[T]]^F e \) we proceed by induction over the structure of terms.

\((T \equiv x):\) then \( [[x]]^F e = e x\). By definition of \( \models_F, e \models_F \Gamma ; \emptyset \) holds if and only if \( e x \in [[\Gamma (x)]]^F \).

By Lemma 4.19(ii) we know that \( [[\Gamma (x)]]^F = \{ d \in F_D \mid \Gamma (x) \in d \} \), so that \( e \models_F \Gamma ; \emptyset \) is equivalent to \( \Gamma (x) \in e x \). On the other hand, by Lemma 4.10(i), we have that \( \Gamma \vdash T : \delta \mid \Delta \) if and only if \( \Gamma (x) \leq_D \delta \), hence \( \Gamma \vdash x : \delta \mid \Delta \) if and only if \( \delta \in e x \).

\((T \equiv \lambda x. M):\) For \( \delta \in [[\lambda x. M]]^F e \), by Lemma 4.18(ii), there exist \( I \) such that for all \( i \in I \) there exist \( \delta_i, \kappa_i \) and \( \rho_i \) such that \( \kappa_i \rightarrow \rho_i \in [[M]]^F e[x \mapsto \uparrow_D \delta_i] \). By induction, for all \( i \in I \), there exist \( \Gamma_i, \Delta_i \) such that \( e \models \Gamma_i \vdash \lambda x. M \models e \models \Gamma_i \vdash \Delta_i \). Let \( \delta_i' = \Gamma_i(x) \) then by rule \((\lambda b)s\) we have \( \Gamma_i \vdash x \mapsto \lambda x. M : \delta_i' \times \kappa_i \rightarrow \rho_i \mid \Delta_i \). On the other hand, from \( e \models x \mapsto \uparrow_D \delta_i \) we know that \( \Gamma_i(x) = \delta_i' \); since \( \delta_i' \leq_D \delta_i \), also \( \delta_i \leq_D \delta_i' \). Then \( \delta_i \times \kappa_i \leq_C \delta_i \times \kappa_i \) follows by the covariance of \( , \) and \( \delta_i \times \kappa_i \leq_D \delta_i \times \kappa_i \) by the contra-variance of the arrow in its first argument. Hence, by applying rule \((\leq D)\), for all \( i \in I \), we obtain \( \Gamma_i \vdash x \models \lambda x. M : \delta_i \times \kappa_i \rightarrow \rho_i \mid \Delta_i \). Take \( \Gamma = \bigwedge_{i \in I} \Gamma_i \times x \) and \( \Delta = \bigwedge_{i \in I} \Delta_i \); then \( \Gamma \vdash \lambda x. M : \delta_i \times \kappa_i \rightarrow \rho_i \mid \Delta \) for all \( i \in I \) by applying rule \((W)\), and therefore \( \Gamma \vdash \lambda x. M : \bigwedge_{i \in I} \delta_i \times \kappa_i \rightarrow \rho_i \mid \Delta \) by applying rule \((\wedge)\) and \( \Gamma \vdash \lambda x. M : \delta \mid \Delta \) by rule \((\leq)\). Observe that, for all \( i \in I \), \( x \notin \text{dom}(\Gamma_i \times x) \) and consequently \( x \notin \text{dom}(\Gamma) \), so that, for all \( i \in I \), \( e \models x \mapsto \uparrow_D \delta_i \) implies \( e \models \Gamma_i \vdash \Lambda \times \Delta_i \) and so \( e \models \Gamma \vdash \Lambda ; \Delta \) as required.

Vice-versa, if \( \Gamma \vdash \lambda x. M : \delta \mid \Delta \) and \( e \models \Gamma ; \Delta \), then, by Lemma 4.10(ii) there exist \( I \) such that for all \( i \in I \) there exist \( \delta_i, \kappa_i \) and \( \rho_i \) such that \( \Gamma, x \vdash \lambda x. M : \delta_i \times \kappa_i \rightarrow \rho_i \mid \Delta_i \) and \( \bigwedge_{i \in I} \delta_i \times \kappa_i \rightarrow \rho_i \leq_D \delta \). Observe that \( e \models \Gamma ; \Delta \) implies \( e \models x \mapsto \uparrow_D \delta_i \models \Gamma ; \Delta \) for all \( i \in I \). By induction, \( \delta_i \rightarrow \rho_i \in [[M]]^F e[x \mapsto \uparrow_D \delta_i] \), so by Lemma 4.18(ii) \( \bigwedge_{i \in I} \delta_i \times \kappa_i \rightarrow \rho_i \in [[\lambda x. M]]^F e \). Because \( \bigwedge_{i \in I} \delta_i \times \kappa_i \rightarrow \rho_i \leq_D \delta \) and \( [[\lambda x. M]]^F e \) is a filter, we conclude that \( \delta \in [[\lambda x. M]]^F e \).

\((T \equiv M N):\) If \( \delta \in [[M N]]^F e \) then, by Lemma 4.18(ii), there exist \( I \) such that, for every \( i \in I \) there exist \( \kappa_i \) and \( \delta_i \in [[N]]^F e \) such that \( \delta_i \times \kappa_i \rightarrow \rho_i \in [[M N]]^F e \) and \( \bigwedge_{i \in I} \delta_i \rightarrow \rho_i \leq_D \delta \). By induction, for all \( i \in I \) there exist \( \Gamma_i, \Delta_i \) for \( j = 1, 2 \), such that \( e \models \Gamma_i \vdash \Delta_i \). Take \( \Gamma = \bigwedge_{i \in I} \Gamma_i \times \Delta_i \) and \( \bigwedge_{i \in I} \Delta_i \vdash N : \delta_i \mid \Delta_i \). Observe that \( e \models \Gamma \vdash \Delta \) and \( \bigwedge_{i \in I} \delta_i \rightarrow \rho_i \in [[M N]]^F e \) and \( \bigwedge_{i \in I} \delta_i \rightarrow \rho_i \leq_D \delta \). Observe that \( e \models \Gamma \vdash \Delta \) and \( \bigwedge_{i \in I} \delta_i \rightarrow \rho_i \in [[M N]]^F e \) and \( \bigwedge_{i \in I} \delta_i \rightarrow \rho_i \leq_D \delta \). By applying rule \((\wedge)\), for all \( i \in I \) we have \( \Gamma \vdash \Delta_i \rightarrow \rho_i \mid \Delta_i \); by applying rule \((\wedge)\), we have \( \bigwedge_{i \in I} \delta_i \rightarrow \rho_i \in [[M N]]^F e \).

Vice-versa, assume \( \Gamma \vdash MN : \delta \mid \Delta \) and \( e \models \Gamma ; \Delta \). By Lemma 4.10(iii) there exist \( I \) such that, for every \( i \in I \) there exist \( \kappa_i, \rho_i \) such that \( \Gamma \vdash \lambda x. M : \delta_i \times \kappa_i \rightarrow \rho_i \mid \Delta_i \) and \( \bigwedge_{i \in I} \kappa_i \rightarrow \rho_i \leq_D \delta \). Observe that, for all \( i \in I \), by induction we have \( \delta_i \times \kappa_i \rightarrow \rho_i \in [[M N]]^F e \) and \( \bigwedge_{i \in I} \kappa_i \rightarrow \rho_i \leq_D \delta \). By applying rule \((\wedge)\), we have \( \bigwedge_{i \in I} \delta_i \rightarrow \rho_i \in [[M N]]^F e \).

\((T \equiv \mu a. C):\) If \( \delta \in [[\mu a. C]]^F e \) then, by Lemma 4.18(iii), there exist \( I \) such that, for every \( i \in I \) there exist \( \kappa_i, \rho_i \) and \( \kappa_i' \) such that \( (\kappa_i' \rightarrow \rho_i) \times \kappa_i' \in [[C]]^F e[x \mapsto \uparrow_C \kappa_i] \). For all \( i \in I \), by induction there exist \( \Gamma_i, \Delta_i \) such that \( e[a \mapsto \uparrow_C \kappa_i] \models \Gamma_i \vdash \Delta_i \) and \( \Gamma_i \vdash C : (\kappa_i' \rightarrow \rho_i) \times \kappa_i' \mid \Delta_i \). Let \( \Gamma = \bigwedge_{i \in I} \Gamma_i \times \Delta_i \) and, for all \( i \in I \), \( \kappa_i = \Delta_i (a) \) and \( \Delta_i ' = \Delta_i \backslash a \), then \( \Gamma \vdash C : (\kappa_i' \rightarrow \rho_i) \times \kappa_i' \mid a : \kappa_i, \Delta_i '\)
so that $\Gamma \vdash \mu x.\mathcal{C}:\kappa_i \rightarrow \rho_i \mid \Delta_i$. Take $\Delta = \bigwedge_{i \in I} \Delta_i'$; then $\Delta \leq_{C} \Delta_i'$ for all $i \in I$, so by applying rule $(S)$ we obtain $\Gamma \vdash \mu x.\mathcal{C}:\kappa_i \rightarrow \rho_i \mid \Delta$, from which we derive $\Gamma \vdash \mu x.\mathcal{C}:\delta \mid \Delta$ by applying rules $(\land)$ and $(\leq)$. On the other hand, for all $i \in I$, since $\alpha \notin \text{dom}(\Delta_i')$ we have $e[\alpha \mapsto (\uparrow_{C} \kappa_i)] = \vdash_{F} \Gamma; \Delta_i$ which implies $e = \vdash_{F} \Gamma; \Delta_i'$, so that $\Delta(\alpha) = \omega$. We conclude that $e \vdash_{F} \Gamma; \Delta$, as desired.

Vice-versa, assume $\Gamma \vdash \mu x.\mathcal{C}:\delta \mid \Delta$ and $e = \vdash_{F} \Gamma; \Delta$. Then by Lemma 4.10(iv) there exists $I$ such that, for every $i \in I$ there exist $\kappa_i, \rho_i$; and $\kappa'_i$ such that $\Gamma \vdash \mathcal{C} : (\kappa_i \rightarrow \rho_i) \times \kappa_i \mid \alpha : \kappa'_i \Delta_i$, and $\bigwedge_{i \in I} \kappa'_i \rightarrow \rho_i \leq_{D} \delta$. But if $e = \vdash_{F} \Gamma; \Delta$ then $e[\alpha \mapsto (\uparrow_{C} \kappa_i)] = \vdash_{F} \Gamma; \alpha : \kappa'_i \Delta_i$; then by induction, for all $i \in I$, $(\kappa_i \rightarrow \rho_i) \times \kappa_i \in \Gamma^F[e[\alpha \mapsto (\uparrow_{C} \kappa_i)]]$. Since $\bigwedge_{i \in I} \kappa'_i \rightarrow \rho_i \leq_{D} \delta$, by Lemma 4.18(iii) we conclude that $\delta \in \llbracket \mu x.\mathcal{C} \rrbracket^F e$.

(T $\equiv [\alpha]M$ : $e$) : $\kappa \in \llbracket [\alpha]M \rrbracket[e] \text{ then by Lemma 4.18(iv) there exist } I \text{ such that, for every } i \in I \text{ there exist } \delta_i, \kappa_i, \text{ and } \delta_i \in \llbracket M \rrbracket[e] \text{ such that } \kappa_i \in e(\alpha) \text{ and } \bigwedge_{i \in I} \delta_i \times \kappa_i \leq_{C} \kappa. \text{ For all } i \in I, \text{ by induction there exist } \Gamma_i, \Delta_i \text{ such that } e = \vdash_{F} \Gamma_i; \Delta_i \text{ and } \Delta_i(\alpha) \leq_{C} \kappa_i \text{ and } \Gamma_i \vdash M : \delta_i \mid \Delta_i. \text{ Let } \Gamma = \bigwedge_{i \in I} \Gamma_i. \text{Then, for all } i \in I, \Gamma \vdash [\alpha]M : \delta_i \times \kappa_i \mid \Delta_i; \text{ take } \Delta = \bigwedge_{i \in I} \Delta_i, \text{ then for all } i \in I, \text{ by applying rule } (S), \text{ we obtain } \\
\Gamma \vdash [\alpha]M : \bigwedge_{i \in I} \delta_i \times \kappa_i \mid \Delta. \text{ We obtain } \Gamma \vdash [\alpha]M : \bigwedge_{i \in I} \delta_i \times \kappa_i \mid \Delta \text{ by applying rule } (\land) \text{ and then } \Gamma \vdash [\alpha]M : \kappa \mid \Delta \text{ by applying rule } (\leq). \text{ Since } e = \vdash_{F} \Gamma; \Delta_i \text{ and } \Delta \leq_{C} \Delta_i, \text{ for all } i \in I, \text{ we conclude that } e = \vdash_{F} \Gamma; \Delta.$

Vice-versa, if $\Gamma \vdash [\alpha]M \times \kappa \mid \Delta$ and $e = \vdash_{F} \Gamma; \Delta$ then, by Lemma 4.10(v) there exists $I$ such that, for every $i \in I$ there exist $\delta_i$ and $\kappa_i$, such that $\Gamma \vdash M : \delta_i \mid \Delta$ and $\Delta(\alpha) \leq_{C} \kappa_i$ and $\bigwedge_{i \in I} \delta_i \times \kappa_i \leq_{C} \kappa. \text{ By induction } \delta_i \in \llbracket M \rrbracket[e] \text{ from } e = \vdash_{F} \Gamma; \Delta \text{ we have that } \kappa_i \in e(\alpha) \text{ for all } i \in I. \text{ Then } \bigwedge_{i \in I} \delta_i \times \kappa_i \in \llbracket [\alpha]M \rrbracket[e] \text{ and therefore that } \kappa \in \llbracket [\alpha]M \rrbracket[e] \text{ since the last set is a filter.}$

**Definition 4.21** Given a basis $\Gamma$ and a context $\Delta$, we define the environment $e_{\Gamma;\Delta} \in \text{ENV}_F$ by:

\[
\begin{align*}
\uparrow_D \delta & \quad \text{if } x ; \delta \in \Gamma \\
\uparrow_C \omega & \quad \text{otherwise}
\end{align*}
\]

Because of the definition of $\Gamma(x)$ and $\Delta(\alpha)$ for $x \in \text{VAR}$ and $\alpha \in \text{NAME}$, $e_{\Gamma;\Delta} \in \text{ENV}_F$ implies:

\[
\begin{align*}
e_{\Gamma;\Delta}(x) & = \left\{ \begin{array}{ll}
\uparrow_D \delta & \text{if } x ; \delta \in \Gamma \\
\uparrow_C \omega & \text{otherwise}
\end{array} \right.
\end{align*}
\]

\[
\begin{align*}
e_{\Gamma;\Delta}(\alpha) & = \left\{ \begin{array}{ll}
\uparrow_C \delta & \text{if } \alpha \in \Delta \\
\uparrow_C \omega & \text{otherwise}
\end{array} \right.
\end{align*}
\]

For this environment, we can show:

**Lemma 4.22** For any $\Gamma', \Delta'$, if $e_{\Gamma;\Delta} \vdash_{F} \Gamma'; \Delta'$, then $\Gamma \leq_{D} \Gamma'$ and $\Delta \leq_{C} \Delta'$.

**Proof:** By definition, if $e_{\Gamma;\Delta} \vdash_{F} \Gamma'; \Delta'$ then:

\[
\begin{align*}
e_{\Gamma;\Delta}(x) & = \uparrow_D \Gamma(x) \in \llbracket \Gamma'(x) \rrbracket^F \\
e_{\Gamma;\Delta}(\alpha) & = \uparrow_C \Delta(\alpha) \in \llbracket \Delta'(\alpha) \rrbracket^F
\end{align*}
\]

for all $x \in \text{VAR}$ and $\alpha \in \text{NAME}$. From Lemma 4.19(ii) we know that $\uparrow_D \Gamma(x) \in \llbracket \Gamma'(x) \rrbracket^F$ if and only if $\Gamma'(x) \in \uparrow_D \Gamma(x)$, that is $\Gamma(x) \leq_{D} \Gamma'(x)$. Similarly $\uparrow_C \Delta(\alpha) \in \llbracket \Delta'(\alpha) \rrbracket^F$ if and only if $\Delta(\alpha) \leq_{C} \Delta'(\alpha)$. Hence $\Gamma \leq_{D} \Gamma'$ and $\Delta \leq_{C} \Delta'$.

