Simply Typed Lambda-Calculus Modulo Type Isomorphisms☆

Alejandro Díaz-Caro[a,b], Gilles Dowek[b]

[a] *Université Paris Ouest, 200 avenue de la République, 92001 Nanterre, France*
[b] *INRIA, 23 avenue d'Italie, CS 81321, 75214 Paris Cedex 13, France*

Abstract

We define a simply typed, non-deterministic lambda-calculus where isomorphic types are equated. To this end, an equivalence relation is settled at the term level. We then provide a proof of strong normalisation modulo equivalence. Such a proof is a non-trivial adaptation of the reducibility method.

Keywords: typed lambda calculus, normalisation, type isomorphisms, deduction modulo

1. Introduction

Isomorphic structures are often identified in informal mathematics. For instance, the natural numbers and non negative integers are never distinguished, although they formally are different structures.

In typed lambda-calculus, in programming languages, and in proof theory, two types A and B are said to be isomorphic, when there exists two functions ϕ from A to B and ψ from B to A such that $\psi\phi\mathbf{r} = \mathbf{r}$ for all terms $\mathbf{r}$ of type A and $\phi\psi\mathbf{s} = \mathbf{s}$ for all terms $\mathbf{s}$ of type B.

In certain formalisms, some isomorphic types are identified. For instance, in Martin-Löf's type theory [22], in the Calculus of Constructions [8], and in Deduction modulo [16, 18], definitionally equivalent types are identified. For example, the types $x \subseteq y$, $x \in \mathcal{P}(y)$ and $\forall z\ (z \in x \Rightarrow z \in y)$ may be identified. However, definitional equality does not handle all the isomorphisms and, for example, the isomorphic types $A \wedge B$ and $B \wedge A$ are not usually identified: a term of type $A \wedge B$ does not have type $B \wedge A$.

Not identifying such types has many drawbacks, for instance if a library contains a proof of $B \wedge A$, a request on a proof of $A \wedge B$ fails to find it [26], if $\mathbf{r}$ and $\mathbf{s}$ are proofs of $(A \wedge B) \Rightarrow C$ and $B \wedge A$ respectively, it is not possible

☆A preliminary version of this work, including also polymorphism, was published as [13]. Such a version presents some of the ideas in this paper, but it does not deal with all the isomorphisms nor include a normalisation proof, the main result on the present work.

Email addresses: `alejandro@diaz-caro.info` (Alejandro Díaz-Caro), `gilles.dowek@inria.fr` (Gilles Dowek)

July 9, 2014

to apply $\mathbf{r}$ to $\mathbf{s}$ to get a proof of C, but we need to explicitly apply a function of type $(B \wedge A) \Rightarrow (A \wedge B)$ to $\mathbf{s}$ before we can apply $\mathbf{r}$ to this term. This has lead to several projects aiming at identifying in one way or another isomorphic types in type theory, for instance with the univalence axiom [27].

In [6], Bruce, Di Cosmo and Longo have provided a characterisation of isomorphic types in the simply typed λ-calculus extended with products and a unit type (see [12] for a concise overview on type isomorphisms, or [11] for a more comprehensive reference). In this work, we define a simply typed λ-calculus extended with products, where all the isomorphic types are identified, and we prove strong normalisation for this calculus. All the isomorphisms in such a setting, are consequences of the following four:

$$A \wedge B \equiv B \wedge A \tag{1}$$
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C \tag{2}$$
$$A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C) \tag{3}$$
$$(A \wedge B) \Rightarrow C \equiv A \Rightarrow B \Rightarrow C \tag{4}$$

For example, $A \Rightarrow B \Rightarrow C \equiv B \Rightarrow A \Rightarrow C$ is a consequence of (4) and (1).

Identifying types requires to also identify terms. For instance, if $\mathbf{r}$ is a closed term of type A, then $\lambda x^A.x$ is a term of type $A \Rightarrow A$, and $\langle \lambda x^A.x, \lambda x^A.x \rangle$ is a term of type $(A \Rightarrow A) \wedge (A \Rightarrow A)$, hence, by isomorphism (3), also a term of type $A \Rightarrow (A \wedge A)$. Thus the term $\langle \lambda x^A.x, \lambda x^A.x \rangle \mathbf{r}$ is a term of type $A \wedge A$. Although this term contains no redex, we do not want to consider it as normal, in particular because it is not an introduction. So we shall distribute the application over the comma, yielding the term $\langle (\lambda x^A.x)\mathbf{r}, (\lambda x^A.x)\mathbf{r} \rangle$ that finally reduces to $\langle \mathbf{r}, \mathbf{r} \rangle$. Similar considerations lead to the introduction of several equivalence rules on terms, one related to the isomorphism (1), the commutativity of the conjunction, $\langle \mathbf{r}, \mathbf{s} \rangle \rightleftarrows \langle \mathbf{s}, \mathbf{r} \rangle$; one related to the isomorphism (2), the associativity of the conjunction, $\langle \langle \mathbf{r}, \mathbf{s} \rangle, \mathbf{t} \rangle \rightleftarrows \langle \mathbf{r}, \langle \mathbf{s}, \mathbf{t} \rangle \rangle$; four to the isomorphism (3), the distributivity of implication with respect to conjunction, e.g. $\langle \mathbf{r}, \mathbf{s} \rangle \mathbf{t} \rightleftarrows \langle \mathbf{rt}, \mathbf{st} \rangle$; and one related to the isomorphisms (4), the currification, $\mathbf{rst} \rightleftarrows \mathbf{r}\langle \mathbf{s}, \mathbf{t} \rangle$. As our comma is associative and commutative, we will write it $+$. For instance, the second equivalence is rewritten $(\mathbf{r} + \mathbf{s}) + \mathbf{t} \rightleftarrows \mathbf{r} + (\mathbf{s} + \mathbf{t})$.

One of the main difficulties in the design of this calculus is the design of the elimination rule for the conjunction. A rule like "if $\mathbf{r} : A \wedge B$ then $\pi_1(\mathbf{r}) : A$", would not be consistent. Indeed, if A and B are two arbitrary types, $\mathbf{s}$ a term of type A and $\mathbf{t}$ a term of type B, then $\mathbf{s} + \mathbf{t}$ has both types $A \wedge B$ and $B \wedge A$, thus $\pi_1(\mathbf{s} + \mathbf{t})$ would have both type A and type B. The approach we have followed is to consider explicitly typed (Church style) terms, and parametrise the projection by the type: if $\mathbf{r} : A \wedge B$ then $\pi_A(\mathbf{r}) : A$ and the reduction rule is then that $\pi_A(\mathbf{s} + \mathbf{t})$ reduces to $\mathbf{s}$ if $\mathbf{s}$ has type A.

But this rule introduces some non-determinism. Indeed, in the particular case where A happens to be equal to B, then both $\mathbf{s}$ and $\mathbf{t}$ have type A and $\pi_A(\mathbf{s} + \mathbf{t})$ reduces both to $\mathbf{s}$ and to $\mathbf{t}$. Notice that although this reduction rule is non-deterministic, it preserves typing. This can be summarised by the slogan *"the subject reduction property is more important than the uniqueness of results"* [17].

Thus, our calculus is one of the many non-deterministic calculi in the sense of [5, 7, 9, 10, 24] and our pair-construction operator $+$ is also the parallel composition operator of a non deterministic calculus.