This implies that every type which is an element of the interpretation of a term is derivable for that term:

**Lemma 4.23** If $\delta \in \llbracket M \rrbracket[e_{\Gamma;\Delta}]$, then $\Gamma \vdash M : \delta \mid \Delta$.

**Proof:** $\delta \in \llbracket M \rrbracket[e_{\Gamma;\Delta}] \Rightarrow \exists \Gamma', \Delta' | e_{\Gamma;\Delta} \vdash \Gamma'; \Delta' \& \Gamma' \vdash M : \delta \mid \Delta'$ (Theorem 4.20) \\
$\Rightarrow \exists \Gamma', \Delta' | \Gamma \leq_{D} \Gamma' \& \Delta \leq_{C} \Delta' \& \Gamma' \vdash M : \delta \mid \Delta'$ (Lemma 4.22) \\
$\Rightarrow \Gamma \vdash M : \delta \mid \Delta$ (rule $(S)$, Lemma 8)
Proof: Let $\Delta \leq_c \Delta'$ implies that $\dom(\Delta) \supseteq \dom(\Delta')$ which is not always the case.

We can now prove the completeness theorem for our type assignment system.

**Theorem 4.25 (Completeness)** Let $\mathcal{M} = (R, D, C)$ be a $\lambda \mu$-model. If $\Gamma \vdash \mathcal{M} : \delta | \Delta$, then $\Gamma \vdash M : \delta | \Delta$.

**Proof:** Let $\Gamma \vdash \mathcal{M} : \delta | \Delta$: since $\mathcal{M}$ is isomorphic to the filter model $\mathcal{F} = (\mathcal{F}_R, \mathcal{F}_D, \mathcal{F}_C)$, we have that $\Gamma \models \mathcal{F} : \delta | \Delta$. By construction, $e_{\Gamma; \Delta} \models \mathcal{F}; \Gamma; \Delta$, so that:

\[
\begin{align*}
\Gamma \models \mathcal{F} : \delta | \Delta & \implies [M]^{\mathcal{F}} e_{\Gamma; \Delta} \in [\delta]^{\mathcal{F}} \quad \text{(Definition 4.15)} \\
& \implies \delta \in [M]^{\mathcal{F}} e_{\Gamma; \Delta} \quad \text{(Lemma 4.19(ii))} \\
& \implies \Gamma \vdash M : \delta | \Delta \quad \text{(Lemma 4.23)}
\end{align*}
\]

5 Closure under conversion

In this section, we show that our notion of type assignment is closed under conversion, i.e. it is closed both under subject reduction and expansion. We will first show that this follows from the semantical results we have established in the previous section; then we show the same result via a syntactical proof. The latter is indeed more informative about the structure of derivations in our system; also we establish the term substitution and, more importantly, the structural substitution lemmas (Lemma 5.2 and Lemma 5.3 respectively).

We begin with the abstract proof, which crucially depends on Lemma 4.23 and hence on Theorem 4.20.

**Theorem 5.1 (Closure under conversion)** Let $M =_{\beta \mu} N$. Then $\Gamma \vdash M : \delta | \Delta \implies \Gamma \vdash N : \delta | \Delta$.

**Proof:** By Theorem 2.7, if $M =_{\beta \mu} N$ then $[M]^{\mathcal{M}} e = [N]^{\mathcal{M}} e$ for any model $\mathcal{M}$ and environment $e \in \text{Env}_{\mathcal{M}}$, which holds in particular for $\mathcal{F}$ and $e_{\Gamma; \Delta}$. So

\[
\begin{align*}
\Gamma \vdash M : \delta | \Delta & \implies \Gamma \models \mathcal{F} : M : \delta | \Delta \quad \text{(Theorem 4.17)} \\
& \implies [M]^{\mathcal{F}} e_{\Gamma; \Delta} \in [\delta]^{\mathcal{F}} \quad \text{(since $e_{\Gamma; \Delta} \models \mathcal{F}; \Gamma; \Delta$)} \\
& \implies \delta \in [M]^{\mathcal{F}} e_{\Gamma; \Delta} \quad \text{(Lemma 4.19(ii))} \\
& \implies \delta \in [N]^{\mathcal{F}} e_{\Gamma; \Delta} \quad \text{(Theorem 2.7)} \\
& \implies \Gamma \vdash N : \delta | \Delta \quad \text{(Lemma 4.23)}
\end{align*}
\]

To illustrate the type assignment system itself, we will now show a more ‘operational’ proof for the same property, by studying how reductions and expansions of the term in the conclusion (the “subject” of the typing judgment) are reflected by transformations of its typing derivations. First we show that type assignment is closed for preforming or reversing both the term substitution and the structural substitution. In the next proofs, we will normally assume that, without loss of generality, $\delta = \kappa \rightarrow \rho$; we can do this, because the case $\delta_1 \wedge \delta_2$ is dealt with by splitting up the two cases, and for $\delta = \omega$ the proof becomes trivial.

The next lemma is standard for type assignment systems of the $\lambda$-calculus, and it is proved here by a simple induction over derivations. However we carry out the proof in detail, since in the case of our system there is a deep interplay between the typing of variables and names, that deserves a careful check.

In the statement we assume that $T \in \text{Trm} \cup \text{Cmd}$ and $\sigma \in L_D \cup L_C$. 

32
Lemma 5.2 (Term Substitution Lemma) \( \Gamma \vdash T[L/x] : \sigma \mid \Delta \) with \( x \notin \text{dom}(\Gamma) \) iff there exists \( \delta' \) such that \( \Gamma, x : \delta' \vdash T : \sigma \mid \Delta \) and \( \Gamma \vdash L : \delta' \mid \Delta \).

Proof:

\((T \equiv x)\): \( \Rightarrow \): Then \( x[L/x] \equiv L \) and \( \sigma = \delta \in \mathcal{L}_D \). If \( \Gamma \vdash x[L/x] : \delta \mid \Delta \), then \( \Gamma, x : \delta \vdash x : \delta \mid \Delta \) by \( (Ax) \) and \( \Gamma \vdash L : \delta \mid \Delta \) by hypothesis.

\((T \equiv x)\): \( \Leftarrow \): If \( \Gamma, x : \delta' \vdash x : \delta \mid \Delta \), then \( \delta' \leq \delta \) by Corollary 4.10. From \( \Gamma \vdash L : \delta' \mid \Delta \) and rule \((\leq)\), we have \( \Gamma \vdash L : \delta \mid \Delta \), so also \( \Gamma \vdash x[L/x] : \delta \mid \Delta \).

\((T \equiv y \neq x)\): \( \Rightarrow \): Then \( \sigma = \delta \in \mathcal{L}_D \). By Thinning, since \( y[L/x] \equiv y \), and \( x \notin \text{fv}(y) \).

\((T \equiv y \neq x)\): \( \Leftarrow \): \( \Gamma \vdash y[L/x] : \delta \mid \Delta \Rightarrow \Gamma \vdash y : \delta \mid \Delta \). Take \( \delta' = \omega \); by Weakening, \( \Gamma, x : \omega \vdash y : \delta \mid \Delta \), and \( \Gamma \vdash L : \omega \mid \Delta \) by rule \((\omega)\).

\((T \equiv \lambda y.N)\): Then \( \sigma = \delta \in \mathcal{L}_D \).

\( \exists \delta' \left[ \Gamma, x : \delta' \vdash \lambda y.N : \delta'' \times \kappa \rightarrow \rho \mid \Delta \& \Gamma \vdash L : \delta' \mid \Delta \right] \quad \Leftarrow \quad \text{(by rule (Abs) and Cor. 4.10)}

\( \exists \delta' \left[ \Gamma, x : \delta', y, \delta'' \vdash N : \kappa \rightarrow \rho \mid \Delta \& \& \Gamma \vdash L : \delta' \mid \Delta \right] \quad \Leftarrow \quad \text{(by induction)}

\( \Gamma, y : \delta'' \vdash N[L/x] : \kappa \rightarrow \rho \mid \Delta \quad \Rightarrow \quad \text{(by rule (Abs) and Cor. 4.10)}

\( \Gamma \vdash \lambda y. (N[L/x]) : \delta'' \times \kappa \rightarrow \rho \mid \Delta \quad \Rightarrow \quad \Gamma \vdash (\lambda y.N)[L/x] : \delta'' \times \kappa \rightarrow \rho \mid \Delta \)

\((T \equiv P.Q)\): Then \( \sigma = \delta \in \mathcal{L}_D \). Notice that \((P.Q)[L/x] \equiv P[L/x] \cdot Q[L/x] \).

\( \Rightarrow \): Then \( \sigma = \delta \in \mathcal{L}_D \) and, by Corollary 4.10, there exist \( \delta'', \kappa, \rho \) such that

\( \Gamma \vdash P[L/x] : \delta'' \times \kappa \rightarrow \rho \mid \Delta, \delta = \kappa \rightarrow \rho, \) and \( \Gamma \vdash Q[L/x] : \delta'' \mid \Delta \). Then by induction, there are \( \delta_1, \delta_2 \) such that:

1. \( \Gamma, x : \delta_1 \vdash P : \delta'' \times \kappa \rightarrow \rho \mid \Delta \) and \( \Gamma \vdash L : \delta_1 \mid \Delta \), as well as
2. \( \Gamma, x : \delta_2 \vdash Q : \delta'' \mid \Delta \) and \( \Gamma \vdash L : \delta_2 \mid \Delta \).

Take \( \delta' = \delta_1 \& \delta_2 \); then by strengthening and \((App)\), we get \( \Gamma, x : \delta' \vdash P.Q : \delta \mid \Delta \) and \( \Gamma \vdash L : \delta' \mid \Delta \) by \((\land I)\).

\( \Leftarrow \): If \( \Gamma, x : \delta' \vdash P.Q : \delta \mid \Delta \), then, by Corollary 4.10 there exist \( \delta'', \kappa, \rho \) such that \( \delta = \delta'' \times \kappa \rightarrow \rho, \)

\( \Gamma, x : \delta' \vdash P : \kappa \rightarrow \rho \mid \Delta \) and \( \Gamma, x : \delta' \vdash Q : \delta'' \mid \Delta \). Then, by induction, we have

\( \Gamma \vdash P[L/x] : \delta'' \times \kappa \rightarrow \rho \mid \Delta \) and \( \Gamma \vdash Q[L/x] : \delta'' \mid \Delta \); the result follows by \((App)\).

\((T \equiv \mu \alpha . C)\): Then \( \sigma = \delta \in \mathcal{L}_D \) and \((\mu \alpha . C)[L/x] \equiv \mu \alpha . C[L/x]. \)

By Corollary 4.10, \( \Gamma \vdash \mu \alpha.C[L/x] : \delta \mid \Delta \) iff there exist a finite \( I \) and some \( \kappa_i, \rho_i, \kappa'_i \) such that for all \( i \in I, \Gamma \vdash C[L/x] : (k_i \rightarrow \rho_i) \times \kappa_i \mid \alpha : \kappa'_i, \Delta \) (with shorter derivations) and \( \land_{i \in I} \kappa'_i \rightarrow \rho_i \leq \delta \). Now, for all \( i \in I \) there exists \( \delta_i \) such that:

\( \Gamma, x : \delta_i \vdash C : (k_i \rightarrow \rho_i) \times \kappa_i \mid \alpha : \kappa'_i, \Delta \& \Gamma \vdash L : \delta_i \mid \Delta \) (by induction)

\( \Rightarrow \quad \Gamma, x : \delta_i \vdash \mu \alpha . C : \kappa'_i \rightarrow \rho_i \mid \Delta \& \Gamma \vdash L : \delta_i \mid \Delta \) (by \((\mu)\))

\( \Rightarrow \quad \Gamma, x : \delta' \vdash \mu \alpha . C : \kappa'_i \rightarrow \rho_i \mid \Delta \& \Gamma \vdash L : \delta' \mid \Delta \) (taking \( \delta' = \land_{i \in I} \delta_i \), by \((S)\) and \((\land)\))

\( \Rightarrow \quad \Gamma, x : \delta' \vdash \mu \alpha . C : \delta \mid \Delta \& \Gamma \vdash L : \delta' \mid \Delta \) (by \((\land)\) and \((\leq)\) as \( \land_{i \in I} \kappa'_i \rightarrow \rho_i \leq \delta \))

\((T \equiv [\alpha]N)\): Then \( \sigma = \kappa \in \mathcal{L}_C \) and \(([\alpha]N)[L/x] \equiv [\alpha]N[L/x] \).

By Corollary 4.10, \( \Gamma \vdash [\alpha]N[L/x] : x \mid \Delta \) iff there exist a finite \( I \) and some \( \delta_i, \kappa_i \) such that for all \( i \in I, \Gamma \vdash N[L/x] : \delta_i \mid \Delta \) (with shorter derivations), and \( \Delta(\alpha) \leq C \kappa_i \) for all \( i \in I \) and \( \land_{i \in I} \delta_i \times \kappa_i \leq C \kappa \). As before we have

\( \Gamma, x : \delta'_i \vdash N[L/x] : \delta_i \mid \Delta \& \Gamma \vdash L : \delta'_i \mid \Delta \)

for all \( i \in I \) and certain \( \delta'_i \); hence we conclude by taking \( \delta' = \land_{i \in I} \delta'_i \) and using \((Cmd),(\land)\) and \((S)\).
The next lemma shows how the structural substitution \( T[a \leftrightarrow L] \) is related to the type of the name \( a \). When \( T \in \text{Trm} \) or \( T \equiv [\beta]N \) with \( a \neq \beta \), the type of \( T[a \leftrightarrow L] \) remains the same as that of \( T \), which is similar to the term substitution lemma, but the context \( \Delta \) used to type \( T[a \leftrightarrow L] \) changes to \( \Delta' \) which is equal to \( \Delta \) but for \( \Delta'(a) = \delta' \times \Delta(a) \), where \( \delta' \) is a type of \( L \) (in the same basis and context). However when \( T \equiv [\alpha]N \) the effect of the structural substitution is more complex, and it affects also the type of \( T \) w.r.t. that of \( T[a \leftrightarrow L] \). The fact that this does not invalidate the type invariance of terms w.r.t. structural substitution is due to the form of rule \((\mu)\) that is essentially a cut rule: indeed the “cut type” changes in case of \( T \) w.r.t. that of \( T[a \leftrightarrow L] \), but then it is hidden in the conclusion.

**Lemma 5.3 (Structural Substitution Lemma)** Let \( M,N,L \in \text{Trm} \) and \( \alpha,\beta \in \text{Name} \) with \( a \neq \beta \), and assume that \( a \notin \text{fn}(L) \); then:

i) \( \Gamma \vdash M[a \leftrightarrow L] : \delta \mid a:a,\Delta \) iff exists \( \delta' \) such that \( \Gamma \vdash L : \delta' \mid \Delta \) and \( \Gamma \vdash M : \delta \mid a:a',\Delta \).

ii) \( \Gamma \vdash ([\beta]N)[a \leftrightarrow L] : \kappa' \mid a:a,\Delta \) iff exists \( \delta' \) such that \( \Gamma \vdash L : \delta' \mid \Delta \) and \( \Gamma \vdash [\beta]N : \kappa' \mid a:a',\Delta \).

iii) \( \Gamma \vdash ([\alpha]N)[a \leftrightarrow L] : (\kappa_1 \rightarrow \rho) \times \kappa_2 \mid a:a,\Delta \) iff exists \( \delta' \) such that \( \Gamma \vdash L : \delta' \mid \Delta \) and \( \Gamma \vdash [\alpha]N : (\delta' \times \kappa_1 \rightarrow \rho) \times \kappa_2 \mid a:a',\Delta \).

**Proof:** The proof is by simultaneous induction on derivations. Let us observe that whenever \( a \notin \text{fn}(T) \) for \( T = M \) in part (ii) and \( T = ([\beta]N) \) in part (iii), we have that \( T[a \leftrightarrow L] = T \), so that the lemma is vacuously true by taking \( \delta' = \omega \). This provides the base case of induction. Of the remaining cases we treat just the relevant ones.

(ii): \( \Rightarrow \): assume that the derivation of \( D \) of \( \Gamma \vdash M[a \leftrightarrow L] : \delta \mid a:a,\Delta \) ends by rule \((\mu)\). Therefore \( M = \mu \gamma.C, \delta = \kappa_1 \rightarrow \rho \) and there is a derivation \( D' \) of \( \Gamma \vdash C[a \leftrightarrow L] : (\kappa_2 \rightarrow \rho) \times \kappa_2 \mid \gamma : \kappa_1, a:a, \Delta \), for some \( \kappa_1, \kappa_2, \rho \), that is shorter than \( D \). We now distinguish the two cases:

\((C \equiv [\beta]N)\): Then induction hypothesis (iii) applies, so that we have that for some \( \delta' \) it holds that \( \Gamma \vdash L : \delta' \mid \gamma : \kappa_1, \Delta \) and \( \Gamma \vdash [\beta]N : (\kappa_2 \rightarrow \rho) \times \kappa_2 \mid \gamma : \kappa_1, a:a', \kappa_1, \delta' \times \kappa, \Delta \). Then, by \((\mu)\) we get \( \Gamma \vdash \mu \gamma.[\beta]N : \kappa_1 \rightarrow \rho \mid a:a', \kappa_1, \delta' \times \kappa, \Delta \). On the other hand, since \( \gamma \) is bound in \( \mu \gamma.[\beta]N \), we can freely assume that \( \gamma \notin \text{fn}(L) \), so that from \( \Gamma \vdash L : \delta' \mid \gamma : \kappa_1, \Delta \) we get \( \Gamma \vdash L : \delta' \mid \Delta \) by Thinning, and we are done.

\((C \equiv [\alpha]N)\): In this case induction hypothesis (iii) applies, so that we have that for some \( \delta' \) it holds that \( \Gamma \vdash L : \delta' \mid \gamma : \kappa_1, \Delta \) and \( \Gamma \vdash [\alpha]N : (\delta' \times \kappa_1 \rightarrow \rho) \times (\delta' \times \kappa_2) \mid \gamma : \kappa_1, a:a', \delta' \times \kappa_1, \kappa_2, \Delta \), since in this case \( \kappa_2 = \kappa \). From this we get \( \Gamma \vdash \mu \gamma.[\alpha]N : \kappa_1 \rightarrow \rho \mid a:a', \delta' \times \kappa_1, \kappa_2, \Delta \) by rule \((\mu)\), and \( \Gamma \vdash L : \delta' \mid \Delta \) by Thinning.

\(\Leftarrow\): we are given a derivation of \( \Gamma \vdash L : \delta' \mid \Delta \) and a derivation \( D \) of \( \Gamma \vdash \mu \gamma.C : \kappa_1 \rightarrow \rho \mid a:a', \kappa_1, \delta' \times \kappa, \Delta \) ending by rule \((\mu)\). Again we distinguish the cases of \( C \). Suppose that \( C \equiv [\alpha]N \). Since rule \((\mu)\) has the premise \( \Gamma \vdash [\alpha]N : (\kappa_2 \rightarrow \rho) \times \kappa_2 \mid \gamma : \kappa_1, a:a', \delta' \times \kappa_1, \kappa_2, \Delta \) where \( \kappa_2 \) is forced to be \( \delta' \times \kappa_1 \), and its derivation is shorter than \( D \), induction hypothesis ((ii) applies, and we conclude. The case \( C \equiv [\beta]N \) is similar and easier.

(iii): In this case we have \( ([\beta]N)[a \leftrightarrow L] = [\beta](N[a \leftrightarrow L]) \). Then we reason by induction over the derivation of \( \Gamma \vdash [\beta](N[a \leftrightarrow L]) : \kappa' \mid a:a, \Delta \) (\( \Rightarrow \) part) and of \( \Gamma \vdash [\beta]N : \kappa' \mid a:a', \delta' \times \kappa, \Delta \) (\( \Leftarrow \) part), using induction hypothesis ((ii) as soon as the subject of the conclusion of the considered subderivations is \( N \).

(iii): \( \Rightarrow \): First observe that \( ([\alpha]N)[a \leftrightarrow L] = [\alpha](N[a \leftrightarrow L])L. \) Now assume that the derivation \( D \) of \( \Gamma \vdash [\alpha](N[a \leftrightarrow L])L : (\kappa_1 \rightarrow \rho) \times \kappa_2 \mid a:a, \Delta \) ends by \((Cmd)\) (more precisely an instance of \((Cmd_1)\)).

The premise of the last rule is then \( \Gamma \vdash (N[a \leftrightarrow L])L : \kappa_1 \rightarrow \rho \mid a:a, \Delta \), that is the conclusion of the sub derivation \( D' \) of \( \Delta \). By Corollary 4.10 there exists a finite \( I \) such that for all \( i \in I \) there exists a derivation \( D_i' :: \Gamma \vdash N[a \leftrightarrow L] : \delta_i \times \kappa_{i1} \rightarrow \rho_i \mid a:a, \Delta \) and
Proof: Theorem 5.4

We will now show that types are preserved under expansion, the opposite of reduction. Let us assume that \( I \) is a singleton set and that \( \land \in \Gamma \rightarrow \rho \) is just \( \kappa \rightarrow \rho \). Then we have the sub derivation of \( D' \):

\[
\Gamma \vdash N[\alpha \leftarrow L]: \delta \times \kappa \rightarrow \rho \mid \alpha: \kappa, \Delta \\
\Gamma \vdash L: \delta \mid \alpha: \kappa, \Delta \\
\Gamma \vdash (N[\alpha \leftarrow L])L: \kappa \rightarrow \rho \mid \alpha: \kappa, \Delta
\]

By induction hypothesis ((i) we know that for some \( \delta_2 \) we have \( \Gamma \vdash L: \delta_2 \mid \Delta \) and \( \Gamma \vdash N: \delta_1 \times \kappa \rightarrow \rho \mid \alpha: \delta_2 \times \kappa, \Delta \) are derivable; on the other hand by \( \alpha \notin \text{fn}(L) \) it follows that \( \Gamma \vdash L: \delta_1 \mid \Delta \) is derivable by Thinning, so that we conclude that \( \Gamma \vdash L: \delta_1 \land \delta_2 \mid \Delta \) by (\land\). Taking \( \delta' = \delta_1 \land \delta_2 \), it holds

\[
\delta' \times \kappa_2 \leq C \delta_2 \times \kappa_2 \quad \text{and} \quad \delta_1 \times \kappa_1 \rightarrow \rho \leq D \delta' \times \kappa_1 \rightarrow \rho.
\]

It follows that

\[
\Gamma \vdash N: \delta_1 \times \kappa_1 \rightarrow \rho \mid \alpha: \delta_2 \times \kappa_2, \Delta \\
\Gamma \vdash N': \delta' \times \kappa_1 \rightarrow \rho \mid \alpha: \delta' \times \kappa_2, \Delta \\
\Gamma \vdash [\alpha]: N: (\delta' \times \kappa_1 \rightarrow \rho) \times \delta' \times \kappa_2 \mid \alpha: \delta' \times \kappa_2, \Delta
\]

is the desired derivation. The general case, when \( I \) has more than one index and \( \land \in \Gamma \rightarrow \rho \), is a proper inequality can be proved as before by several uses of (\land\), (\leq\) and (\leq\).

\[
\Leftrightarrow: \text{assume that } D :: \Gamma \vdash [\alpha]: N: (\delta' \times \kappa_1 \rightarrow \rho) \times \delta' \times \kappa_2 \mid \alpha: \delta' \times \kappa_2, \Delta \text{ is a derivation ending by rule } (Cmd); \text{ then there exists the immediate subderivation } D' :: \Gamma \vdash N: \delta' \times \kappa_1 \rightarrow \rho \mid \alpha: \delta' \times \kappa_2, \Delta \text{ of } D. \text{ By this and the hypothesis that } \Gamma \vdash L: \delta' \mid \Delta \text{ we may apply induction hypothesis } (i), \text{ getting that } \Gamma \vdash N[\alpha \leftarrow L]: \delta' \times \kappa_1 \rightarrow \rho \mid \alpha: \kappa, \Delta \text{ is derivable. Since } \alpha \notin \text{fn}(L) \text{ and (as we can assume) } \alpha \notin \text{dom}(\Delta), \text{ we have } \Gamma \vdash L: \delta' \mid \alpha: \kappa, \Delta \text{ by } (W), \text{ and we can build the following derivation:}
\]

\[
\Gamma \vdash N[\alpha \leftarrow L]: \delta' \times \kappa_1 \rightarrow \rho \mid \alpha: \kappa_2, \Delta \\
\Gamma \vdash L: \delta' \mid \alpha: \kappa_2, \Delta \\
\Gamma \vdash (N[\alpha \leftarrow L])L: \kappa \rightarrow \rho \mid \alpha: \kappa_2, \Delta
\]

We will now show that types are preserved under expansion, the opposite of reduction.

**Theorem 5.4 (Subject expansion)** If \( M \rightarrow N \), and \( \Gamma \vdash N: \delta \mid \Delta \), then \( \Gamma \vdash M: \delta \mid \Delta \).

**Proof:** By induction on the definition of reduction, where we focus on the rules.

\[
(\lambda x.M)N \rightarrow M[N/x]: \text{if } \Gamma, x: \beta \vdash M[N/x]: \delta \mid \Delta, \text{ then by Lemma 5.2 there exists a } \delta' \text{ such that } \Gamma, x: \beta' \vdash M: \delta \mid \Delta \text{ and } \Gamma \vdash N: \delta' \mid \Delta; \text{ assume (without loss of generality) that } \delta = \kappa \rightarrow \rho, \text{ then, by applying rule } (Abs) \text{ to the first result we get } \Gamma \vdash \lambda x.M: \delta' \times \kappa \rightarrow \rho \mid \Delta \text{ and by } (App) \text{ we get } \Gamma \vdash \lambda x.M: \delta \mid \Delta.
\]

\[
(\mu \alpha.C)N \rightarrow \mu \alpha.C[\alpha \leftarrow N]: \text{if } \Gamma \vdash \mu \alpha.C[\alpha \leftarrow L]: \delta \mid \Delta, \text{ then (without loss of generality) } \delta = \kappa \rightarrow \rho, \text{ and by Lemma 4.10, there exists } \kappa' \text{ such that } \Gamma \vdash C[\alpha \leftarrow N]: (\kappa' \rightarrow \rho) \times \kappa' \mid \alpha: \kappa, \Delta.
\]

We now distinguish the following cases of \( C \):
(C ≡ [β]L), with β ≠ α: Then, by Lemma 5.3((ii), there exists δ' such that Γ ⊢ N: δ' | Δ, and Γ ⊢ [β]L: (κ' → ρ) × δ' | α:δ' × κ,Δ. Then, by rule (μ), Γ ⊢ μα.[β]L: δ' × κ → ρ | Δ, and Γ ⊢ (μα.[β]L)N : κ → ρ | Δ follows by rule (App).

(C ≡ [α]N): Then, by Lemma 5.3((iii), there exists δ' such that Γ ⊢ N: δ' | Δ, and D := Γ ⊢ [α]L: (δ' × κ → ρ) × δ' × κ | α:δ' × κ,Δ. Assume w.l.o.g. that the last rule of D is (Cmd) (the general case being reducible to this one by means of (λ), (≤) and (S)) then the immediate subderivation of D has conclusion Γ ⊢ L: δ' × κ → ρ | α:δ' × κ,Δ. Since α is bound in μα.[α]L we can assume that α ∉ dom(Δ), so that we can build the following derivation:

\[
\frac{Γ ⊢ L : δ' × κ → ρ | α:δ' × κ,Δ}{Γ ⊢ [α]L : (δ' × κ → ρ) × (δ' × κ) | α:δ' × κ,Δ} \quad \text{(Cmd)}
\]

\[
\frac{Γ ⊢ μα.[α]L : δ' × κ → ρ | Δ}{Γ ⊢ (μα.[α]L)N : κ → ρ | Δ} \quad \text{(App)}
\]

\[
\muα[β]γ.[α']M → μα.[α']M[β/γ] : \text{If } Γ ⊢ μα.[α']M[β/γ] : δ | Δ, \text{ then by Lemma 4.10 there exist } κ_1, κ_2, κ_3 \text{ such that } δ = κ_2 → ρ, \text{ and } Γ ⊢ M[β/γ] : κ_2 → ρ | α:κ_1, α':κ_2, β:κ_3, Δ.
\]

Since M can contain β as well, this means that there are κ^1, κ^2 with κ_3 = κ^1 ∧ κ^2, such that κ^1 is an intersection of the types used for the ‘original’ β, and κ^2 for those inserted by the substitution. Then we have Γ ⊢ M : κ | κ_2 → ρ | κ_1, α':κ_2, β:κ_3, γ:κ_3, Δ as well, and, by weakening, also Γ ⊢ M : κ_2 → ρ | α:κ_1, α':κ_2, β:κ_3, γ:κ_3, Δ. We can now derive:

\[
\frac{Γ ⊢ M : κ_2 → ρ | α:κ_1, α':κ_2, β:κ_3, γ:κ_3, Δ}{Γ ⊢ [α']M : (κ_2 → ρ) × κ_2 | α:κ_1, α':κ_2, β:κ_3, γ:κ_3, Δ} \quad \text{(Cmd)}
\]

\[
\frac{Γ ⊢ [α']M : (κ_2 → ρ) × κ_2 | α:κ_1, α':κ_2, β:κ_3, γ:κ_3, Δ}{Γ ⊢ μγ.[α']M : κ_3 → ρ | α:κ_1, α':κ_2, β:κ_3, γ:κ_3, Δ} \quad \text{(Cmd)}
\]

\[
\frac{Γ ⊢ μγ.[α']M : κ_3 → ρ | α:κ_1, α':κ_2, β:κ_3, γ:κ_3, Δ}{Γ ⊢ μα.[β]μγ.[α']M : κ_1 → ρ | α':κ_2, β:κ_3, Δ} \quad \text{(μ)}
\]

which establishes the result.

We will now prove the counterpart of the previous theorem, and show that types are preserved under reduction.

**Theorem 5.5 (Subject Reduction)** If M → N, and Γ ⊢ M : δ | Δ, then Γ ⊢ N : δ | Δ

**Proof:** By considering the three reduction rules.

(λα.M)N → M[N/x]: Assume (without loss of generality) that δ = κ → ρ. Then by Lemma 4.10 there exists δ' such that Γ ⊢ λα.M : δ' × κ → ρ | Δ and also Γ ⊢ N : δ' | Δ; from the first, by the same lemma, also Γ ⊢ x:δ' ⊢ M : κ → ρ | Δ. Then, by Lemma 5.2, we have Γ ⊢ M[N/x] : κ → ρ | Δ.

(μα.c)N → μα.c[α ⇐ N]: Assume (without loss of generality) that δ = κ → ρ. Then by Lemma 4.10 there exist δ' such that Γ ⊢ μα.c : δ' × κ → ρ | Δ and Γ ⊢ N : δ' | Δ, and, by the same lemma, from the first there exists k' such that Γ ⊢ c : (κ' → ρ) × κ' | α:δ' × κ,Δ. As in the proof of Theorem 5.4, we must distinguish the two possible cases of c:
(C ⊨ [β]L, with β ≠ α): Then, by Lemma 5.3(ii), \( \Gamma \vdash [\beta]L[\alpha \leftarrow N]: (\kappa' \times \rho) \times \kappa' \mid \alpha : \kappa, \Delta \), and \( \Gamma \vdash [\mu a.[\beta]L[\alpha \leftarrow N]: \kappa \rightarrow \rho \mid \Delta \) follows by rule \((\mu)\).

\((C \equiv [\alpha]L)\): Assume w.l.o.g. that the last rule in the derivation of 
\( \Gamma \vdash [\alpha]L:(\kappa' \times \rho) \times \kappa' \mid \alpha: \delta \times \kappa, \Delta \) is \((\text{Cmd})\). Then \( \kappa' = \delta' \times \kappa \), so that by Lemma 5.3(iii), we have that 
\( \Gamma \vdash ([\alpha]L)[\alpha \leftarrow N] \equiv [\alpha](L[\alpha \leftarrow N])N : (\kappa \rightarrow \rho) \times \kappa \mid \alpha : \kappa, \Delta \) from which we deduce 
\( \Gamma \vdash [\mu a.(\alpha]L)[\alpha \leftarrow N] : \kappa \rightarrow \rho \mid \Delta \) by \((\mu)\).

\(\mu a.[\beta]M \rightarrow [\mu a.[\alpha'](M[\beta/\gamma])\): If \( \Gamma \vdash [\mu a.[\alpha'][\mu \gamma.[\alpha'(M[\beta/\gamma])\) then by Lemma 4.10 there exist \( \rho, \kappa_1, \kappa_2, \kappa_3 \) such that \( \Gamma \vdash \rho : \kappa \rightarrow \rho \mid \alpha : \kappa_1, \alpha': \kappa_2, \beta : \kappa_3, \gamma : \kappa_3, \Delta \) and \( \delta = \kappa \rightarrow \rho \). Then, obviously, also \( \Gamma \vdash \rho : \kappa' \rightarrow \rho \mid \alpha : \kappa_1, \alpha': \kappa_2, \beta : \kappa_3, \Delta \), and applying rule \((\times)\) and \((\mu)\) to this derivation gives \( \Gamma \vdash [\mu a.[\alpha'](M[\beta/\gamma])\) \(\delta \mid \alpha : \kappa_2, \beta : \kappa_3, \Delta\).

We end this section by two examples.