In non-deterministic calculi, the parallel composition is such that if $\mathbf{r}$ and $\mathbf{s}$ are two λ-terms, the term $\mathbf{r}+\mathbf{s}$ represents the computation that runs either $\mathbf{r}$ or $\mathbf{s}$ non-deterministically, that is such that $(\mathbf{r}+\mathbf{s})\mathbf{t}$ reduces either to $\mathbf{rt}$ or $\mathbf{st}$. In our case, $\pi_B((\mathbf{r}+\mathbf{s})\mathbf{t})$ is equivalent to $\pi_B(\mathbf{rt}+\mathbf{st})$, which reduces to $\mathbf{rt}$ or $\mathbf{st}$.

The calculus developed in this paper is also related to the algebraic calculi [1, 2], some of which have been designed to express quantum algorithms. In this case, the pair $\mathbf{s}+\mathbf{t}$ is not interpreted as a non-deterministic choice but as a superposition of two processes running $\mathbf{s}$ and $\mathbf{t}$. In this case the projection π is the related to the projective measurement, that is the only non deterministic operation. In such calculi, the distributivity rule $(\mathbf{r}+\mathbf{s})\mathbf{t} \rightleftarrows \mathbf{rt}+\mathbf{st}$ is seen as the pointwise definition of the sum of two functions.

The main difficulty in the normalisation proof seems to be related to the fact that our equivalence relation is "confusing", that is, it equates types with different main connectives such as the isomorphism (3). In [18], for instance, only the case of "non confusing" equivalence relations is considered: if two non atomic types are equivalent, they have the same head symbol and their arguments are equivalent. It is clear however that this restriction needs to be dropped if we want to identify, for instance, $A \Rightarrow (B \wedge C)$ and $(A \Rightarrow B) \wedge (A \Rightarrow C)$.

2. The Calculus

2.1. Formal Definition

In this section we present the calculus. We consider the following grammar of types, with one atomic type τ,

$$A, B, C, \dots \;::=\; \tau \mid A \Rightarrow B \mid A \wedge B \;.$$

The Isomorphisms (1), (2), (3) and (4) are made explicit by a congruent equivalence relation between types:

$$\begin{array}{rclcrcl} A \wedge B & \equiv & B \wedge A, & \qquad & A \Rightarrow (B \wedge C) & \equiv & (A \Rightarrow B) \wedge (A \Rightarrow C), \\ (A \wedge B) \wedge C & \equiv & A \wedge (B \wedge C), & & (A \wedge B) \Rightarrow C & \equiv & A \Rightarrow B \Rightarrow C. \end{array}$$

The set of terms is defined inductively by the grammar

$$\mathbf{r}, \mathbf{s}, \mathbf{t} \;::=\; x^A \mid \lambda x^A.\mathbf{r} \mid \mathbf{rs} \mid \mathbf{r}+\mathbf{s} \mid \pi_A(\mathbf{r})$$

The type system is given in Table 1. Typing judgements are of the form $\mathbf{r} : A$. A term $\mathbf{r}$ is typable if there exists a type A such that $\mathbf{r} : A$.

Because of the associativity property of $+$, the term $\mathbf{r}+(\mathbf{s}+\mathbf{t})$ is the same as the term $(\mathbf{r}+\mathbf{s})+\mathbf{t}$, so we can just express it as $\mathbf{r}+\mathbf{s}+\mathbf{t}$, that is, the

$$\frac{}{x^A : A}\ (ax) \qquad [A\equiv B]\ \frac{\mathbf{r} : A}{\mathbf{r} : B}\ (\equiv)$$

$$[(FV(\mathbf{r})\cup\{x^A\})^f]\ \frac{\mathbf{r} : B}{\lambda x^A.\mathbf{r} : A \Rightarrow B}\ (\Rightarrow_i) \qquad [FV(\mathbf{rs})^f]\ \frac{\mathbf{r} : A \Rightarrow B \quad \mathbf{s} : A}{\mathbf{rs} : B}\ (\Rightarrow_e)$$

$$[FV(\mathbf{r}+\mathbf{s})^f]\ \frac{\mathbf{r} : A \quad \mathbf{s} : B}{\mathbf{r}+\mathbf{s} : A \wedge B}\ (\wedge_i) \qquad \frac{\mathbf{r} : A \wedge B}{\pi_A(\mathbf{r}) : A}\ (\wedge_{e_n}) \qquad \frac{\mathbf{r} : A}{\pi_A(\mathbf{r}) : A}\ (\wedge_{e_1})$$

Table 1: The type system

parenthesis are meaningless, and pairs become lists. Hence, for completeness, we also allow to project a term with respect to its full type, that is, if $\mathbf{r} : A$, then $\pi_A(\mathbf{r})$ reduces to $\mathbf{r}$.

Since our reduction relation is oriented by the types, we follow [20, 25], and use a presentation of typed lambda-calculus without contexts, which makes the reduction rules clearer. To this end each variable occurrence is labelled by its type, such as. $\lambda x^A.x^A$ or $\lambda x^A.y^B$. We sometimes omit the labels and write, for example, $\lambda x^A.x$ for $\lambda x^A.x^A$. The type system forbids terms such as $\lambda x^A.x^B$ when A and B are different types, by imposing preconditions to the applicability of the typing rules. Let $S = \{x_1^{A_1}, \dots, x_n^{A_n}\}$ be a set of variables, we write S^f to express that this set is functional, that is when $x_i = x_j$ implies $A_i = A_j$. For example $\{x^A, y^{A\Rightarrow B}\}^f$, but not $\{x^A, x^{A\Rightarrow B}\}^f$. We write the preconditions of a typing rule, at its left.

The set $FV(\mathbf{r})$ of free variables of $\mathbf{r}$ is defined as usual in the λ-calculus (cf. [4, §2.1]). For example $FV(\lambda x^A.xy^Bz^C) = \{y^B, z^C\}$. We say that a term $\mathbf{r}$ is closed whenever $FV(\mathbf{r}) = \emptyset$.

Given two terms $\mathbf{r}$ and $\mathbf{s}$ we denote by $\mathbf{r}[\mathbf{s}/x]$ the term obtained by simultaneously substituting the term $\mathbf{s}$ for all the free occurrences of x in $\mathbf{r}$, subject to the usual proviso about renaming bound variables in $\mathbf{r}$ to avoid capture of the free variables of $\mathbf{s}$.

Each term of the language has a main type associated, which can be obtained from type annotations, and other types induced by type equivalences using rule $(\equiv)$.

Lemma 2.1. *If $\mathbf{r} : A$ and $\mathbf{r} : B$, then $A \equiv B$.*

Proof. Straightforward structural induction on the typing derivation of $\mathbf{r}$. □

The operational semantics of the calculus is given in Table 2, where there are two distinct relations between terms: a symmetric relation $\rightleftarrows$ and a reduction relation $\hookrightarrow$. We write $\hookrightarrow^*$ and $\rightleftarrows^*$ for the transitive and reflexive closure of $\hookrightarrow$ and $\rightleftarrows$ respectively. Note that $\rightleftarrows^*$ is an equivalence relation. We write $\rightsquigarrow$ for the relation $\hookrightarrow$ modulo $\rightleftarrows^*$ (i.e. $\mathbf{r} \rightsquigarrow \mathbf{s}$ iff $\mathbf{r} \rightleftarrows^* \mathbf{r}' \hookrightarrow \mathbf{s}' \rightleftarrows^* \mathbf{s}$), and $\rightsquigarrow^*$ for its reflexive and transitive closure.