**Example 5.6** As stated by the last results, we now show that we can assign to \((\lambda xyz.xz(yz))(\lambda ab.a)\) any type that is assignable to \(\lambda ba.a\) since \((\lambda xyz.xz(yz))(\lambda ab.a) \rightarrow^* \lambda ba.a\). We first derive a type for \(\lambda ba.a\).

\[
\begin{align*}
\vdash \lambda ba.a : \omega \times (\kappa \rightarrow \rho) \rightarrow \kappa \rightarrow \rho \\
\end{align*}
\]

Let \( \Gamma = x: (\kappa \rightarrow \rho) \times \omega \times \kappa \rightarrow \rho, y: \omega, z: \kappa \rightarrow \rho \), then we can derive:

\[
\begin{align*}
\vdash x : (\kappa \rightarrow \rho) \times \omega \times \kappa \rightarrow \rho & \quad \text{(Ax)} \\
\vdash xz : \omega \times \kappa \rightarrow \rho & \quad \text{(Ax)} \\
\vdash yz : \omega & \quad \text{(ω)} \\
\vdash \lambda x : \kappa \rightarrow \rho & \quad \text{(App)} \\
\vdash \lambda y : \omega \times (\kappa \rightarrow \rho) \rightarrow \kappa \rightarrow \rho & \quad \text{(App)} \\
\vdash \lambda x y : \omega \times (\kappa \rightarrow \rho) \times \omega \times (\kappa \rightarrow \rho) \times \kappa \rightarrow \rho & \quad \text{(App)} \\
\vdash (\lambda x y z)(\kappa \rightarrow \rho) \times \omega \times \kappa \rightarrow \rho & \quad \text{(App)} \\
\end{align*}
\]

**Example 5.7** Consider the reduction \((\mu a.[\alpha]x) x \rightarrow \mu a.[\alpha](x)[x \leftarrow x] \equiv [\mu a.[\alpha]xx\), where the last term is not a proof term in the sense of Parigot, but it is interesting here since typing the self application \(xx\) is a characteristic of intersection type systems. Then we type both terms by \(\kappa \rightarrow \rho\), for any \(\kappa\) and \(\rho\), as follows. Let \(\delta \in \mathcal{L}_\Delta\) be arbitrary. First observe that \(x: \delta \times (\delta \times \kappa \rightarrow v) \vdash x : \delta \mid x: \delta \times (\delta \times \kappa \rightarrow v) \vdash x : \delta \times \kappa \rightarrow \rho \mid \delta \times \kappa \rightarrow \rho \) can be deduced using \((Ax)\) followed by \((\leq)\), and from the latter we get \(sx: \delta \times (\delta \times \kappa \rightarrow v) \vdash x : \delta \times \kappa \rightarrow v \mid \alpha : \delta \times \kappa \) by \((W)\). Now we have:

\[
\begin{align*}
\vdash x : \delta \times (\delta \times \kappa \rightarrow \rho) \vdash x : \delta \times \kappa \rightarrow \rho & \quad \text{(Cmd)} \\
\vdash x : \delta \times (\delta \times \kappa \rightarrow \rho) \vdash x : \delta \times \kappa \rightarrow \rho & \quad \text{(App)} \\
\end{align*}
\]
and, by deriving \( x: \delta \wedge (\delta \times \kappa \rightarrow \nu) \vdash x : \delta \mid a : \kappa \) from \( x: \delta \wedge (\delta \times \kappa \rightarrow \nu) \vdash x : \delta \) by \((W)\):

\[
\frac{x: \delta \wedge (\delta \times \kappa \rightarrow \rho) \vdash x : \delta \times \kappa \rightarrow \rho \mid a : \kappa \quad x: \delta \wedge (\delta \times \kappa \rightarrow \nu) \vdash x : \delta \mid a : \kappa}{x: \delta \wedge (\delta \times \kappa \rightarrow \rho) \vdash xx : \kappa \rightarrow \rho \mid a : \kappa} \quad \text{(App)}\\
\frac{x: \delta \wedge (\delta \times \kappa \rightarrow \rho) \vdash [a] xx : (\kappa \rightarrow \rho) \times \kappa \mid a : \kappa}{x: \delta \wedge (\delta \times \kappa \rightarrow \rho) \vdash \mu a. [a] xx : \kappa \rightarrow \rho} \quad \text{(\(\mu\))}
\]

Observe that the “cut type” in the first derivation, which is \(\delta \times \kappa\) appearing twice in the type \((\delta \times \kappa \rightarrow \rho) \times (\delta \times \kappa)\) of the premise of rule \((\mu)\), is different than the cut type \(\kappa\) in \((\kappa \rightarrow \rho) \times \kappa\) occurring in the premise of \((\mu)\) of the second derivation; indeed the latter is of a smaller size than the former, as it happens in a cut elimination procedure.

A similar but simpler derivation can be obtained from the previous one in the case of the reduction \((\mu a. [a] x) y \rightarrow \mu a. ([a] x) [a \leftarrow y] \equiv \mu a. [a] xy\), where there is no self-application: indeed we have that \(x: \delta \times \kappa \rightarrow \rho, y: \delta \vdash (\mu a. [a] x) y : \kappa \rightarrow \rho\) and \(x: \delta \times \kappa \rightarrow \rho, y: \delta \vdash \mu a. [a] xy : \kappa \rightarrow \rho\) are derivable in a very similar manner.

## 6 Characterisation of Strong Normalisation

One of the main results for \(\lambda \mu\), proved in [44], states that all \(\lambda \mu\)-terms that correspond to proofs of second-order natural deduction are strongly normalising; the reverse of this property does not hold for Parigot’s system, since there, for example, not all terms in normal form are typeable.

The full characterisation of strong normalisation (\(M\) is strongly normalising if and only if \(M\) is typeable in a given system) for the \(\lambda\)-calculus is a property that is shown for various intersection systems (see [33], §17.2 and the references there). So it is a natural question whether there exists a similar characterization of strongly normalising \(\lambda \mu\)-terms in the present context, by suitably restricting the system in section 4. We answer this question here; the proof is a revised version of that one in [11], obtained by a simplified type syntax and just by restricting the full system, instead of considering one of its variants.

We relate our result to that one for the original Parigot’s system in the next section.

### 6.1 The restricted type system

In the case of untyped \(\lambda\)-calculus, the characterisation of strong normalisation result states that a \(\lambda\)-term is strongly normalisable if and only if it is typeable in a restricted system of intersection types, where \(\omega\) is not admitted as a type and consequently the rule \((\omega)\) is not part of the system. Alas a straightforward extension of this result does not hold for the \(\lambda \mu\)-calculus, at least with respect to the system presented in this paper. This is due to the fact that the natural interpretation of a type \(\kappa = \delta_1 \times \cdots \delta_k \times \omega\) (for \(k > 0\)) is the set of continuations whose leading \(k\) elements are in the denotations of \(\delta_1, \ldots, \delta_k\); but since continuations are infinite tuples, the ending \(\omega\) represents the lack of information about the remaining infinite part. Thus such occurrences of \(\omega\) cannot be simply deleted without substantially changing the semantics of the type system and questioning the soundness of its rules.

We solve the problem of restricting the type assignment system to the extent of typing strongly normalising terms by defining a particular subset of the type language. In it the type \(\omega\) is allowed only in certain harmless positions such that its meaning becomes just the universe of terms we are looking for, i.e. the strongly normalising ones. This amounts to restricting the sets of types \(\mathcal{L}_D\) and \(\mathcal{L}_C\) to those having \(\omega\) only as the final part of product types. We shall then suitably modify the standard interpretation of intersection types adapting Tait’s computability argument.
For what concerns the atomic types, a single constant type \( \nu \) suffices for our purposes. Therefore restricted types are of two sorts instead of three.

**Definition 6.1 (Restricted Types and Pre-order)**

i) The sets \( \mathcal{L}_D^r \) of (restricted) term types and \( \mathcal{L}_C^r \) of (restricted) continuation types are defined inductively by the following grammars, where \( \nu \) is a type constant:

\[
\begin{align*}
\mathcal{L}_D^r & : \quad \delta ::= \kappa \rightarrow \nu \mid \delta \land \delta \\
\mathcal{L}_C^r & : \quad \kappa ::= \omega \mid \delta \times \kappa \mid \kappa \times \kappa 
\end{align*}
\]

ii) We define the set \( \mathcal{L}_r \) of (restricted) types as \( \mathcal{L}_r = \mathcal{L}_D^r \cup \mathcal{L}_C^r \) and the relations \( \leq_A \) over \( \mathcal{L}_r^A \) (for \( A = D, C \)) as the pre-order induced by the least intersection type theories \( \mathcal{T}_A^r \) such that:

\[
\begin{align*}
\kappa \leq_C \omega & \quad \sigma \land \tau \leq_A \sigma \land \tau \quad \sigma \leq_A \tau \land \tau \quad A = D, C. \\
(\delta_1 \times \delta_2) \land (\delta_2 \times \delta_2) & \leq_C (\delta_1 \land \delta_2) \times (\kappa_1 \land \kappa_2) \\
\delta_1 \leq_D \delta_2 & \quad \kappa_1 \leq_C \kappa_2 \\
\delta_1 \times \kappa_1 & \leq_C \delta_2 \times \kappa_2 \\
\kappa_2 \leq_C \kappa_1 & \quad \kappa_1 \rightarrow \nu \leq_D \kappa_2 \rightarrow \nu
\end{align*}
\]

By definition, we have that \( \mathcal{L}_D^r \subseteq \mathcal{L}_D \) and \( \mathcal{L}_C^r \subseteq \mathcal{L}_C \). All the rules axiomatising \( \leq_A \) are instances of the rules axiomatising \( \leq_D \) and \( \leq_C \) in Definitions 3.9 and 3.13, hence \( \leq_A \subseteq \leq_A \) for \( A = D, C \); otherwise stated the theories \( \mathcal{T}_A \) can be seen as extensions of the respective theories \( \mathcal{T}_A^r \). It is natural to ask whether \( \mathcal{T}_A \) (for \( A = D, C \)) is conservative w.r.t. \( \mathcal{T}_A^r \), which is not obvious: in a derivation of \( \sigma \leq_A \tau \) in the formal theory \( \mathcal{T}_A \) even if \( \sigma, \tau \in \mathcal{L}_A^r \), one could have used a type \( \sigma' \notin \mathcal{L}_A^r \) and the transitivity rule with premises \( \sigma \leq A \sigma' \leq A \tau \), which cannot be derived in the formal presentation of \( \leq_A \). For example consider the inequalities:

\[
\delta_1 \times \delta_2 \times \omega \leq_C \delta_1 \times \omega \times \omega = C \delta_1 \times \omega,
\]

where the axiom \( \omega \times \omega = C \omega \) is used (see Definition 3.13). If \( \delta_1, \delta_2 \in \mathcal{L}_D^r \) then both \( \delta_1 \times \delta_2 \times \omega \) and \( \delta_1 \times \omega \times \omega \) are in \( \mathcal{L}_C^r \), but \( \delta_1 \times \omega \times \omega \notin \mathcal{L}_C^r \) because \( \omega \notin \mathcal{L}_D^r \).

However this is not a counterexample, since \( \delta_2 \times \omega \leq^r \omega \) which implies that \( \delta_1 \times \delta_2 \times \omega \leq^r \delta_1 \times \omega \) is derivable in \( \mathcal{T}_C^r \). As a matter of fact, we can show that \( \sigma \leq^r \tau \) if and only if \( \sigma \leq \tau \) for any \( \sigma, \tau \in \mathcal{L}_r^A \) by a semantic argument.

**Lemma 6.2** Take \( R = \{ \succeq \subseteq \top \} \) and set \( \Theta_R(\nu) = \top \). Then for all \( \sigma, \tau \in \mathcal{L}_A^r \), where \( A = D, C \), if \( \Theta_A(\tau) \subseteq \Theta_A(\sigma) \) then \( \sigma \leq_A \tau \).

**Proof:** By induction over the structure of \( \sigma \) and \( \tau \).

(\( \sigma = \delta, \tau = \delta' \in \mathcal{L}_D^r \)): then \( \delta = \wedge_{i \in I}(k_i \rightarrow \nu) \) and \( \delta' = \wedge_{i \in I}(k'_i \rightarrow \nu) \); hence we have:

\[
\begin{align*}
\Theta_D(\delta') = \cup_{i \in I}(\Theta_D(k'_i) \Rightarrow \top) & \subseteq \cup_{i \in I}(\Theta_D(k_i) \Rightarrow \top) = \Theta_D(\delta) \quad \Rightarrow \quad (\text{by hypothesis}) \\
\forall j \in I \exists i_j \in I \left[ \Theta_C(k'_i) \subseteq \Theta_C(k_i) \right] & \Rightarrow \quad (\text{by induction}) \\
\forall j \in I \exists i_j \in I \left[ k'_i \leq_C k_i \right] & \Rightarrow \quad (\text{by induction}) \\
\forall j \in I \exists i_j \in I \left[ k_i \rightarrow \nu \leq_D k'_i \rightarrow \nu \right] & \Rightarrow \quad (\text{by hypothesis}) \\
\delta \leq^r \wedge_{i \in I} k_i \rightarrow \nu & \leq_D \wedge_{i \in I} (k'_i \rightarrow \nu) = \delta'
\end{align*}
\]

where (*) follows by contraposition: if for some \( j_0 \in I \) we had \( \Theta_C(k_i) \nsubseteq \Theta_C(k'_i) \) for all \( i \in I \) then \( \cup_{i \in I}(\Theta_D(k'_i) \Rightarrow \top)(\Theta_D(k'_i)) = \top \) and \( \cup_{i \in I}(\Theta_D(k_i) \Rightarrow \top)(\Theta_D(k'_i)) = \bot. \)

39
\( \sigma = \kappa, \tau = \kappa' \in L^s_C \): since the ordering over \( C = D \times D \times \cdots \) is component-wise, so that
\[
\langle d_1, d_2, \ldots \rangle \sqcup \langle d'_1, d'_2, \ldots \rangle = \langle d_1 \sqcup d'_1 \times d_2 \sqcup d'_2, \ldots \rangle
\]
and \( \Theta_C(\kappa_1 \land \kappa_2) = \Theta_C(\kappa_1) \cup \Theta_C(\kappa_2) \), we can assume without loss of generality that \( \kappa = \delta_1 \times \cdots \times \delta_n \times \omega \) and \( \kappa' = \delta'_1 \times \cdots \times \delta'_m \times \omega \). Hence the hypothesis reads as
\[
\Theta_C(\kappa') = \langle \Theta_D(\delta'_1), \ldots, \Theta_D(\delta'_m), \bot, \ldots \rangle \sqsubseteq \langle \Theta_D(\delta_1), \ldots, \Theta_D(\delta_n), \bot, \ldots \rangle = \Theta_C(\kappa)
\]
where \( \bot, \ldots \) stands for infinitely many \( \bot \). This implies that \( \Theta_D(\delta'_i) \sqsubseteq \Theta_D(\delta_i) \) for all \( i \leq \min(m, n) \). It is easy to see that \( \Theta_D(\delta) \neq \bot \) for any \( \delta \in L^s_D \), hence \( m \leq n \) and, by induction, \( \delta'_i \leq^s_D \delta_i \) for all \( i \leq m \). Now \( \delta_{m+1} \times \cdots \times \delta_n \times \omega \leq^s_C \omega \) by axiom, hence \( \delta_i \times \cdots \times \delta_{m+1} \times \cdots \times \delta_n \times \omega \leq^s \delta'_1 \times \cdots \times \delta'_m \times \omega \).

**Theorem 6.3**  The pre-orders \( \leq^s_D \) and \( \leq^s_C \) are the restriction to \( L^s \) of \( \leq_D \) and \( \leq_C \), respectively.

**Proof:** Let \( \sigma, \tau \in L^s_A \) for either \( A = D, C \), then:
\[
\sigma \leq_A \tau \quad \Rightarrow \quad \| \sigma \|^A \subseteq \| \tau \|^A \quad \text{(Corollary 4.14)} \\
\Rightarrow \quad \uparrow_A \Theta_A(\sigma) \subseteq \uparrow_A \Theta_A(\tau) \quad \text{(Lemma 4.13)} \\
\Rightarrow \quad \Theta_A(\tau) \subseteq \Theta_A(\sigma) \quad \text{(since \( \Theta_A(\sigma) \in \uparrow_A \Theta_A(\tau) \))} \\
\Rightarrow \quad \sigma \leq_A^s \tau \quad \text{(Lemma 6.2)}
\]

Since trivially \( \leq^s_A \subseteq \leq_A \), this establishes the thesis.