Symmetric relation:

$$
\begin{array}{rcll}
\mathbf{r}+\mathbf{s} & \rightleftarrows & \mathbf{s}+\mathbf{r} & (\textsc{comm})\\
(\mathbf{r}+\mathbf{s})+\mathbf{t} & \rightleftarrows & \mathbf{r}+(\mathbf{s}+\mathbf{t}) & (\textsc{asso})\\
\lambda x^A.(\mathbf{r}+\mathbf{s}) & \rightleftarrows & \lambda x^A.\mathbf{r}+\lambda x^A.\mathbf{s} & (\textsc{dist}_{ii})\\
(\mathbf{r}+\mathbf{s})\mathbf{t} & \rightleftarrows & \mathbf{rt}+\mathbf{st} & (\textsc{dist}_{ie})\\
\pi_{A\Rightarrow B}(\lambda x^A.\mathbf{r}) & \rightleftarrows & \lambda x^A.\pi_B(\mathbf{r}) & (\textsc{dist}_{ei})\\
\text{If } \mathbf{r}:A\Rightarrow(B\wedge C),\ \pi_{A\Rightarrow B}(\mathbf{r})\mathbf{s} & \rightleftarrows & \pi_B(\mathbf{rs}) & (\textsc{dist}_{ee})\\
\mathbf{rst} & \rightleftarrows & \mathbf{r}(\mathbf{s}+\mathbf{t}) & (\textsc{curry})\\
\text{If } A\equiv B,\ \mathbf{r} & \rightleftarrows & \mathbf{r}[A/B] & (\alpha\text{–Types})\\
\text{If } \mathbf{r}=_\alpha \mathbf{s},\ \mathbf{r} & \rightleftarrows & \mathbf{s} & (\alpha\text{–Terms})
\end{array}
$$

Reductions:

$$
\begin{array}{rcll}
\text{If } \mathbf{s}:A,\ (\lambda x^A.\mathbf{r})\mathbf{s} & \hookrightarrow & \mathbf{r}[\mathbf{s}/x] & (\beta)\\
\text{If } \mathbf{r}:A,\ \pi_A(\mathbf{r}+\mathbf{s}) & \hookrightarrow & \mathbf{r} & (\pi_n)\\
\text{If } \mathbf{r}:A,\ \pi_A(\mathbf{r}) & \hookrightarrow & \mathbf{r} & (\pi_1)
\end{array}
$$

Table 2: Operational semantics

Each isomorphism taken as equivalence between types induces an equivalence between terms, given by relation $\rightleftarrows$. Four possible rules exist however for the isomorphism (3), depending of which distribution is taken into account: elimination or introduction of conjunction, and elimination or introduction of implication.

As an interesting remark, in a paper of Nipkow [23] some of these equivalences appear explicitly in the calculus. However his system includes also η and surjective pairing, which is not compatible with our system. Indeed, since a pair can be rewritten into an abstraction (rule $\textsc{dist}_{ii}$), it is possible to use η in an abstraction or surjective pairing in a pair in a way that produces loops. For example:

$$
\begin{aligned}
\lambda x^A.(\mathbf{r}+\mathbf{s}) &\rightleftarrows_{\textsc{dist}_{ii}} \lambda x^A.\mathbf{r}+\lambda x^A.\mathbf{s}\\
&\hookrightarrow_\eta \lambda y^A.((\lambda x^A.\mathbf{r}+\lambda x^A.\mathbf{s})y)\\
&\rightleftarrows_{\textsc{dist}_{ie}} \lambda y^A.((\lambda x^A.\mathbf{r})y+(\lambda x^A.\mathbf{s})y)\\
&\hookrightarrow_{\beta\times 2} \lambda y^A.(\mathbf{r}[y/x]+\mathbf{s}[y/x])\\
&\rightleftarrows_\alpha \lambda x^A.(\mathbf{r}+\mathbf{s})
\end{aligned}
$$

Hence, η and surjective pairing are not included in our calculus.

2.2. Examples

Example 2.2. Let $\mathbf{s} : A$ and $\mathbf{t} : B$. Then $\pi_{B\Rightarrow A}((\lambda x^{A\wedge B}.x)\mathbf{s})\mathbf{t} : A$,

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\lambda x^{A\wedge B}.x : (A\wedge B)\Rightarrow(A\wedge B)}{\lambda x^{A\wedge B}.x : A\Rightarrow B\Rightarrow(A\wedge B)}\,(\equiv) \qquad \mathbf{s} : A}{(\lambda x^{A\wedge B}.x)\mathbf{s} : B\Rightarrow(A\wedge B)}\,(\Rightarrow_e)}{(\lambda x^{A\wedge B}.x)\mathbf{s} : (B\Rightarrow A)\wedge(B\Rightarrow B)}\,(\equiv)}{\pi_{B\Rightarrow A}((\lambda x^{A\wedge B}.x)\mathbf{s}) : B\Rightarrow A}\,(\wedge_{e_n}) \qquad \mathbf{t} : B}{\pi_{B\Rightarrow A}((\lambda x^{A\wedge B}.x)\mathbf{s})\mathbf{t} : A}\,(\Rightarrow_e)
$$

The reduction is as follows:

$$
\begin{aligned}
\pi_{B\Rightarrow A}((\lambda x^{A\wedge B}.x)\mathbf{s})\mathbf{t} \rightleftarrows \pi_A((\lambda x^{A\wedge B}.x)\mathbf{s}\mathbf{t}) &\rightleftarrows \pi_A((\lambda x^{A\wedge B}.x)(\mathbf{s}+\mathbf{t}))\\
&\hookrightarrow \pi_A(\mathbf{s}+\mathbf{t}) \hookrightarrow \mathbf{s}
\end{aligned}
$$

Example 2.3. Let $\mathbf{r} : A$, $\mathbf{s} : B$. Then $(\lambda x^A.\lambda y^B.x)(\mathbf{r}+\mathbf{s}) \rightleftarrows (\lambda x^A.\lambda y^B.x)\mathbf{r}\mathbf{s} \hookrightarrow^* \mathbf{r}$. However, if $A \equiv B$, it is also possible to reduce in the following way

$$
\begin{aligned}
(\lambda x^A.\lambda y^B.x)(\mathbf{r}+\mathbf{s}) &\rightleftarrows (\lambda x^A.\lambda y^A.x)(\mathbf{r}+\mathbf{s})\\
&\rightleftarrows (\lambda x^A.\lambda y^A.x)(\mathbf{s}+\mathbf{r})\\
&\rightleftarrows (\lambda x^A.\lambda y^A.x)\mathbf{s}\mathbf{r}\\
&\hookrightarrow^* \mathbf{s}
\end{aligned}
$$

Hence, the encoding of the projector also behaves non-deterministically.

Example 2.4. Let $\mathbf{TF} = \lambda x^A.\lambda y^B.(x+y)$. It is easy to check that $\mathbf{TF} : A \Rightarrow B \Rightarrow (A\wedge B)$, and by rule $(\equiv)$ it also has the type $(A\Rightarrow B\Rightarrow A)\wedge(A\Rightarrow B\Rightarrow B)$. Therefore, $\pi_{A\Rightarrow B\Rightarrow A}(\mathbf{TF}) : A\Rightarrow B\Rightarrow A$ is well typed. In addition, if $\mathbf{r} : A$ and $\mathbf{s} : B$, we have $\pi_{A\Rightarrow B\Rightarrow A}(\mathbf{TF})\mathbf{r}\mathbf{s} : A$. Notice that $\pi_{A\Rightarrow B\Rightarrow A}(\mathbf{TF})\mathbf{r}\mathbf{s} \rightleftarrows \pi_{B\Rightarrow A}(\mathbf{TF}\mathbf{r})\mathbf{s} \rightleftarrows \pi_A(\mathbf{TF}\mathbf{r}\mathbf{s}) \hookrightarrow^* \pi_A(\mathbf{r}+\mathbf{s}) \hookrightarrow \mathbf{r}$.

Example 2.5. Let $\mathbf{T} = \lambda x^A.\lambda y^B.x$ and $\mathbf{F} = \lambda x^A.\lambda y^B.y$. Then $\mathbf{T}+\mathbf{F} : (A\Rightarrow B\Rightarrow A)\wedge(A\Rightarrow B\Rightarrow B)$, hence $\pi_{(A\Rightarrow B\Rightarrow A)\wedge(A\Rightarrow B\Rightarrow B)}(\mathbf{T}+\mathbf{F}+\mathbf{TF})$ reduces non-deterministically either to $\mathbf{T}+\mathbf{F}$ or to $\mathbf{TF}$. Moreover, notice that $\mathbf{T}+\mathbf{F}$ and $\mathbf{TF}$ are observationally equivalent (notation $\mathbf{T}+\mathbf{F} \sim \mathbf{TF}$), that is, $(\mathbf{T}+\mathbf{F})\mathbf{r}\mathbf{s}$ and $\mathbf{TF}\mathbf{r}\mathbf{s}$ both reduce to the same term $(\mathbf{r}+\mathbf{s})$. Hence in this very particular case, the non-deterministic choice does not play any role. We will come back to the encoding of booleans on this calculus on Section 4.3.

2.3. Subject Reduction

Our system has the subject reduction property, that is, the set of types assigned to a term is invariant under $\rightleftarrows$ and $\hookrightarrow$. Before proving subject reduction, we need the following results.

Lemma 2.6 (Generation Lemmas)**.**

1. *If* $x^A : B$*, then* $A \equiv B$*.*
2. *If* $\lambda x^A.\boldsymbol{r} : B$*, then* $B \equiv A \Rightarrow C$*,* $\boldsymbol{r} : C$ *and* $(FV(\boldsymbol{r}) \cup \{x^A\})^f$*.*
3. *If* $\boldsymbol{rs} : B$*, then* $\boldsymbol{r} : A \Rightarrow B$ *and* $\boldsymbol{s} : A$*.*
4. *If* $\boldsymbol{r} + \boldsymbol{s} : A$*, then* $A \equiv B \wedge C$ *with* $\boldsymbol{r} : B$ *and* $\boldsymbol{s} : C$*.*
5. *If* $\pi_A(\boldsymbol{r}) : B$*, then* $A \equiv B$ *and* $\boldsymbol{r} : B$ *or* $\boldsymbol{r} : B \wedge C$*.*

Proof. The proof follows by a straightforward induction on the typing derivation. To notice that such an induction is straightforward, it suffices to realize that the only typing rule not changing the term, is $(\equiv)$. For example, if $\lambda x^A.\mathbf{r} : B$, then the only way to type this term is either by rule $(\Rightarrow_i)$, and so $B = A \Rightarrow C$ for some, C, $\mathbf{r} : C$ and $(FV(\mathbf{r}) \cup \{x^A\})^f$, or by rule $(\equiv)$, and so the induction hypothesis applies and $B \equiv A \Rightarrow C$. □

In the remaining of this paper, we may use Lemma 2.6 implicitly.

Lemma 2.7 (Substitution Lemma)**.** *If* $\boldsymbol{r} : A$*,* $\boldsymbol{s} : B$ *and* $(FV(\boldsymbol{r}) \cup \{x^B\})^f$*, then* $\boldsymbol{r}[\boldsymbol{s}/x^B] : A$

Proof. We proceed by structural induction on $\mathbf{r}$.

- Let $\mathbf{r} = x^A$. Since $(FV(x^A) \cup \{x^B\})^f$ implies $A = B$, we have $\mathbf{s} : A$. Notice that $x^A[\mathbf{s}/x^A] = \mathbf{s}$, so $x^A[\mathbf{s}/x^B] : A$.

- Let $\mathbf{r} = y^A$. Notice that $y^A[\mathbf{s}/x^B] = y^A$, so $y^A[\mathbf{s}/x^B] : A$.

- Let $\mathbf{r} = \lambda y^C.\mathbf{r}'$. Then $A \equiv C \Rightarrow D$, with $\mathbf{r}' : D$. By the induction hypothesis $\mathbf{r}'[\mathbf{s}/x^B] : D$, and so, by rule $(\Rightarrow_i)$, $\lambda y^C.\mathbf{r}'[\mathbf{s}/x^B] : C \Rightarrow D$. Since $\lambda y^C.\mathbf{r}'[\mathbf{s}/x^B] = (\lambda y^A.\mathbf{r}')[\mathbf{s}/x^B]$, using rule $(\equiv)$, $(\lambda y^C.\mathbf{r}')[\mathbf{s}/x^B] : A$.

- Let $\mathbf{r} = \mathbf{r}_1\mathbf{r}_2$. Then $\mathbf{r}_1 : C \Rightarrow A$ and $\mathbf{r}_2 : C$. By the induction hypothesis $\mathbf{r}_1[\mathbf{s}/x^B] : C \Rightarrow A$ and $\mathbf{r}_2[\mathbf{s}/x^B] : C$, and so, by rule $(\Rightarrow_e)$, $(\mathbf{r}_1[\mathbf{s}/x^B])(\mathbf{r}_2[\mathbf{s}/x^B]) : A$. Since $(\mathbf{r}_1[\mathbf{s}/x^B])(\mathbf{r}_2[\mathbf{s}/x^B]) = (\mathbf{r}_1\mathbf{r}_2)[\mathbf{s}/x^B]$, we have $(\mathbf{r}_1\mathbf{r}_2)[\mathbf{s}/x^B] : A$.

- Let $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2$. Then $\mathbf{r}_1 : A_1$ and $\mathbf{r}_2 : A_2$, with $A \equiv A_1 \wedge A_2$. By the induction hypothesis $\mathbf{r}_1[\mathbf{s}/x^B] : A_1$ and $\mathbf{r}_2[\mathbf{s}/x^B] : A_2$, and so, by rule $(\wedge_i)$, $(\mathbf{r}_1[\mathbf{s}/x^B]) + (\mathbf{r}_2[\mathbf{s}/x^B]) : A_1 \wedge A_2$. Since $(\mathbf{r}_1[\mathbf{s}/x^B]) + (\mathbf{r}_2[\mathbf{s}/x^B]) = (\mathbf{r}_1 + \mathbf{r}_2)[\mathbf{s}/x^B]$, using rule $(\equiv)$, we have $(\mathbf{r}_1 + \mathbf{r}_2)[\mathbf{s}/x^B] : A$.