**Definition 6.4** (Restricted Bases, Contexts, Judgments, and Type Assignment)

i) A restricted basis is a basis \( \Gamma \) such that \( \delta \in L^s_D \) for all \( x: \delta \in \Gamma \). Similarly, a restricted name context is a context \( \Delta \) with \( \kappa \in L^s_C \) for all \( a: \kappa \in \Delta \). Finally, for \( T \in \text{Trm} \cup \text{Cmd} \) we say that \( \Gamma \vdash T : \sigma \mid \Delta \) is a restricted judgement if \( \sigma \in L^s \) and \( \Gamma \) and \( \Delta \) are a restricted basis and a restricted name context respectively.

ii) The restricted judgement \( \Gamma \vdash T : \sigma \mid \Delta \) is derivable in the restricted typing system, written \( \Gamma \vdash^r T : \sigma \mid \Delta \), if it is derivable in the system of Definition 4.2 without using rule (\( \omega \)), and all the judgements in the derivation are restricted.

By its very definition and Theorem 6.3, the restricted system is just the intersection type system of Section 4, where types occurring in judgments are restricted, and the rule (\( \omega \)) is disallowed. This has the technical advantage that we can use in proofs the results from the previous section, notably Lemma 4.9.

### 6.2 Typability implies Strong Normalisation

In this subsection we will show that – as can be expected of a well-defined notion of type assignment that does not type recursion and has no general rule that types all terms – all typeable terms are strongly normalising.

For the full system of Definition 4.2 this is not the case. In fact, by means of types not allowed in the restricted system, it is possible to type the fixed-point constructor \( \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) \) in a non-trivial way, as shown by the following derivation:
Example 6.5 The fixed-point combinator is typeable in the system of Definition 4.2:

\[
\begin{align*}
\vdash \omega \times \omega \rightarrow v, x : \omega \vdash f : \omega \times \omega \rightarrow v & \quad (Ax) \\
\vdash f : \omega \times \omega \rightarrow v, x : \omega \vdash f(x) : \omega \rightarrow v & \quad (\text{App}) \\
\vdash \lambda x . f(x) : \omega \times \omega \rightarrow v & \quad (Abs) \\
\vdash \lambda f . (\lambda x . f(x))((\lambda x . f(x)) : (\omega \times \omega \rightarrow v) \times \omega \rightarrow v) & \quad (\text{App})
\end{align*}
\]

Notice that this term does not have a normal form, so it is not strongly normalisable.

We begin with the proof that if a term is typeable in the restricted system then it is strongly normalising. We show this by adapting to our system Tait’s computability argument and the idea of saturated sets (see [36], Chapter 3).

Definition 6.6 (Term Stacks) The set \( \text{Stk} \) of (finite) term stacks, whose elements we shall denote by \( \vec{L} \), is defined by the following grammar:

\[
\text{Stk} : \quad \vec{L} ::= \epsilon | L :: \vec{L}
\]

where \( \epsilon \) denotes the empty stack and \( L \in \text{Trm} \). Moreover, we define stack application as follows:

\[
\begin{align*}
M \epsilon & \triangleq M \\
M (P :: \vec{L}) & \triangleq (MP) \vec{L}
\end{align*}
\]

So, if \( \vec{L} \equiv L_1 :: \cdots :: L_k :: \epsilon \) we have \( ML \equiv M L_1 \cdots L_k \). We extend the notion of structural substitution to stacks as follows:

\[
\begin{align*}
T [x \leftarrow \epsilon] & \triangleq T \\
T [x \leftarrow P :: \vec{L}] & \triangleq (T [x \leftarrow P]) [x \leftarrow \vec{L}]
\end{align*}
\]

for \( T \in \text{Trm} \cup \text{Cmd} \), when each \( L_i \) does not contain \( a \).

We normally omit the trailing \( \epsilon \) of a stack. Notice that

\[
[a]M[a \leftarrow \vec{L}] \equiv [a]M[a \leftarrow L_1][a \leftarrow L_2] \cdots [a \leftarrow L_n] \equiv [a](M[a \leftarrow \vec{L}])\vec{L}
\]

The notion of string normalisation is formally defined as:

Definition 6.7 The set \( \text{SN} \) of terms that are strongly normalisable is the set of all terms \( M \in \text{Trm} \) such that no infinite reduction sequence out of \( M \) exists; we write \( \text{SN}(M) \) for \( M \in \text{SN} \), and \( \text{SN}^* \) for the set of finite stacks of terms in \( \text{SN} \), and write \( \text{SN}(\vec{L}) \) if \( \vec{L} \in \text{SN}^* \).

The following property of strong normalising terms is straightforward:

Property 6.8

i) If \( \text{SN}(x\vec{M}) \) and \( \text{SN}(\vec{N}) \), then \( \text{SN}(x\vec{M}\vec{N}) \).

ii) If \( \text{SN}(M[N/x]\vec{P}) \) and \( \text{SN}(\vec{N}) \), then \( \text{SN}((\lambda x . M)\vec{N}\vec{P}) \).

iii) If \( \text{SN}(M) \), then \( \text{SN}(\mu \alpha . [\beta] M) \).

iv) If \( \text{SN}(\mu \alpha . [\beta] M[a \leftarrow \vec{N}] \vec{L}) \) and \( \text{SN}(\vec{N}) \), then \( \text{SN}((\mu \alpha . [\beta] M)\vec{N}\vec{L}) \).

v) If \( \text{SN}(\mu \alpha . [\alpha] M[a \leftarrow \vec{N}] \vec{L}) \), then \( \text{SN}((\mu \alpha . [\alpha] M)\vec{N}\vec{L}) \).
Definition 6.9 (Type Interpretation) We define a mapping

$$\| : (L^r_D \to \varphi(\text{Trm})) + (L^r_C \to \varphi(\text{Stk}))$$

interpreting types as sets of terms and stacks, as follows:

$$\| \kappa \to \nu \| = \{ T \mid \forall \vec{L} \in \| \kappa \| [SN(TL)] \}$$

$$\| \nu \| = SN^*$$

$$\| \delta \times \kappa \| = \{ N :: \vec{L} \mid N \in \| \delta \| & \vec{L} \in \| \kappa \| \}$$

$$\| \sigma \wedge \tau \| = \| \sigma \| \cap \| \tau \|$$

$$(\sigma, \tau \in L^*_A & A = D, C).$$

We now show that the interpretation of a type is a set of strongly normalisable terms and that neutral terms (those starting with a variable) are in the interpretation of any type.

Lemma 6.10 For any $\delta \in L^*_D$ and $\kappa \in L^*_C$:

i) $\| \delta \| \subseteq SN$ and $\| \kappa \| \subseteq SN^*$.

ii) $x\vec{N} \in SN \Rightarrow x\vec{N} \in \| \delta \|$.

Proof: By induction on the structure of types.

(i): Let $\delta = \kappa \to \nu$ and $M \in \| \delta \|$: then, for any $\vec{L} \in \| \kappa \|$, by definition, $SN(M\vec{L})$, so in particular $SN(M)$.

The case $\kappa = \omega$ follows by definition; the case $\kappa = \delta \times \kappa'$ follows by induction, since $\| \delta \| \subseteq SN$ and $\| \kappa' \| \subseteq SN^*$. The cases $\delta = \delta_1 \wedge \delta_2$ and $\kappa = \kappa_1 \wedge \kappa_2$ follow immediately by induction.

(ii): Let $x\vec{N} \in SN$ and $\delta = \kappa \to \nu$. If $\vec{L} \in \| \kappa \|$ then $\vec{L} \in SN^*$ by point (i), so that $x\vec{N}\vec{L} \in SN$ by observing that the only possible reductions are inside the components of $\vec{N}$ and $\vec{L}$, which are in $SN$ by assumption. Then by definition $x\vec{N} \in \| \delta \|$.

Observe that $\nu \notin L^*_D$, and therefore we do not have the clause $\| \nu \| = SN$ in Definition 6.9, as it was the case in [11]. This clause would be consistent with the previous definition, but having $\nu \in L_D$ (the unrestricted language of term types) enforces the equation $\nu = D \omega$ which is false in the above interpretation. In fact, $\lambda x.xx \in SN = \| \nu \|$, so that $(\lambda x.xx) :: e \in SN^* = \| \nu \|$, but $\lambda x.xx)((\lambda x.xx) :: e) \equiv (\lambda x.xx)(\lambda x.xx) \notin SN$, and therefore $\| \nu \| \not\subseteq \| \omega \to \nu \|$.

In [11] we avoided this incoherence by ruling out $\omega$ from $L^*_C$ and by interpreting types $\kappa \to \nu$ differently according to the shape of $\kappa$; in fact, in that paper $\| \kappa \to \nu \|$ is the set of representable functions from $SN^*$ to $SN$ only when $\kappa \neq \omega$, while $\| \omega \to \nu \|$ is just $SN$.

We will now show that our type interpretation respects type inclusion.

Lemma 6.11 For all $\sigma, \tau \in L^*_C$: $\sigma \leq^* \tau \Rightarrow \| \sigma \| \subseteq \| \tau \|$.

Proof: By easy induction over the rules in Definition 6.1. For $\kappa \leq^* \omega$, notice that $\| \kappa \| \subseteq SN^* = \| \omega \|$ by Lemma 6.10 (i). If $\kappa \leq^* \kappa'$, then $\kappa = \delta_1 \times \cdots \times \delta_n \times \omega$ and $\kappa' = \delta'_1 \times \cdots \times \delta'_m \times \omega$ and $m \leq n$ and $\delta_i \leq^* \delta'_i$; then by induction $\| \delta_i \| \subseteq \| \delta'_i \|$ for all $i \leq m$; notice that $\| \delta_{i+1} \times \cdots \times \delta_n \times \omega \| \subseteq \| \omega \| = SN^*$ by Lemma 6.10 (ii), so we can conclude $\| \kappa \| \subseteq \| \kappa' \|$. Let $\kappa_1 \to \nu \leq^* \kappa_2 \to \nu$ because $\kappa_2 \leq^* \kappa_1$; then by induction, $\| \kappa_2 \| \subseteq \| \kappa_1 \|$. Assume now $M \in \| \kappa_1 \to \nu \|$ then by definition for all $\vec{L} \in \| \kappa_1 \|$, we have $M\vec{L} \in SN$. Since $\| \kappa_2 \| \subseteq \| \kappa_1 \|$, also for all $\vec{L} \in \| \kappa_2 \|$, we have $M\vec{L} \in SN$, and therefore also $M \in \| \kappa_2 \to \nu \|$.

The next lemma states that our type interpretation is closed under expansion for the logical and for the structural reduction, with the proviso that the term or stack to be substituted is an element of an interpreted type as well.

Lemma 6.12 For any $\delta, \delta' \in L^*_D$ and $\kappa \in L^*_C$:
Definition 6.13

with a normal term (or stack).

Lemma 6.14

substitution instance \( M \) to achieve that, we first show, in Lemma 6.14, that for any a term
tively.

\( M \) not appear in \( \delta \). By induction on the structure of types. For the first case, if

is strongly normalisable. We need these substitutions to be applied all ‘in one go’, so define

The second and third case follow similarly, but using Proposition 6.8. (ii) and Proposition 6.8. (iv) respectively.

In Theorem 6.15 we will show that all typeable terms are strongly normalisable. In order to achieve that, we first show, in Lemma 6.14, that for any a term \( M \) typeable with \( \delta \), any full substitution instance \( M_\xi \) (i.e. replacing all free term variables by terms, and feeding stacks to all free names) is an element of the interpretation of \( \delta \), which by Lemma 6.10 implies that \( M_\xi \) is strongly normalisable. We need these substitutions to be applied all ‘in one go’, so define a notion of parallel substitution. The main result is then obtained by taking the substitution that replaces term variables by themselves and names by stacks of term variables. The reason we first prove the result for any substitution is that, in the proof of Lemma 6.14, in the case for \( \lambda x.M \) and \( \mu a.Q \) the substitution is extended, by replacing the bound variable or name with a normal term (or stack).

Definition 6.13

i) A partial mapping \( \xi : (\text{VAR} \to \text{TRM}) + (\text{NAME} \to \text{TRM}^*) \) is a parallel substitution if, for

every \( p,q \in \text{dom}(\xi) \), if \( p \neq q \) then \( p \notin \text{fv}(\xi q) \) and \( p \notin \text{fn}(\xi q) \).

ii) Borrowing a notation for valuations, for a parallel substitution \( \xi \) we define the applica-
tion of \( \xi \) to a term by:

\[
\begin{align*}
([\alpha]M)_\xi & \triangleq [\alpha]M_\xi \bar{L} & \text{if } \bar{\xi} \alpha = \bar{L} \\
([\beta]M)_\xi & \triangleq [\beta]M_\xi & \text{if } \beta \notin \text{dom}(\xi) \\
(\mu \beta.Q)_\xi & \triangleq \mu \beta.Q_\xi \\
x_\xi & \triangleq N & \text{if } \bar{\xi} x = N \\
y_\xi & \triangleq y & \text{if } y \notin \text{dom}(\xi) \\
(\lambda x.M)_\xi & \triangleq \lambda x.M_\xi \\
(MN)_\xi & \triangleq M_\xi N_\xi
\end{align*}
\]

iii) We define \( \xi[N/x] \) and \( \xi[\alpha \leftarrow \bar{L}] \) by, respectively,

\[
\begin{align*}
\xi[N/x] y & \triangleq \begin{cases} 
N & \text{if } y = x \\
\xi y & \text{otherwise}
\end{cases} \\
\xi[\alpha \leftarrow \bar{L}] \beta & \triangleq \begin{cases} 
\bar{L} & \text{if } \bar{\xi} \alpha = \bar{L} \\
\xi \beta & \text{otherwise}
\end{cases}
\end{align*}
\]

iv) We say that \( \xi \) extends \( \Gamma \) and \( \Delta \), if, for all \( x: \delta \in \Gamma \) and \( \alpha : \kappa \in \Delta \), we have, respectively, \( \xi x \in \| \delta \| \) and \( \bar{\xi} \alpha \in \| \kappa \| \).

Notice that we do allow a variable to appear in its own image under \( \xi \). Since \( x \) does not appear in \( M[N/x] \) whenever \( x \) does not appear in \( M \) and \( N \), this does not violate Barendregt’s convention.

Lemma 6.14 (Replacement Lemma) Let \( \xi \) be a parallel substitution that extends \( \Gamma \) and \( \Delta \). Then:

\[
\text{if } \Gamma \vdash x : \sigma | \Delta \text{ then } T_\xi \in \| \sigma \|.
\]
Proof: By induction on the structure of derivations. We show some more illustrative cases.

(Abs): Then $T = \lambda x. M$, $\sigma = \delta \times \kappa \to \nu$, and $\Gamma, x : \delta \vdash M : \kappa \to \nu$. Take $N \in \| \delta^r \|$; since $x$ is bound, by Barendregt’s convention we can assume that it does not occur free in the image of $\xi$, so $\xi[N/x]$ is a well-defined parallel substitution that extends $\Gamma, x : \delta$ and $\Delta$. Then by induction, we have $M_{\xi[N/x]} \in \| \kappa \to \nu \|$. Since $x$ does not occur free in the image of $\xi$, $M_{\xi[N/x]} = M_\kappa[N/x]$, so also $M_\kappa[N/x] \in \| \kappa \to \nu \|$. By Lemma 6.12 ((i)), also $(\lambda x. M_\kappa) N \in \| \kappa \to \nu \|$. By definition of $\| \kappa \to \nu \|$, for any $\tilde{L} \in \| \kappa \|$ we have $(\lambda x. M_\kappa) N \tilde{L} \in \| \nu \|$; notice that $N :: \tilde{L} \in \| \delta \times \kappa \|$, so $(\lambda x. M_\kappa)_\xi \in \| \delta^r \times \kappa \to \nu \|$. 

(µ): Then $T = \mu a . C$, $\delta = \kappa \to \nu$, and there exists $\kappa'$ such that $\Gamma \vdash C : (\kappa' \to \nu) \times \kappa' \mid a : \kappa, \Delta$. Take $\tilde{L} \in \| \kappa \|$. Since $a$ is bound in $T$, we can assume it does not occur free in the image of $\xi$, so $\xi[a \leftarrow \tilde{L}]$ is a well-defined parallel substitution that extends $\Gamma$ and $a : \kappa, \Delta$, so by induction, $C_{\xi[a \leftarrow \tilde{L}]} \in \| \kappa \to \nu \|$. By Definition 6.9, $\mathcal{SN}(C_{\xi[a \leftarrow \tilde{L}]})$. Since $a$ does not occur free in the image of $\xi$, $C_{\xi[a \leftarrow \tilde{L}]} = C_{\xi}[\alpha \leftarrow \tilde{L}]$, so we have $C_{\xi}[\alpha \leftarrow \tilde{L}] \in \| \kappa \to \nu \|$. But then by Lemma 6.8 ((iii)) also $\mathcal{SN}((\mu a. C_\xi)[\alpha \leftarrow \tilde{L}])$. Then by Proposition 6.8((v), also $\mathcal{SN}((\mu a. C_\xi)) \Rightarrow (\mu a. C)_\xi \in \| \kappa \to \nu \|$. 