- Let $\mathbf{r} = \pi_A(\mathbf{r}')$. Then either $\mathbf{r}' : A$, or $\mathbf{r}' A \wedge C$. By the induction hypothesis, either $\mathbf{r}'[\mathbf{s}/x^B] : A$ or $\mathbf{r}'[\mathbf{s}/x^B] : A \wedge C$. In any case, either by rule $\wedge_{e_1}$ or $\wedge_{e_n}$, $\pi_A(\mathbf{r}'[\mathbf{s}/x^B]) : A$. Since $\pi_A(\mathbf{r}'[\mathbf{s}/x^B]) = \pi_A(\mathbf{r}')[\mathbf{s}/x^B]$, we have $\pi_A(\mathbf{r}')[\mathbf{s}/x^B] : A$. □

Theorem 2.8 (Subject reduction)**.** *If* $\boldsymbol{r} : A$ *and* $\boldsymbol{r} \hookrightarrow \boldsymbol{s}$ *or* $\boldsymbol{r} \rightleftarrows \boldsymbol{s}$ *then* $\boldsymbol{s} : A$*.*

Proof. We proceed by induction on the rewrite relation.

$\mathbf{r}+\mathbf{s} \rightleftarrows \mathbf{s}+\mathbf{r}$**:** If $\mathbf{r}+\mathbf{s} : A$, then $A \equiv A_1 \wedge A_2 \equiv A_2 \wedge A_1$, with $\mathbf{r} : A_1$ and $\mathbf{s} : A_2$. Then,

$$\dfrac{\dfrac{\mathbf{s} : A_2 \quad \mathbf{r} : A_1}{\mathbf{s}+\mathbf{r} : A_2 \wedge A_1}\,(\wedge_i)}{\mathbf{s}+\mathbf{r} : A}\,(\equiv)$$

$(\mathbf{r}+\mathbf{s})+\mathbf{t} \rightleftarrows \mathbf{r}+(\mathbf{s}+\mathbf{t})$**:** If $(\mathbf{r}+\mathbf{s})+\mathbf{t} : A$, then $A \equiv (A_1 \wedge A_2) \wedge A_3 \equiv A_1 \wedge (A_2 \wedge A_3)$, with $\mathbf{r} : A_1$, $\mathbf{s} : A_2$ and $\mathbf{t} : A_3$. Then,

$$\dfrac{\dfrac{\mathbf{r} : A_1 \quad \dfrac{\mathbf{s} : A_2 \quad \mathbf{t} : A_3}{\mathbf{s}+\mathbf{t} : A_2 \wedge A_3}\,(\wedge_i)}{\mathbf{r}+(\mathbf{s}+\mathbf{t}) : A_1 \wedge (A_2 \wedge A_3)}\,(\wedge_i)}{\mathbf{r}+(\mathbf{s}+\mathbf{t}) : A}\,(\equiv)$$

$\lambda x^A.(\mathbf{r}+\mathbf{s}) \rightleftarrows \lambda x^A.\mathbf{r}+\lambda x^A.\mathbf{s}$**:** If $\lambda x^A.(\mathbf{r}+\mathbf{s}) : B$, then $B \equiv A \Rightarrow (C_1 \wedge C_2) \equiv (A \Rightarrow C_1) \wedge (A \Rightarrow C_2)$, with $\mathbf{r} : C_1$ and $\mathbf{s} : C_2$. Then,

$$\dfrac{\dfrac{\dfrac{\mathbf{r} : C_1}{\lambda x^A.\mathbf{r} : A \Rightarrow C_1}\,(\Rightarrow_i) \quad \dfrac{\mathbf{s} : C_2}{\lambda x^A.\mathbf{s} : A \Rightarrow C_2}\,(\Rightarrow_i)}{\lambda x^A.\mathbf{r}+\lambda x^A.\mathbf{s} : (A \Rightarrow C_1) \wedge (A \Rightarrow C_2)}\,(\wedge_i)}{\lambda x^A.\mathbf{r}+\lambda x^A.\mathbf{s} : A}\,(\equiv)$$

$(\mathbf{r}+\mathbf{s})\mathbf{t} \rightleftarrows \mathbf{rt}+\mathbf{st}$**:** If $(\mathbf{r}+\mathbf{s})\mathbf{t} : A$, then $\mathbf{r}+\mathbf{s} : B \Rightarrow A$, and $\mathbf{t} : B$. Hence $A \equiv A_1 \wedge A_2$, with $\mathbf{r} : B \Rightarrow A_1$ and $\mathbf{s} : B \Rightarrow A_2$. Then,

$$\dfrac{\dfrac{\dfrac{\mathbf{r} : B \Rightarrow A_1 \quad \mathbf{t} : B}{\mathbf{rt} : A_1}\,(\Rightarrow_e) \quad \dfrac{\mathbf{s} : B \Rightarrow A_2 \quad \mathbf{t} : B}{\mathbf{st} : A_2}\,(\Rightarrow_e)}{\mathbf{rt}+\mathbf{st} : A_1 \wedge A_2}\,(\wedge_i)}{\mathbf{rt}+\mathbf{st} : A}\,(\equiv)$$

$\pi_{A\Rightarrow B}(\lambda x^A.\mathbf{r}) \rightleftarrows \lambda x^A.\pi_B(\mathbf{r})$**:** If $\pi_{A\Rightarrow B}(\lambda x^A.\mathbf{r}) : C$, then $C \equiv A \Rightarrow B$ and either $\lambda x^A.\mathbf{r} : A \Rightarrow (B \wedge D)$ or $\lambda x^A.\mathbf{r} : A \Rightarrow B$. Hence either $\mathbf{r} : B \wedge D$, or $\mathbf{r} : B$. In any case, either by rule $(\wedge_{e_1})$ or $(\wedge_{e_n})$, $\pi_B(\mathbf{r}) : B$, so

$$\dfrac{\dfrac{\pi_B(\mathbf{r}) : B}{\lambda x^A.\pi_B(\mathbf{r}) : A \Rightarrow B}\,(\Rightarrow_i)}{\lambda x^A.\pi_B(\mathbf{r}) : C}\,(\equiv)$$

$\pi_{A\Rightarrow B}(\mathbf{r})\mathbf{s} \rightleftarrows \pi_B(\mathbf{rs})$ **with** $\mathbf{r} : A \Rightarrow (B \wedge C)$**:** Then $\mathbf{s} : A$, and if $\pi_{A\Rightarrow B}(\mathbf{r})\mathbf{s} : D$, then $D \equiv B$.

$$\dfrac{\dfrac{\dfrac{\mathbf{r} : A \Rightarrow (B \wedge C) \quad \mathbf{s} : B}{\mathbf{rs} : B \wedge C}\,(\Rightarrow_e)}{\pi_B(\mathbf{rs}) : B}\,(\wedge_{e_n})}{\pi_B(\mathbf{rs}) : D}\,(\equiv)$$

$\mathbf{rst} \rightleftarrows \mathbf{r}(\mathbf{s}+\mathbf{t})$: If $\mathbf{rst} : A$, then $\mathbf{r} : B \Rightarrow C \Rightarrow A$, $\mathbf{s} : B$ and $\mathbf{t} : C$. Then,

$$\dfrac{\dfrac{\mathbf{r} : B \Rightarrow C \Rightarrow A}{\mathbf{r} : (B \wedge C) \Rightarrow A}\ (\equiv) \qquad \dfrac{\mathbf{s} : B \quad \mathbf{t} : C}{\mathbf{s}+\mathbf{t} : B \wedge C}\ (\wedge_i)}{\mathbf{r}(\mathbf{s}+\mathbf{t}) : A}\ (\Rightarrow_e)$$

$\mathbf{r} \rightleftarrows \mathbf{r}[A/B]$ with $A \equiv B$: If $\mathbf{r} : C$, since $C \equiv C[A/B]$, a straightforward induction on $\mathbf{r}$ allows to prove $\mathbf{r}[A/B] : C$.

$\mathbf{r} \rightleftarrows \mathbf{s}$ with $\mathbf{r} =_\alpha \mathbf{s}$: Straightforward induction on $\mathbf{r}$.

$(\lambda x^A.\mathbf{r})\mathbf{s} \hookrightarrow \mathbf{r}[\mathbf{s}/x]$ with $\mathbf{s} : A$: If $(\lambda x^A.\mathbf{r})\mathbf{s} : B$, then $\lambda x^A.\mathbf{r} : A \Rightarrow B$ and $\mathbf{s} : B$, and so $\mathbf{r} : B$ and $(FV(\mathbf{r}) \cup \{x^A\})^f$. Then by Lemma 2.7, $\mathbf{r}[\mathbf{s}/x^A] : B$.

$\pi_A(\mathbf{r}+\mathbf{s}) \hookrightarrow \mathbf{r}$ with $\mathbf{r} : A$: If $\pi_A(\mathbf{r}+\mathbf{s}) : B$, then $B \equiv A$, and so, by rule $(\equiv)$, $\mathbf{r} : B$.

$\pi_A(\mathbf{r}) \hookrightarrow \mathbf{r}$ with $\mathbf{r} : A$: If $\pi_A(\mathbf{r}) : B$, then $B \equiv A$, and so, by rule $(\equiv)$, $\mathbf{r} : B$. ☐

3. Strong Normalisation and Normal Forms

3.1. Strong Normalisation

Now we prove the strong normalisation property. In our setting, strong normalisation means that every reduction sequence fired from a typed term eventually terminates in a term in *normal form* modulo $\rightleftarrows^*$. In other words, no $\hookrightarrow$ reduction can be fired from it, even after $\rightleftarrows$ steps. Formally, we define $\mathrm{Red}(\mathbf{r}) = \{\mathbf{s} \mid \mathbf{r} \rightsquigarrow \mathbf{s}\}$. Hence, a term $\mathbf{r}$ is in normal form if $\mathrm{Red}(\mathbf{r}) = \emptyset$. When $\mathbf{r}$ is strongly normalising, we write $(\mathbf{r})\wr$ for the maximum number of $\rightsquigarrow$-steps needed to get a normal form of $\mathbf{r}$. We denote by SN the set of strongly normalising terms.

We use the notation $\overline{(A_i)}_{i=1}^n \Rightarrow B$ for $A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B$, with the convention that $\overline{(A_i)}_{i=1}^0 \Rightarrow B = B$. Finally, is $\vec{\mathbf{s}} = \mathbf{s}_1 \ldots \mathbf{s}_n$.

The normalisation proof is based in the representation lemma for types (Lemma 3.4), for which we define conjunction-free types as follows.

Definition 3.1. A conjunction-free type is a type without conjunctions, which can be produced by the following grammar:

$$S, R, T ::= \tau \mid S \Rightarrow R$$

The canonical form of a type, written $\mathsf{can}(A)$, is a conjunction of conjunction-free types, and it is defined inductively by

$$\mathsf{can}(\tau) = \tau \qquad \mathsf{can}(A \wedge B) = \mathsf{can}(A) \wedge \mathsf{can}(B)$$

$$\mathsf{can}(A \Rightarrow B) = \ \mathsf{let}\ \textstyle\bigwedge_{i=1}^n S_i = \mathsf{can}(A)\ \mathsf{in}\ \ \mathsf{let}\ \bigwedge_{j=1}^m R_j = \mathsf{can}(B)\ \mathsf{in}\ \ \bigwedge_{j=1}^m \overline{(S_i)}_{i=1}^n \Rightarrow R_j$$

Example 3.2. $\mathsf{can}((S_1 \wedge S_2) \Rightarrow (R_1 \wedge R_2)) = (S_1 \Rightarrow S_2 \Rightarrow R_1) \wedge (S_1 \Rightarrow S_2 \Rightarrow R_2)$

Lemma 3.3. *For any A, $A \equiv \mathsf{can}(A)$.*

Proof. We proceed by structural induction on A.

- Let $A = \tau$. Then $A = \mathsf{can}(A)$.
- Let $A = B \wedge C$. By the induction hypothesis $B \equiv \mathsf{can}(B)$ and $C \equiv \mathsf{can}(C)$, hence $A \equiv \mathsf{can}(B) \wedge \mathsf{can}(C) = \mathsf{can}(B \wedge C) = \mathsf{can}(A)$.
- Let $A = B \Rightarrow C$. By the induction hypothesis $B \equiv \bigwedge_{i=1}^{n} S_i$ and $C \equiv \bigwedge_{j=1}^{m} R_j$, so $A \equiv (\bigwedge_{i=1}^{n} S_i) \Rightarrow (\bigwedge_{j=1}^{m} R_j) \equiv \bigwedge_{j=1}^{m} (\bigwedge_{i=1}^{n} S_i) \Rightarrow R_j$ which is finally equivalent to $\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=1}^{n} \Rightarrow R_j$. □

Lemma 3.4. *For any A, $\mathsf{can}(A) = \bigwedge_{i=1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$, with $n \geq 1$ and $\forall i, m_i \geq 0$.*

Proof. We proceed by structural induction on A.

- $A = \tau$. Then take $n = 1$ and $m_1 = 0$.
- $A = B \wedge C$. By the induction hypothesis $\mathsf{can}(B) = \bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$ and $\mathsf{can}(C) = \bigwedge_{i=k+1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$, so $\mathsf{can}(B \wedge C) = \mathsf{can}(B) \wedge \mathsf{can}(C) = \bigwedge_{i=1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$
- $A = B \Rightarrow C$. By the induction hypothesis $\mathsf{can}(B) = \bigwedge_{i=1}^{n} \overline{(S_{ik})}_{k=1}^{m_i} \Rightarrow \tau$ and $\mathsf{can}(C) = \bigwedge_{j=1}^{o} \overline{(R_{ji})}_{i=1}^{p_j} \Rightarrow \tau$. Then we have that $\mathsf{can}(B \Rightarrow C) = \bigwedge_{j=1}^{o} \overline{(\overline{(S_{ik})}_{k=1}^{m_i} \Rightarrow \tau)}_{i=1}^{n} \Rightarrow \overline{(R_{ji})}_{i=1}^{p_j} \Rightarrow \tau = \bigwedge_{j=1}^{o} \overline{(T_{ji})}_{i=1}^{n+p_j} \Rightarrow \tau$, with $T_{ji} = \overline{(S_{ik})}_{k=1}^{m_l} \Rightarrow \tau$ if $i \leq n$, and $T_{ji} = R_{j(i-n)}$ if $i > n$. □

Definition 3.5. The interpretation of canonical types is given by

$$\left[\!\!\left[\bigwedge_{i=1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau\right]\!\!\right] = \left\{\mathbf{r} \mid \forall i, \left[\begin{array}{l} \mathbf{s}_{ij} \in [\![S_{ij}]\!] \\ j = 1, \dots, m_i \end{array} \text{ implies } \pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}\right]\right\}$$

where $n \geq 1$, and $m \geq 0$.

The interpretation of a general type A is defined by $[\![\mathsf{can}(A)]\!]$.

In order to prove that equivalent types have the same interpretation (Corollary 3.10), we need first the following intermediate results.