(Cmd): Then $T = [\alpha]M$, $\sigma = \delta \times \kappa$, $\Delta = \alpha : \kappa, \Delta'$, and $\Gamma \vdash M : \delta \mid \Delta'$. We distinguish two subcases.

$\alpha = \beta$: Then $M = \mu a . [\alpha]M'$, $\delta = \kappa \to \nu$, and $\Gamma \vdash M' : \kappa \to \nu \mid \alpha : \kappa, \Delta$. Take $\tilde{L} \in \| \kappa \|$. Since $\alpha$ is bound in $M$, we can assume it does not occur free in the image of $\xi$, so $\xi[a \leftarrow \tilde{L}]$ is a well-defined parallel substitution that extends $\Gamma$ and $\Delta, \alpha : \kappa$, and by induction, $M'_{\xi[a \leftarrow \tilde{L}]} \in \| \kappa \to \nu \|$. Since $a$ does not occur free in the image of $\xi$, $M'_{\xi[a \leftarrow \tilde{L}]} = M'_{\xi}[\alpha \leftarrow \tilde{L}]$, so we have $M'_{\xi}[\alpha \leftarrow \tilde{L}] \in \| \kappa \to \nu \|$, and therefore $M'_{\xi}[\alpha \leftarrow \tilde{L}] \tilde{L} \in \| \nu \|$. Then by Definition 6.9, $\mathcal{SN}(M'_{\xi}[\alpha \leftarrow \tilde{L}])$, but then also $\mathcal{SN}(M'_{\xi}[\alpha \leftarrow \tilde{L}])$, by Lemma 6.8 ((iii)). So $\mu a . [\alpha]M'_{\xi}[\alpha \leftarrow \tilde{L}] \tilde{L} \in \| \nu \|$. Then by Lemma 6.12 ((iii)), $(\mu a . [\alpha]M')_\xi \tilde{L} \in \| \nu \|$, so $(\mu a . [\alpha]M')_{\xi} \in \| \kappa \to \nu \|$. 

$\alpha \neq \beta$: Then $\Delta = \beta : \kappa', \Delta'$, and $\Gamma \vdash M' : k' \to \nu \mid \alpha : \kappa, \beta : \kappa', \Delta$. Assume $\tilde{L} \in \| \kappa \|$, then $\xi[a \leftarrow \tilde{L}]$ extends $\Gamma$ and $\alpha : \kappa, \beta : \kappa', \Delta$. Then, by induction, $M'_{\xi[a \leftarrow \tilde{L}]} \in \| \kappa' \to \nu \|$. Now let $\tilde{Q} \in \| \kappa' \|$, then $M'_{\xi[a \leftarrow \tilde{L}]} \tilde{Q} \in \| \nu \|$, and then also $(M' \tilde{Q})_{\xi[a \leftarrow \tilde{L}]} \in \| \nu \|$. Then $\mathcal{SN}(M' \tilde{Q})_{\xi[a \leftarrow \tilde{L}]}$ by Definition 6.9, and $\mathcal{SN}(M' \tilde{Q})_{\xi[a \leftarrow \tilde{L}]}$ by Lemma 6.8 ((iii)), so, again by Definition 6.9, $\mu a . \beta)(M' \tilde{Q})_{\xi[a \leftarrow \tilde{L}]} \in \| \nu \|$. As in the previous part, $a$ is not free in the image of $\xi$, and therefore also $\mu a . \beta)(M' \tilde{Q})_{\xi[a \leftarrow \tilde{L}] \tilde{L} \in \| \nu \|}$. Then, by Lemma 6.12 ((ii)), $(\mu a . \beta)(M' \tilde{Q})_{\xi[a \leftarrow \tilde{L}]] \tilde{L} \in \| \nu \|}$. Notice that $\beta M'_{\xi} \tilde{Q} = \beta M'_{\xi}[\beta \leftarrow \tilde{Q}]$; since $\xi \beta = \tilde{Q}$, we can infer that $\beta M'_{\xi} \tilde{Q} = \beta M'_{\xi}$, so $(\mu a . \beta)(M' \tilde{Q})_{\xi[a \leftarrow \tilde{L}] \tilde{L} \in \| \nu \|}$. But then $(\mu a . \beta)(M')_{\xi} \in \| \kappa \to \nu \|$. 

We now come to the main result of this section, that states that all terms typeable in the restricted system are strongly normalisable.

**Theorem 6.15 (Typeable terms are SN)** If $\Gamma \vdash M : \delta \mid \Delta$ for some $\Gamma$, $\Delta$ and $\delta$, then $M \in \mathcal{SN}$. 

Proof: Let $\xi$ be a parallel substitution such that 

$$
\begin{align*}
\xi x &= x & \text{for } x \in \text{dom}(\Gamma) \\
\xi a &= \bar{y}_a & \text{for } a \in \text{dom}(\Delta)
\end{align*}
$$

where the length of the stack $\bar{y}_a$ is $|\kappa|$ if $a : \kappa \in \Delta$ (notice that $\xi$ is well defined). By Lemma 6.10, $\xi$ extends $\Gamma$ and $\Delta$. Hence, by Lemma 6.14, $M_\xi \in \| \delta \|$, and then $M_\xi \in \mathcal{SN}$ by Lemma 6.10 ((i)).
Now
\[ M_\xi \equiv M[x_1/x_1, \ldots, x_n/x_n, \alpha_1 \leftarrow \bar{y}_{a_1}, \ldots, \alpha_m \leftarrow \bar{y}_{a_m}] \]
\[ \equiv M[\alpha_1 \leftarrow \bar{y}_{a_1}, \ldots, \alpha_m \leftarrow \bar{y}_{a_m}] \]

Then, by Proposition 6.8, for any \( \bar{y} \) also \((\mu \alpha_1[\beta_1] \cdots \mu \alpha_m[\beta_m] M)\bar{y}_{a_1} \cdots \bar{y}_{a_m} \in \mathcal{SN}\), and therefore also \( M \in \mathcal{SN}\).

### 6.3 Strongly Normalising Terms are Typeable

In this section we will show the counterpart of the previous result, i.e. that all strongly normalisable terms are typeable in our restricted intersection system. This result has been claimed in many papers [46, 5], but has rarely been proven completely.

First we give the shape of terms and commands in normal forms.

**Definition 6.16 (Normal Forms)** The sets \( \mathcal{N} \subseteq \text{Trm} \) and \( \mathcal{C} \subseteq \text{Cmd} \) of normal forms are defined by the grammar:

\[
\mathcal{N} ::= xN_1 \cdots N_k | \lambda x.N | \mu x.C
\]
\[
\mathcal{C} ::= [\beta]N
\]

It is straightforward to verify that \( \mathcal{N} \) and \( \mathcal{C} \) coincide with the sets of irreducible terms and commands, respectively.

We can now show that all terms and commands in normal form are typeable in the restricted system.

**Lemma 6.17**

i) If \( N \in \mathcal{N} \) then there exist \( \Gamma, \Delta \) and a type \( \kappa \rightarrow \nu \in \mathcal{L}_D^\kappa \) such that \( \Gamma \vdash N : \kappa \rightarrow \nu \mid \Delta \).

ii) If \( C \in \mathcal{C} \) then there exist \( \Gamma, \Delta \) and a type \( \kappa \in \mathcal{L}_C^\kappa \) such that \( \Gamma \vdash C : (\kappa \rightarrow \nu) \times \kappa \mid \Delta \).

**Proof:** By simultaneous induction over the definitions of \( \mathcal{N} \) and \( \mathcal{C} \).

\( (N \equiv xN_1 \cdots N_k) \): Since \( N_1, \ldots, N_k \in \mathcal{N} \), by induction \((i)\) we have that, for all \( i \leq k \) there exist \( \Gamma_i, \Delta_i \) and \( \delta_i \) such that \( \Gamma_i \vdash N_i : \delta_i \mid \Delta_i \) (the structure of each \( \delta_i \) plays no role in this part). Take

\[
\Gamma = \Gamma_1 \wedge \cdots \wedge \Gamma_k \wedge \{x:(\delta_1 \times \cdots \times \delta_k \times \omega) \rightarrow \nu\}, \text{ and } \Delta = \Delta_1 \wedge \cdots \wedge \Delta_k.
\]

Then, by weakening, also \( \Gamma \vdash N_i : \delta_i \mid \Delta \) for all \( i \leq k \), and \( \Gamma \vdash x:(\delta_1 \times \cdots \times \delta_k \times \omega) \rightarrow \nu \mid \Delta \).

By repeated applications of \((\text{App})\) we get \( \Gamma \vdash xN_1 \cdots N_k : \omega \rightarrow \nu \mid \Delta \).

\( (N \equiv \lambda x.M) \): By induction \((i)\) there exist \( \Gamma, \delta' \), and \( \Delta \) such that \( \Gamma, x: \delta' \vdash M : \kappa \rightarrow \nu \mid \Delta \) (if \( x \notin \text{fv}(M) \), we can add \( x: \delta \) by weakening, for any \( \delta' \in \mathcal{L}_D^\kappa \)). Then by \((\text{Abs})\) we obtain \( \Gamma \vdash \lambda x.M : \delta \times \kappa \rightarrow \nu \mid \Delta \).

\( (N \equiv \mu x.C) \): By induction \((i)\) there are \( \Gamma, \kappa, \kappa', \) and \( \Delta \) such that \( \Gamma \vdash C : (\kappa \rightarrow \nu) \times \kappa \mid \alpha : \kappa' \times \Delta \) (if \( \alpha \notin \text{fn}(C) \), we can add \( \alpha : \kappa' \) by weakening, for any \( \kappa' \in \mathcal{L}_C^\kappa \)). We get \( \Gamma \vdash \mu x.C : \kappa' \rightarrow \nu \mid \Delta \) by applying rule \((\mu)\).

\( (C \equiv [\beta]N) \): By induction \((i)\) there are \( \Gamma, \delta = \kappa \rightarrow \nu \), and \( \Delta \) such that \( \Gamma \vdash N : \delta \mid \Delta \) for some \( \kappa' \). If \( \beta \notin \text{dom}(\Delta) \), by weakening also \( \Gamma \vdash N : \beta : \kappa, \Delta \), and by rule \((\text{Cmd})\) we obtain \( \Gamma \vdash \beta [\beta]N : (\kappa \rightarrow \nu) \times \kappa \mid \beta : \kappa, \Delta \). If \( \Delta = \beta \times \kappa' \), we can construct the derivation:

\[
\begin{align*}
\Gamma \vdash N' : \kappa' \rightarrow \nu & \mid \beta : \kappa, \Delta' \\
\Gamma \vdash \beta [\beta]N : (\kappa' \rightarrow \nu) \times (\kappa \times \kappa') & \mid \beta \times \kappa \times \kappa' \times \Delta' \\
\end{align*}
\]

which shows the result.
Note that in the last case the, using \((\kappa' \rightarrow \nu) \times (\kappa \wedge \kappa')\) instead of \((\kappa' \rightarrow \nu) \times \kappa\) (otherwise we could not apply rule \((\mu)\) to type \(\mu \alpha. [B]N\) comes at the price of weakening the assumption \(B \cdot \kappa\) to \(B \cdot \kappa \wedge \kappa'\). However, this is not a disadvantage since we get, for example, \(\Gamma \vdash_\kappa \mu B. [B]N : (\kappa \wedge \kappa') \rightarrow \nu \mid \Delta'\) which safely records in the antecedent type \(\kappa \wedge \kappa'\) the functionality of \(N\); notice that, in fact \(\kappa' \rightarrow \nu \leq^R (\kappa \wedge \kappa') \rightarrow \nu\).

We will now show that typing is closed under expansion of contracts, with respect to both logical and structural reduction, with the proviso that the term that gets substituted is typeable as well.

**Lemma 6.18 (Restricted \(\beta\)-Expansion)** If \(\Gamma \vdash_\kappa M[N/x] : \delta \mid \Delta\) and \(\Gamma \vdash_\kappa N : \delta' \mid \Delta\) then \(\Gamma \vdash_\kappa (\lambda x. M)N : \delta \mid \Delta\).

**Proof:** By Lemma 4.9 we can assume without loss of generality that \(\delta = \kappa \rightarrow \nu\). We consider two cases:

1. \((x \notin \text{fv}(M))\): Then \(M[N/x] \equiv M\) and we may assume \(x \notin \text{dom}(\Gamma)\); by Lemma 4.5 we have \(\Gamma, x: \delta \vdash_\kappa M : \kappa \rightarrow \nu \mid \Delta\) so that \(\Gamma \vdash_\kappa \lambda x. M : \delta' \times \kappa \rightarrow \nu \mid \Delta\) by rule \((\text{Abs})\), and then \(\Gamma \vdash_\kappa (\lambda x. M)N : \kappa \rightarrow \nu \mid \Delta\) by using rule \((\text{App})\).

2. \((x \in \text{fv}(M))\): Since there exists a derivation that justifies \(\Gamma \vdash M[N/x] : \kappa \rightarrow \nu \mid \Delta\), and \(N\) is a true sub-term of \(M[N/x]\), there exists \(k \geq 1\) sub-derivations that have a judgement for \(N\) in their conclusion, so there exist \(\delta_1, \ldots, \delta_k\) such that \(\Gamma \vdash N : \delta_i \mid \Delta\) for all \(i \leq k\). Since \(\delta_1 \wedge \cdots \wedge \delta_k \leq^R \delta_j\), we can derive \(\Gamma, x: \delta_1 \wedge \cdots \wedge \delta_k \vdash_\kappa M : \kappa \rightarrow \nu \mid \Delta\) by replacing, for all \(j \leq k\), each sub-derivation for \(\Gamma \vdash N : \delta_j \mid \Delta\) by:

\[
\Gamma, x: \delta_1 \wedge \cdots \wedge \delta_k \vdash x: \delta_1 \wedge \cdots \wedge \delta_k \mid \Delta \\
\Gamma, x: \delta_1 \wedge \cdots \wedge \delta_k \vdash x : \delta_j \mid \Delta \quad (\leq)
\]

Hence \(\Gamma \vdash_\kappa \lambda x. M : (\delta_1 \wedge \cdots \wedge \delta_k) \times \kappa \rightarrow \nu \mid \Delta\) by rule \((\text{Abs})\). On the other hand, by \((\wedge)\), we have \(\Gamma \vdash_\kappa N : \delta_1 \wedge \cdots \wedge \delta_k \mid \Delta\), and we obtain \(\Gamma \vdash_\kappa (\lambda x. M)N : \delta \mid \Delta\) using rule \((\text{App})\).

**Lemma 6.19 (Restricted \(\mu\)-Expansion Lemma)**

i) If both \(\Gamma \vdash_\kappa ([a]M)[a \leftarrow L] : (\kappa_1 \rightarrow \nu) \times \kappa_2 \mid \kappa : \kappa_2, \Delta\) and \(\Gamma \vdash_\kappa L : \delta \mid \Delta\) then there exists \(\delta'\) such that \(\Gamma \vdash_\kappa [a]M : (\delta' \times \kappa_1 \rightarrow \nu) \times (\delta' \times \kappa_2) \mid \kappa : \delta' \times \kappa_2, \Delta\) and \(\Gamma \vdash_\kappa L : \delta' \mid \Delta\).

ii) If \(T \neq [a]M\), then \(\Gamma \vdash_\kappa T[a \leftarrow L] : \sigma \mid \kappa : \kappa, \Delta\) and \(\Gamma \vdash_\kappa L : \delta \mid \Delta\) then there exists \(\delta'\) such that \(\Gamma \vdash_\kappa T : \sigma \mid \kappa : \delta' \times \kappa, \Delta\) and \(\Gamma \vdash_\kappa L : \delta' \mid \Delta\).

**Proof:** Simultaneous, by induction on the definition of structural substitution.

i) By definition, \([a]M[a \leftarrow L] = [a][M[a \leftarrow L]]L\), so \(\Gamma \vdash_\kappa [a]M[a \leftarrow L] : (\kappa_1 \rightarrow \nu) \times \kappa_2 \mid \kappa : \kappa_2, \Delta\).