Definition 3.6. Let $\mathsf{can}^{lo}(A)$ be defined in a similar way than $\mathsf{can}(A)$ but where each time there is a conjunction, it is taken in quasi-lexicographic order (that is, strings are ordered firstly by length, and then lexicographically), and with the parenthesis associated to the right.

Example 3.7. Let $S_1 \leq S_2 \leq S_3$ and $R_1 \leq R_2$.

- $\mathsf{can}^{lo}((\tau \Rightarrow \tau) \wedge \tau) = \tau \wedge (\tau \Rightarrow \tau)$.
- $\mathsf{can}^{lo}(\bigwedge_{i=1}^{3} S_i) = S_1 \wedge (S_2 \wedge S_3)$.
- $\mathsf{can}^{lo}((S_2 \wedge S_3) \wedge S_1) = S_1 \wedge (S_2 \wedge S_3)$.
- $\mathsf{can}^{lo}((S_2 \wedge S_1) \Rightarrow R) = S_1 \Rightarrow S_2 \Rightarrow R$.
- $\mathsf{can}^{lo}((S_2 \wedge S_1) \Rightarrow (R_1 \wedge R_2)) = (S_1 \Rightarrow S_2 \Rightarrow R_1) \wedge (S_1 \Rightarrow S_2 \Rightarrow R_2)$.

Lemma 3.8. *If $A \equiv B$, then $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(B)$*

Proof. By induction on the equivalence relation.

- $A \wedge B \equiv B \wedge A$. Let $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(\bigwedge_{i=1}^{n} S_i)$ and $\mathsf{can}^{lo}(B) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} R_j)$. Then $\mathsf{can}^{lo}(A \wedge B) = \mathsf{can}^{lo}((\bigwedge_{i=1}^{n} S_i) \wedge (\bigwedge_{j=1}^{m} R_j) = \mathsf{can}^{lo}(B \wedge A)$.
- $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$. Analogous to the previous case.
- $A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C)$. Let $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(\bigwedge_{i=1}^{n} S_i)$, $\mathsf{can}^{lo}(B) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{k} R_j)$ and $\mathsf{can}^{lo}(C) = \mathsf{can}^{lo}(\bigwedge_{j=k+1}^{m} R_j)$, so $\mathsf{can}^{lo}(B \wedge C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{n} R_j)$. Hence, $\mathsf{can}^{lo}(A \Rightarrow (B \wedge C)) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=1}^{n} \Rightarrow R_j) = \mathsf{can}^{lo}(\mathsf{can}^{lo}(A \Rightarrow B) \wedge \mathsf{can}^{lo}(A \Rightarrow C)) = \mathsf{can}^{lo}((A \Rightarrow B) \wedge (A \Rightarrow C))$.
- $(A \wedge B) \Rightarrow C \equiv A \Rightarrow B \Rightarrow C$. Let $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(\bigwedge_{i=1}^{k} S_i)$, $\mathsf{can}^{lo}(B) = \mathsf{can}^{lo}(\bigwedge_{i=k+1}^{n} S_i)$ and $\mathsf{can}^{lo}(C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} R_j)$. Hence, $\mathsf{can}^{lo}((A \wedge B) \Rightarrow C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=1}^{n} \Rightarrow R_j)$.

  On the other hand, $\mathsf{can}^{lo}(B \Rightarrow C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=k+1}^{n} \Rightarrow R_j)$, so $\mathsf{can}^{lo}(A \Rightarrow B \Rightarrow C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=1}^{k} \Rightarrow \overline{(S_i)}_{i=k+1}^{n} \Rightarrow R_j)$, and notice that this is equal to $\mathsf{can}^{lo}(\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=1}^{n} \Rightarrow R_j) = \mathsf{can}^{lo}((A \wedge B) \Rightarrow C)$.
- Congruence:
  - Let $A \equiv B$ as a consequence of $A = B$. Trivial case.
  - Let $A \equiv C$ as a consequence of $A \equiv B$ and $B \equiv C$. By the induction hypothesis $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(B)$ and $\mathsf{can}^{lo}(B) = \mathsf{can}^{lo}(C)$, hence $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(C)$.
  - Let $A \Rightarrow C \equiv B \Rightarrow C$ as a consequence of $A \equiv B$. Let $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(\bigwedge_{i=1}^{n} S_i)$, and $\mathsf{can}^{lo}(C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} R_j)$. Then $\mathsf{can}^{lo}(A \Rightarrow C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=1}^{n} \Rightarrow R_j)$. By the induction hypothesis, we have that $\mathsf{can}^{lo}(B) = \mathsf{can}^{lo}(\bigwedge_{i=1}^{n} S_i)$, and hence $\mathsf{can}^{lo}(B \Rightarrow C) = \mathsf{can}^{lo}(\bigwedge_{j=1}^{m} \overline{(S_i)}_{i=1}^{n} \Rightarrow R_j) = \mathsf{can}^{lo}(A \Rightarrow C)$.
  - Let $A \wedge C \equiv B \wedge C$ as a consequence of $A \equiv B$. $\mathsf{can}^{lo}(A \wedge C) = \mathsf{can}^{lo}(\mathsf{can}^{lo}(A) \wedge \mathsf{can}^{lo}(C))$, which by the induction hypothesis, is equal to $\mathsf{can}^{lo}(\mathsf{can}^{lo}(B) \wedge \mathsf{can}^{lo}(C)) = \mathsf{can}^{lo}(B \wedge C)$. $\square$

Lemma 3.9. *$\forall A$, $[\![\mathsf{can}(A)]\!] = [\![\mathsf{can}^{lo}(A)]\!]$.*

Proof. Let $\mathsf{can}(A) = \bigwedge_{i=1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$. Hence, $[\![\mathsf{can}(A)]\!] = \{\mathbf{r} \mid \forall i$, if for $j = 1, \dots, m_i$, $\mathbf{s}_{ij} \in [\![S_{ij}]\!]$, then $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}\}$, which, by rule ($\alpha$–Types) is equal to $\{\mathbf{r} \mid \forall i$, if for $j = 1, \dots, m_i$, $\mathbf{s}_{ij} \in [\![S_{ij}]\!]$, then $\pi_{\mathsf{can}^{lo}(\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau)}(\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}\} = [\![\mathsf{can}^{lo}(A)]\!]$. ◻

Corollary 3.10. *If $A \equiv B$, then $[\![\mathsf{can}(A)]\!] = [\![\mathsf{can}(B)]\!]$.*

Proof. By Lemma 3.8, $A \equiv B$ implies $\mathsf{can}^{lo}(A) = \mathsf{can}^{lo}(B)$, and by Lemma 3.9, $[\![\mathsf{can}(A)]\!] = [\![\mathsf{can}^{lo}(A)]\!]$ for all A. Hence,

$$[\![\mathsf{can}(A)]\!] = [\![\mathsf{can}^{lo}(A)]\!] = [\![\mathsf{can}^{lo}(B)]\!] = [\![\mathsf{can}(B)]\!]$$ ◻

Lemma 3.11. *$\forall A$, $[\![\mathsf{can}(A)]\!] \neq \emptyset$.*

Proof. If $\vec{\mathbf{s}} \in \mathsf{SN}$, then both $x^A\vec{\mathbf{s}}$ and $\pi_B(x^A)\vec{\mathbf{s}}$ are in SN, hence for all A, $x^A \in [\![\mathsf{can}(A)]\!]$. ◻