By Lemma 4.10 there exist \(\delta_1\) such that \(\Gamma \vdash_\kappa \lambda \alpha \cdot \delta_1 \cdot \kappa_1 \rightarrow \nu \mid \kappa : \kappa_2, \Delta\) and \(\Gamma \vdash_\kappa L : \delta_1 \mid \kappa : \kappa_2, \Delta\). Then by induction (ii) there exists \(\delta_2\) such that \(\Gamma \vdash_\kappa M : (\delta_1 \times \kappa_1) \rightarrow \nu \mid \kappa : \delta_2 \times \kappa_2, \Delta\) and \(\Gamma \vdash_\kappa L : \delta_2 \mid \kappa : \kappa_2, \Delta\). Take now \(\delta' = \delta_1 \wedge \delta_2\), then we can construct:

\[
\Gamma \vdash_\kappa : (\delta_1 \times \kappa_1) \rightarrow \nu \mid \kappa : \delta_2 \times \kappa_2, \Delta \quad (S) \\
\Gamma \vdash_\kappa M : (\delta_1 \times \kappa_1) \rightarrow \nu \mid \kappa : \delta_2 \times \kappa_2, \Delta \quad (S) \\
\Gamma \vdash_\kappa : (\delta_1 \times \kappa_1) \rightarrow \nu \mid \kappa : \delta_2 \times \kappa_2, \Delta \\
\Gamma \vdash_\kappa \lambda \alpha :: \delta' \times \kappa_1, \Delta \quad (\text{Cmd})
\]

ii) We have five cases to consider:

\((T = x)\): Then \(T[a \leftarrow L] = x\); take \(\delta' = \delta\), then by assumption, \(\Gamma \vdash_\kappa L : \delta' \mid \Delta\); from \(\Gamma \vdash_\kappa x : \sigma \mid \kappa : \kappa, \Delta\), by thinning and weakening we obtain \(\Gamma \vdash_\kappa x : \sigma \mid \kappa : \delta' \times \kappa, \Delta\).
\((T = \lambda x. N)\): Then \(T[x \Leftarrow L] = \lambda x. N[x \Leftarrow L]\). This case follows by straightforward induction.

\((T = PQ)\): Then \(T[x \Leftarrow L] = (P[x \Leftarrow L])(Q[x \Leftarrow L])\). By Lemma 4.10, \(\sigma = \kappa' \rightarrow \rho\) and there exists \(\delta_1\) such that \(\Gamma \vdash P[x \Leftarrow L] : \delta_1 \times \kappa' \rightarrow \rho\) and \(\alpha : \kappa, \Delta\) and \(\Gamma \vdash Q[x \Leftarrow L] : \delta_1 \rightarrow \gamma : \kappa, \Delta\). Then by induction, there are \(\delta_2\) and \(\delta_3\) such that \(\Gamma \vdash P : \delta_1 \times \kappa' \rightarrow \rho\) and \(\alpha : \delta_2 \times \kappa, \Delta\) and \(\Gamma \vdash L : \delta_2 \mid \Delta\), as well as \(\Gamma \vdash Q : \delta_1 \mid \alpha : \delta_3 \times \kappa, \Delta\) and \(\Gamma \vdash L : \delta_3 \mid \Delta\). Take now \(\delta' = \delta_2 \land \delta_3\), then by \(\land\) we get \(\Gamma \vdash L : \delta' \mid \Delta\) and we can construct:

\[
\begin{align*}
\Gamma &\vdash P : \delta_1 \times \kappa' \rightarrow \rho \mid \alpha : \delta_2 \times \kappa, \Delta \quad \text{(S)} \\
\Gamma &\vdash Q : \delta_1 \mid \alpha : \delta_3 \times \kappa, \Delta \quad \text{(S)} \\
\Gamma &\vdash \bot \quad \text{(App)}
\end{align*}
\]

\((T = \mu \beta. C)\): Then \(T[x \Leftarrow L] = \mu \beta. C[x \Leftarrow L]\). If \(\Gamma \vdash C[x \Leftarrow L] : \sigma \mid \alpha : \kappa, \Delta\), then by Lemma 4.10 there exists \(\kappa_1, \kappa_2\) such that \(\sigma = \kappa_1 \rightarrow \nu\) and

\[
\begin{align*}
\Gamma &\vdash C[x \Leftarrow L] : (\kappa_2 \rightarrow \nu) \times \kappa_2 \mid \alpha : \kappa, \beta : \kappa_1, \Delta. \\
\end{align*}
\]

Then, by induction \((ii)\) there exists \(\delta'\) such that \(\Gamma \vdash C[x \Leftarrow L] : (\delta' \times \kappa_2 \rightarrow \nu) \times (\delta' \times \kappa_2) \mid \alpha : \delta' \times \kappa, \beta : \kappa_1, \Delta\). By applying rule \((\mu)\), we get

\(\Gamma \vdash \mu \beta. C[x \Leftarrow L] : \kappa_1 \rightarrow \nu \mid \alpha : \delta' \times \kappa, \Delta\).

\((T = [\beta] M)\): Then \(T[x \Leftarrow L] = [\beta] M[x \Leftarrow L]\). If \(\Gamma \vdash [\beta] M[x \Leftarrow L] : \sigma \mid \alpha : \kappa, \Delta\), then by Lemma 4.10 there exists \(\kappa'\) such that \(\sigma = (\kappa' \rightarrow \nu) \times \kappa', \Delta' = \beta : \kappa', \Delta'\), and

\[
\begin{align*}
\Gamma &\vdash [\beta] M : \kappa' \rightarrow \nu \mid \alpha : \kappa, \beta : \kappa', \Delta'. \\
\end{align*}
\]

Then by induction \((ii)\) there exists \(\delta'\) such that \(\Gamma \vdash M : \kappa' \rightarrow \nu \mid \alpha : \delta' \times \kappa, \beta : \kappa', \Delta'\) and \(\Gamma \vdash L : \delta' \mid \beta : \kappa', \Delta'\). By applying rule \((\text{Cmd})\) to the first result, we derive \(\Gamma \vdash [\beta] M : (\kappa' \rightarrow \nu) \times \kappa' \mid \alpha : \delta' \times \kappa_2, \beta : \kappa', \Delta'\).

Notice that the additional assumption that \(\Gamma \vdash L : \delta\) is only of use in the case \(T = x\). Moreover, notice that the result strongly depends on the fact that \(L\) is typeable in the same basis and context as \(T[x \Leftarrow L]\).

It is tempting to conclude from Lemma 6.19 that if \(M \rightarrow N\) by contracting a redex \(PQ\) such that \(\Gamma \vdash Q : \delta' \mid \Delta\) for some \(\delta'\), then \(\Gamma \vdash N : \delta \mid \Delta\) implies \(\Gamma \vdash M : \delta \mid \Delta\). In fact, this result has been claimed on various papers in the past, but, unfortunately, is false even for the pure \(\lambda\)-calculus itself. Take \(\lambda x.(\lambda y.x)(x)\) which reduces to \(\lambda x.x\) that we can type as follows:

\[
\begin{align*}
\frac{x : \omega \rightarrow \nu, x : \omega \rightarrow \nu \vdash x : \omega \rightarrow \nu}{\vdash \lambda x.x : (\omega \rightarrow \nu) \times \omega \rightarrow \nu} \quad \text{(Abs)}
\end{align*}
\]

We cannot infer this type for \(\lambda x.(\lambda y.x)(x)\). To type the sub-term \(xx\) the minimum we can do is (setting \(\delta = \kappa \rightarrow \nu\)):

\[
\begin{align*}
\frac{x : \delta \land (\delta \times \kappa \rightarrow \nu), x : \delta \land (\delta \times \kappa \rightarrow \nu) \vdash x : \delta \land (\delta \times \kappa \rightarrow \nu)}{(Ax)} \\
\frac{\vdash x : \delta \land (\delta \times \kappa \rightarrow \nu)}{\vdash x : \delta \land (\delta \times \kappa \rightarrow \nu)} \quad \text{(Ax)}
\end{align*}
\]

For \(\lambda y.x\) we can construct:

\[
\begin{align*}
\frac{y : \delta, x : \delta \land (\delta \times \kappa \rightarrow \nu), y : \delta \land (\delta \times \kappa \rightarrow \nu) \vdash x : \delta \land (\delta \times \kappa \rightarrow \nu)}{(Ax)} \\
\frac{\vdash y : \delta, x : \delta \land (\delta \times \kappa \rightarrow \nu)}{\vdash y : \delta, x : \delta \land (\delta \times \kappa \rightarrow \nu)} \quad \text{(Ax)}
\end{align*}
\]

\[
\begin{align*}
\frac{x : \delta \land (\delta \times \kappa \rightarrow \nu), \lambda y.x : \delta \land (\delta \times \kappa \rightarrow \nu) \vdash x : \delta \land (\delta \times \kappa \rightarrow \nu)}{(Abs)}
\end{align*}
\]

47
for any $\kappa'$ such that $\delta \land (\delta \times \kappa \to \nu) \preceq^R \kappa' \to \nu$, and therefore

\[
\frac{x : \delta \land (\delta \times \kappa \to \nu) \vdash \lambda y. x : \delta \times \kappa' \to \nu}{x : \delta \land (\delta \times \kappa \to \nu) \vdash xx : \delta}
\]

\[
\frac{x : \delta \land (\delta \times \kappa \to \nu) \vdash (\lambda y. x)(xx) : \kappa' \to \nu}{\vdash \lambda x. (\lambda y. x)(xx) : (\delta \land (\delta \times \kappa \to \nu)) \times \kappa' \to \nu}
\]

(App)

(Abs)

So we cannot infer the type $(\omega \to \nu) \times \omega \to \nu$ for $\lambda x. (\lambda y. x)(xx)$ and a general subject-expansion result is impossible to prove. However, we can show that type assignment is preserved under expansions of leftmost-outermost redexes.

**Definition 6.20** An occurrence of a redex $R = (\lambda x. P)Q$ or $(\mu a. [\beta] P)Q$ in a term $M$ is called the left-most outer-most redex of $M$ (lor $M$), if and only if:

1. there is no redex $R'$ in $M$ such that $R' = C[R]$ with $C[-] \neq [-]$ (outer-most);
2. there is no redex $R'$ in $M$ such that $M = C_0[C_1[R']][C_2[R]]$ (left-most).

We write $M \to_{lor} N$ when $M$ reduces to $N$ by contracting $lor (M)$.

The following lemma formulates a subject expansion result for our system with respect to left-most outer-most reduction. A proof for this property in the context of strict intersection type assignment for the $\lambda$-calculus appeared in [6, 9].

**Lemma 6.21** Let $M \to_{lor} N$, lor $(M) = RQ$, $\Gamma_1 \vdash_R N : \delta_1 \mid \Delta_1$ with $\delta_1$ not an intersection, and $\Gamma_2 \vdash_R Q : \delta_2 \mid \Delta_2$, then there exist $\Gamma_3$, $\Delta_3$ and $\delta_3$ such that $\Gamma_3 \vdash_R M : \delta_3 \mid \Delta_3$.

**Proof:** By induction on the structure of terms. By assumption $\delta_1 = \kappa_1 \to \nu$. Then there are $\delta_j \ (j \in \mathbb{N})$ such that $\Delta_1 \vdash \delta_1 \times \cdots \times \delta_n \times \kappa_1 \to \nu \mid \Delta_1$ and $\Gamma_1 \vdash_R P_i : \delta'_i \mid \Delta_1$, for all $i \in \mathbb{N}$.

$(M = VP_1 \cdots P_n)$: Then either:

1. $V$ is a redex $(\lambda y. P)Q$, so lor $(M) = V$; let $V' = P[Q/y]$, where the substitution is capture avoiding, so all free variables in $Q$ are not bound in $P[Q/y]$ and we can assume that $\Gamma_2$ and $\Delta_2$ do not have types for bound variables and names in $P$. By Lemma 6.19, there exists $\Gamma_3$, $\Delta_3$ and $\delta_3$ such that $\Gamma_3 \vdash_R (\lambda y. P)Q : \delta_3 \mid \Delta_3$.
2. $V$ is a redex $(\mu a. [\beta] P)Q$, so lor $(M) = V$; let $V' \equiv \mu a. [\beta] P[a \leftarrow Q]$ or $V' \equiv \mu a. [\beta] P[a \leftarrow Q]$; we can assume that $\Gamma_2$ and $\Delta_2$ do not have types for bound variables and names in $\mu a. [\beta] P$. By Lemma 6.19, there exists $\Gamma_3$, $\Delta_3$ and $\delta_3$ such that $\Gamma_3 \vdash_R (\mu a. [\beta] P)Q : \delta_3 \mid \Delta_3$.
3. $V \equiv z$ and there is an $i \in \mathbb{N}$ such that lor $(M) = lor (P_i)$, $N \equiv zP_1 \cdots P_i \cdots P_n$, and $P_i \to_{lor} P'_i$. By induction there are $\Gamma'$, $\Delta'$, $\delta''_j$ such that $\Gamma' \vdash_R P_j : \delta''_j \mid \Delta'$. Take $\Gamma_3 = \Gamma_1 \land \Gamma', \delta''_j \vdash_R M' : \kappa' \times \nu \mid \Delta'$. Then, by rule (Abs), $\Gamma_1 \vdash_R \lambda y. M' : \delta \times \kappa' \to \nu \mid \Delta'$; take $\Gamma_3 = \Gamma', \Delta_3 = \Delta', \delta = \delta''_i \times \kappa' \to \nu$.

In all cases, $\Gamma_3 \vdash_R VP_1 \cdots P_n : \delta_3 \mid \Delta_3$.

$(M = \lambda y. M')$: If $M \to_{lor} N$, then $N = \lambda y. N'$ and $M' \to_{lor} N'$. Then there exists $\delta$ and $\kappa$ such that $\delta_1 = \delta \times \kappa \to \nu$, and $\Gamma_1, y : \delta \vdash_R N' : \kappa \to \nu \mid \Delta_1$. By induction, there exists $\Gamma'$, $\Delta'$, $\delta'$ and $\kappa'$ such that $\Gamma' : \delta' \times \kappa' \to \nu \mid \Delta'$. Then, by rule (Abs), $\Gamma' \vdash_R \lambda y. M' : \delta \times \kappa' \to \nu \mid \Delta'$; take $\Gamma_3 = \Gamma'$, $\Delta_3 = \Delta'$, and $\delta = \delta' \times \kappa' \to \nu$.

$(M = \mu a. [\beta] M')$: If $M \to_{lor} N$, then $N = \mu a. [\beta] N'$ and $M' \to_{lor} N'$. Then there exists $\kappa_1$ and $\kappa_2$ such that $\delta_1 = \kappa_1 \to \nu$, $\Delta_1 = a : \kappa_2, \Delta'_1$ and $\Gamma_1 \vdash_R N' : \kappa_2 \to \nu \mid \beta : \kappa_1, \Delta'_1$. By induction, there exists $\Gamma'$, $\Delta'$, $\kappa'$ and $\kappa''$ such that $\Gamma' : \kappa'' \to \nu \mid a : \kappa', \Delta'$. Then, by rule (Q), $\Gamma' \vdash_R \mu a. [\beta] M' : \kappa' \times \nu \mid \beta : \kappa'', \Delta'$. Take $\Gamma_3 = \Gamma'$, $\Delta_3 = \beta : \kappa'', \Delta'$, and $\delta_3 = \kappa'' \to \nu$.

$(M = \mu a. [\alpha] M')$: If $M \to_{lor} N$, then $N = \mu a. [\alpha] N'$ and $M' \to_{lor} N'$. Then there exists $\kappa$ such that $\delta_1 = \kappa \to \nu$, $\Delta_1 = a : \kappa, \Delta'_1$ and $\Gamma_1 \vdash_R N' : \kappa \to \nu \mid a : \kappa, \Delta'_1$. By induction, there exists $\Gamma'$, $\Delta'$,
and \( \kappa_1, \kappa_2 \) such that \( \Gamma' \vdash_{\text{r}} M' : \kappa_2 \rightarrow \nu | \alpha : \kappa_1, \Delta' \). Take \( \kappa' = \kappa_1 \land \kappa_2 \), then by weakening and rule \(( \leq )\), also \( \Gamma' \vdash_{\text{r}} M' : \kappa' \rightarrow \nu | \alpha : \kappa', \Delta' \). Then, by rule \(( \mu )\), \( \Gamma' \vdash_{\text{r}} \mu \alpha.[\alpha] M' : \kappa' \rightarrow \nu | \Delta' \). Take \( \Gamma_3 = \Gamma', \Delta_3 = \Delta' \), and \( \delta_3 = \kappa' \rightarrow \nu \).

We can now show that all strongly normalisable \( \lambda \mu \)-terms are typeable in the restricted system.

**Theorem 6.22 (Typeability of \( \mathcal{S} \mathcal{N} \)-Terms)** For all \( M \in \mathcal{S} \mathcal{N} \) there exist \( \Gamma \) and \( \Delta \) and a type \( \delta \) such that \( \Gamma \vdash_{\text{r}} M : \delta \mid \Delta \).

**Proof:** By induction on the maximum of the lengths of reduction sequences for a strongly normalisable term to its normal form (denoted by \( \#(M) \)).

\(((M) = 0)\): Then \( M \) is in normal form, and by Lemma 6.17, there exist \( \Gamma \) and \( \delta \) such that \( \Gamma \vdash_{\text{r}} M : \delta \mid \Delta \).

\(((M) \geq 1)\): Let \( M \rightarrow_{\text{lor}} N \) by contracting \( PQ \). Then \( \#(N) < \#(M) \), and \( \#(Q) < \#(M) \) (since \( Q \) is a proper subterm of a redex in \( M \)), so by induction there exists \( \Gamma_1, \Gamma_2, \Delta_1, \Delta_2, \delta_1 \) and \( \delta_2 \) such that \( \Gamma_1 \vdash_{\text{r}} N : \delta_1 \mid \Delta_1 \) and \( \Gamma_2 \vdash_{\text{r}} Q : \delta_2 \mid \Delta_2 \). Then, by Lemma 6.21, there exist \( \Gamma, \Delta \) and \( \delta \) such that \( \Gamma \vdash_{\text{r}} M : \delta \mid \Delta \).

7 Simply typed \( \lambda \mu \)-calculus and Intersection Types

In the previous sections we considered the \( \lambda \mu \)-calculus as a type-free calculus; in this section we will show that we can establish a connection between our intersection type assignment system and Parigot’s logical assignment system. In this section we will show that logical formulas translate into types of the appropriate sort, and moreover that this can be done in the restricted system of Section 6. Hence, the characterisation result carries on and can be used to establish the strong normalisation property of Parigot’s calculus, thought with different means.

The calculus presented in [43] was intended as the proof calculus of a fragment of classical logic. Logical formulas of the implicational fragment of the propositional calculus can be assigned to \( \lambda \mu \)-terms much as in the formulae-as-types paradigm of the Curry-Howard correspondence between typed \( \lambda \)-calculus and intuitionistic logic. With \( \lambda \mu \) Parigot created a multi-conclusion typing system. In the notation of [48], the derivable statements have the shape \( \Gamma \vdash M : A \mid \Delta \), where \( A \) is the main conclusion of the statement, expressed as the active conclusion, and \( \Delta \) contains the alternative conclusions, consisting of pairs of names and types; the left-hand context \( \Gamma \), as usual, contains pairs term variables and types, and represents the assumptions about free term variables of \( M \).

As with implicative intuitionistic logic, the reduction rules for the terms that represent proofs correspond to proof contractions; the difference is that the reduction rules for the \( \lambda \)-calculus are the logical reductions, i.e. deal with the elimination of a type constructor that has been introduced directly above. In addition to these, Parigot expressed also the structural rules, where elimination takes place for a type constructor that appears in one of the alternative conclusions (the Greek variable is the name given to a subterm): he therefore needed to express that the focus of the derivation (proof) changes, and this is achieved by extending the syntax with two new constructs \([\alpha]M\) and \( \mu \alpha.M \) that act as witness to deactivation and activation, which together move the focus of the derivation.

We use a version of Parigot’s logical system as presented in [43], which is equivalent to the original one if just terms (so not also proper commands, i.e. elements of \( \text{Cmd} \)) are typed. This implies that the rule for \( \bot \) does not need to be taken into account.

We briefly recall Parigot’s first-order type assignment system, that we call the Simply_TYPED \( \lambda \mu \)-calculus.
Definition 7.1 (Parigot’s Simply Typed $\lambda\mu$-calculus)

i) The set LF of Logical Formulas is defined by the following grammar:

$$A, B ::= \varphi | A \rightarrow B$$

where $\varphi$ ranges over an infinite, denumerable set of Proposition (Type) Variables.

ii) Judgments are of the form $\Pi \vdash \tau : A | \Sigma$, where $M \in \text{Trm}$; $\Pi$ and $\Sigma$ are finite mappings from \text{Var} and \text{Name}, respectively, to formulas, and are normally written as finite sets of pairs of term variables and formulas and of names and formulas respectively, as in $\Pi = \{x_1:A_1, \ldots, x_n:A_n\}$ and $\Sigma = \{a_1:B_1, \ldots, a_m:B_m\}$.

iii) The inference rules of this system are:

$$(Ax): \quad \Pi, x : A \vdash x : A | \Sigma \quad (\mu_1): \quad \Pi \vdash M : A | \alpha : A, \Sigma \quad (\mu_2): \quad \Pi \vdash \mu \alpha . [\beta] M : A | \beta : B, \Sigma$$

$$(\rightarrow I): \quad \Pi, x : A \vdash M : B | \Sigma \quad (\rightarrow E): \quad \Pi \vdash M : A \rightarrow B | \Sigma \quad \Pi \vdash N : A | \Sigma$$

We write $\Gamma \vdash \tau : A | \Sigma$ to denote that this judgement is derivable in this system.

Through the Curry-Howard correspondence (formulas as types and proofs as terms), the underlying logic of this system is the minimal classical logic ([4]).

Comparing Parigot’s system with ours we observe that rules $\rightarrow I$ and $\rightarrow E$ bear some similarity with $\text{(Abs)}$ and $\text{(App)}$, and rules $\mu_1$ and $\mu_2$ are similar to a combination of $\mu$ and $\text{(Cmd)}$:

Lemma 7.2 The following rules are derivable in the system of Definition 4.2 (and in the restricted system as well):

$$\Gamma \vdash M : \kappa \rightarrow \nu \mid \alpha : \kappa, \Delta \quad \Gamma \vdash M : \kappa' \rightarrow \nu \mid \alpha : \kappa, \beta : \kappa', \Delta$$

$$(\mu_1): \quad \Gamma \vdash \mu \alpha . [\beta] M : \kappa \rightarrow \nu \mid \beta : \kappa', \Delta$$

$$(\mu_2): \quad \Gamma \vdash \mu \alpha . [\beta] M : \kappa \rightarrow \nu \mid \beta : \kappa', \Delta$$

Proof: Consider the derivations:

$$\Gamma \vdash M : \kappa \rightarrow \nu \mid \alpha : \kappa, \Delta \quad (\mu_1) \quad \Gamma \vdash M : \kappa' \rightarrow \nu \mid \alpha : \kappa, \beta : \kappa', \Delta \quad (\mu_2)$$

As an example illustrating the fact that the system for pure $\lambda\mu$ is more expressive than the simply typed $\lambda$-calculus, we consider the following proof of Peirce’s Law, which we take from [41]:

$$\vdash \lambda x . \mu \alpha . [\alpha] (x (\lambda y . \mu \beta . [\alpha] y)) : (A \rightarrow B) \rightarrow A$$

We observe that the term $\lambda x . \mu \alpha . [\alpha] (x (\lambda y . \mu \beta . [\alpha] y))$ is typable in the restricted type system (and hence in the full system as well) by a derivation with the very same structure. Indeed,
let $\delta_{A \to B} \triangleq \kappa_{A \to B} \to v$, where $\kappa_{A \to B} \triangleq (\kappa_A \to v) \times \kappa_B$ and $\kappa_A, \kappa_B \in L^\kappa_C$ are arbitrary; set further $\kappa_{(A \to B) \to A} \triangleq \delta_{A \to B} \times \kappa_A$ and $\delta_{(A \to B) \to A} \triangleq \kappa_{(A \to B) \to A} \to v$ ($\triangleq \delta_{A \to B} \times \kappa_A$), then:

The translation functions $\lambda \mu L^\kappa_C$ are defined by:

$$\phi^C \triangleq v \times \omega$$

$$(A \to B)^C \triangleq (A^C \to v) \times B^C$$

$$A^D \triangleq A^C \to v$$

We extend these mappings to bases and name contexts by: $\Pi^D = \{ x : A^D \mid x : A \in \Pi \}$ and $\Sigma^C = \{ \alpha : A^C \mid \alpha : A \in \Sigma \}$.

It is straightforward to show that the above translations are well defined.

**Theorem 7.4 (Derivability preservation)** If $\Pi \vdash_T M : A \mid \Sigma$ then $\Pi^D \vdash M : A^D \mid \Sigma^C$.

**Proof:** Each rule of the simply-typed $\lambda \mu$-calculus has a corresponding one in the restricted intersection type system; hence it suffices to show that rules are preserved when translating formulas into types. The case of $(Ax)$ is straightforward.

$$(\to I):$$

$$\Pi^D, x : A^D \vdash M : B^D \mid \Sigma^C$$

$$\Pi^D \vdash \lambda x : M : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D, x : A^C \to v \vdash M : B^C \to v \mid \Sigma^C$$

$$\Pi^D \vdash \lambda x : M : ((A^C \to v) \times B^C) \to v \mid \Sigma^C$$

$$\Pi^D \vdash M : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D \vdash N : A^D \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : A^C \to v \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A^C \to v) \times B^C \to v \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D \vdash MN : B^C \to v \mid \Sigma^C$$

$$(\to E):$$

$$\Pi^D \vdash M : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D \vdash N : A^D \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : A^C \to v \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A^C \to v) \times B^C \to v \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D \vdash MN : B^C \to v \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A \to B)^D \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : A^C \to v \mid \Sigma^C$$

$$\Pi^D \vdash M \mid N : (A^C \to v) \times B^C \to v \mid \Sigma^C$$

$$(\mu_1):$$

$$\Pi^D \vdash M : A^D \mid \alpha : A^C, \Sigma^C$$

$$\Pi^D \vdash \alpha : A^C, \Sigma^C$$

$$\Pi^D \vdash M : A^C \to v \mid \Sigma^C$$

$$(\mu_2):$$

$$\Pi^D \vdash M : B^D \mid a : A^C, \beta : B^C, \Sigma^C$$

$$\Pi^D \vdash M \mid a : A^C, \beta : B^C, \Sigma^C$$

Strong normalisation of typeable terms in Parigot’s simply-typed $\lambda \mu$-calculus (first proved in [43]) now follows as a corollary of our characterisation result.

**Corollary 7.5 (Strong Normalisability of Parigot’s Simply Typed $\lambda \mu$-calculus)** If $\Pi \vdash_T M : A \mid \Sigma$, then $M \in SN$.

**Proof:** By Theorem 7.4 and Lemma 7.2, if $\Pi \vdash_T M : A \mid \Sigma$ then $\Pi^D \vdash_{\kappa} M : A^D \mid \Sigma^C$. That $M$ is $SN$ now follows from Theorem 6.22.
We end this section by observing that negation can be added to the syntax of logical formulas without disrupting the correspondence with the (restricted) intersection types. Let \( \bot \) be added to the formula syntax as a special atom for falsehood. Then consider the logical system which is obtained from Definition 7.1 by replacing rules (\( \mu_1 \)) and (\( \mu_2 \)) by:

\[
\begin{align*}
\Gamma \vdash b : \bot & \mid a : A, \Sigma & \text{(Activate)} \\
\Gamma \vdash \mu a. b : A & \mid \Sigma & \text{(Passivate)} \\
\end{align*}
\]

The resulting system is the same as in [16], and in [45] § 3.1 but with the two contexts instead of the single one. The only difference is that in rule (Passivate) we require the assumption \( a : A \) in the name contexts, which is just added to the conclusion in both [16] and [45] since there contraction and weakening are implicitly assumed.

In this new system, the rules (\( \mu_1 \)) and (\( \mu_2 \)) are admissible. By defining \( \neg A \triangleq A \to \bot \) and considering all the formulas in the \( \Sigma \) context as negated, rule (Activate) corresponds to the reductio ad absurdum rule of classical logic. On the other hand rule (Passivate) can be read as the \( \neg \)-elimination rule, saying that from \( A \) and \( \neg A \) we get falsity.

To interpret \( \bot \) into our intersection type system we need to keep track of the contradiction from which it arises; therefore we write the slightly different rule:

\[
\begin{align*}
\Gamma \vdash M : A & \mid a : A, \Sigma & \text{(Passivate')} \\
\end{align*}
\]

where \( \bot_A \) is a new constant for each formula \( A \). Now, by adding

\[
\bot_A^C \triangleq (A^C \to \nu) \times A^C
\]

to the translation, we obtain:

\[
\begin{align*}
\Pi^D \vdash c : \bot_C & \mid a : A^C, \Sigma^C & \Delta \\
\Pi^D \vdash \mu a. c : A^D & \mid \Sigma^C & \mu \text{ (\( \mu \))} \\
\Pi^D \vdash M : A^D & \mid a : A^C, \Sigma^C & \text{\( M \)} \\
\Pi^D \vdash [a]M : \bot_A & \mid a : A^C, \Sigma^C & \text{\( \text{Cmd} \)}
\end{align*}
\]

8 Related work

The starting point of the present work is [8], where a type assignment system for \( \lambda \mu \)-terms with intersection and union types was proved to be invariant under reduction and expansion. That system had no apparent semantical justification, that motivated the present new construction. W.r.t. [8] our system doesn’t use union types, and introduces product types for continuations, which is its characteristic. The introduction of product types is inspired to the continuation model in [49], which is the main source of this paper, together with [15], which is the first construction of a \( \lambda \)-model as a filter model. However, as explained in the Introduction and in the main body of the paper, we have followed the inverse path, from the model to the type system, building over [19] and [1].

Towards the end of [10] we conjectured that in an appropriate subsystem of the present one it should be possible to type exactly all strongly normalising \( \lambda \mu \)-terms, and established the result in [11], thought via a less elegant variant of our system than the restricted system in § 6. The first to state the characterisation result for the \( \lambda \)-calculus was Pottinger [46] using a notion of type assignment similar to the intersection system of [17, 20], but extended in that it is also closed for \( \nu \)-reduction, which is equivalent to adding a co-variant type
inclusion relation; in particular, this is a system that and is defined without the type constant \( \omega \). However, to show that all typeable terms are strongly normalisable, [46] only suggests a proof using Tait’s computability technique [50]. A detailed proof, using computability, in the context of the \( \omega \)-free BCD-system [15] is given in [5]; to establish the same result saturated sets are used by Krivine in [36] (chapter 4), in Ghilezan’s survey [30], and in [9].

The converse of that result, namely the property that all strongly normalisable terms are typeable has proven to be more elusive: it has been claimed in many papers but not shown in full (we mention [46, 5, 30]); in particular, the proof for the property that type assignment is closed for subject expansion (the converse of subject reduction) is dubious. Subject expansion can only reliably be shown for left-most outermost reduction, which is used for the proofs in [36, 12, 6, 9], and our result follows that approach.

The translation in § 7 mapping simple types into our extension of intersection types is a form of negative translation; in [35] it is extended to the system in [8], therefore relating the original intersection and union type assignment system for \( \lambda \mu \) to ours.

The model in [49] is not a model of de Groote and Saurin \( \Lambda \mu \)-calculus, but a variant of it, dubbed “stream model” in [39], provides a sound interpretation of the extended calculus. Building over stream models, in [23] it has been proved that the same type theory, but with different rules of the type assignment, gives a finitary description of the model matching the reduction in the stronger sense that the approximation theorem holds.

In [40] it is considered an extension of \( \Lambda \mu \), called \( \Lambda \mu_{\text{cons}} \). A type assignment for \( \Lambda \mu_{\text{cons}} \) based on [23] type system is proposed, and subject reduction and strong normalization of the reduction on the typed \( \Lambda \mu_{\text{cons}} \) are proved. Moreover in [38] the same system is shown to enjoy Friedman’s theorem.

Conclusions

We have presented a filter model for the \( \lambda \mu \)-calculus which is an instance of Streicher and Reus’s continuation model, and a type assignment system such that the set of types that can be given to a term coincides with its denotation in the model. The type theory and the assignment system can be viewed as the logic for reasoning about the computational meaning of \( \lambda \mu \)-terms, much as it is the case for \( \lambda \)-calculus.

By restricting the assignment system to a subset of the intersection types we have obtained an assignment system where exactly the strongly normalising \( \lambda \mu \)-terms are typeable.

Finally we have given a translation of intersection types into logical formulas and proved that if a term is typeable in Parigot’s type assignment system for \( \lambda \mu \) then it is typeable by its translation in the restricted intersection type system. As a byproduct we have a new proof that proof-terms in the \( \lambda \mu \)-calculus are strongly normalising.

References