Lemma 3.12. *$\forall A$, $[\![\mathsf{can}(A)]\!] \subseteq \mathsf{SN}$.*

Proof. Let $\mathsf{can}(A) = \bigwedge_{i=1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$ and $\mathbf{r} \in [\![\mathsf{can}(A)]\!]$. Assume $\mathbf{r} \notin \mathsf{SN}$, then for any $\vec{\mathbf{s}}$, $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\mathbf{r})\vec{\mathbf{s}} \notin \mathsf{SN}$. A contradiction. ◻

Lemma 3.13. *If $\boldsymbol{r} \in [\![\mathsf{can}(A)]\!]$ and $\boldsymbol{s} \in [\![\mathsf{can}(B)]\!]$, then $\boldsymbol{r} + \boldsymbol{s} \in [\![\mathsf{can}(A \wedge B)]\!]$.*

Proof. Let $\mathsf{can}(A) = \bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$ and $\mathsf{can}(B) = \bigwedge_{i=k+1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$, so $\mathsf{can}(A \wedge B) = \bigwedge_{i=1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$. Then we have that for all $i = 1, \dots, k$, if for $j = 1, \dots, m_i$, $\mathbf{t}_{ij} \in [\![S_{ij}]\!]$, then $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\mathbf{r})\vec{\mathbf{t}}_i \in \mathsf{SN}$ and for all $i = k+1, \dots, n$, if for $j = 1, \dots, m_i$, $\mathbf{t}_{ij} \in [\![S_{ij}]\!]$, then $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\mathbf{s})\vec{\mathbf{t}}_i \in \mathsf{SN}$. Therefore, for all $i = 1, \dots, n$, if for $j = 1, \dots, m_i$, $\mathbf{t}_{ij} \in [\![S_{ij}]\!]$, we have $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\mathbf{r} + \mathbf{s})\vec{\mathbf{t}}_i \in \mathsf{SN}$, so $\mathbf{r} + \mathbf{s} \in [\![\mathsf{can}(A \wedge B)]\!]$. ◻

Lemma 3.14. *If $\boldsymbol{r} \in \mathsf{SN}$, then $\pi_A(\boldsymbol{r}) \in \mathsf{SN}$.*

Proof. We proceed by induction on the sum of the number of steps to reach the normal form by any path starting on $\mathbf{r}$. The possible reduction from $\pi_A(\mathbf{r})$ are:

- $\pi_A(\mathbf{r}')$, and so the induction hypothesis applies,
- $\mathbf{r}'$, with $\mathbf{r}' : A$ and either $\mathbf{r} \rightleftarrows \mathbf{r}' + \mathbf{t}$ or just $\mathbf{r} \rightleftarrows \mathbf{r}'$. In any case, since $\mathbf{r} \in \mathsf{SN}$, then $\mathbf{r}' \in \mathsf{SN}$. ◻

Let σ be a substitution. We say that σ is adequate if for all x^A, $\sigma(x^A) \in [\![\mathsf{can}(A)]\!]$.

The following lemma shows that any adequate substitution applied to a term, is in the interpretation of the type of such term. This lemma, together with Lemma 3.12, implies that a typed term is strongly normalising (Theorem 3.16).

Lemma 3.15 (Adequacy)**.** *If $\boldsymbol{r} : A$ and σ adequate, then $\sigma\boldsymbol{r} \in [\![\mathsf{can}(A)]\!]$.*

Proof. We proceed by induction on the typing derivation.

- Let $x^A : A$ as a consequence of rule (ax). Since σ is adequate, $\sigma(x^A) \in [\![\mathsf{can}(A)]\!]$.
- Let $\mathbf{r} : B$ as a consequence of $\mathbf{r} : A$, $A \equiv B$ and rule $(\equiv)$. By the induction hypothesis $\forall\sigma$ adequate, $\sigma\mathbf{r} \in [\![\mathsf{can}(A)]\!]$, so by Lemma 3.10, $\sigma\mathbf{r} \in [\![\mathsf{can}(B)]\!]$.
- Let $\lambda x^A.\mathbf{r} : A \Rightarrow B$ as a consequence of $\mathbf{r} : B$ and rule $(\Rightarrow_i)$. Let $\mathsf{can}(A) = \bigwedge_{i=1}^n \overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau$ and $\mathsf{can}(B) = \bigwedge_{j=1}^m \overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau$. By the induction hypothesis, $\sigma\mathbf{r} \in [\![\mathsf{can}(B)]\!]$, that is, for all j, if $\mathbf{s}_{jk} \in [\![R_{jk}]\!]$, for $k = 1, \dots, h_j$, then $\pi_{\overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_j \in \mathsf{SN}$. Notice that $\sigma\lambda x^A.\mathbf{r} = \lambda x^A.\sigma\mathbf{r}$. We must show that

$$\lambda x^A.\sigma\mathbf{r} \in \left[\!\!\left[\bigwedge_{j=1}^m \overline{(\overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau)}_{i=1}^n \Rightarrow \overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau\right]\!\!\right]$$

  that is, we must show that $\forall j$, if for $i = 1, \dots, n$, $\mathbf{t}_i \in \left[\!\!\left[\overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau\right]\!\!\right]$ and for $k = 1, \dots, h_j$, $\mathbf{s}_{jk} \in [\![R_{jk}]\!]$, then

$$\pi_{\overline{(\overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau)}_{i=1}^n \Rightarrow \overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}(\lambda x^A.\sigma\mathbf{r})\vec{\mathbf{t}}\vec{\mathbf{s}}_j \in \mathsf{SN}$$

  By Lemma 3.3, $A \equiv \mathsf{can}(A)$, so

$$\begin{aligned}
&\pi_{\overline{(\overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau)}_{i=1}^n \Rightarrow \overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}(\lambda x^A.\sigma\mathbf{r})\vec{\mathbf{t}}\vec{\mathbf{s}}_j \\
&\rightleftarrows \pi_{\overline{(\overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau)}_{i=1}^n \Rightarrow \overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}(\lambda x^{\bigwedge_{i=1}^n \overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau}.\sigma\mathbf{r})\vec{\mathbf{t}}\vec{\mathbf{s}}_j \\
&\rightleftarrows^* \pi_{\overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}((\lambda x^{\bigwedge_{i=1}^n \overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau}.\sigma\mathbf{r})\vec{\mathbf{t}})\vec{\mathbf{s}}_j \\
&\rightleftarrows^* \pi_{\overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}((\lambda x^{\bigwedge_{i=1}^n \overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau}.\sigma\mathbf{r})(\sum_{i=1}^n \mathbf{t}_i))\vec{\mathbf{s}}_j
\end{aligned}$$

  Since $\mathbf{r}, \vec{\mathbf{t}}, \vec{\mathbf{s}}_i \in \mathsf{SN}$, we proceed by induction on the sum of the number of steps to reach the normal form of each of these terms. The possible reductions fired from $\pi_{\overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}((\lambda x^{\bigwedge_{i=1}^n \overline{(S_{il})}_{l=1}^{p_i} \Rightarrow \tau}.\sigma\mathbf{r})(\sum\limits_{i=1}^n \mathbf{t}_i))\vec{\mathbf{s}}_j$ are:

  - reducing one of $\mathbf{r}, \mathbf{t}_i, \mathbf{s}_{jk}$, then the induction hypothesis applies,
  - $\pi_{\overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}(\sigma\mathbf{r}[\sum_{i=1}^n \mathbf{t}_i/x])\vec{\mathbf{s}}_j$. Then consider $\sigma' = \sigma, [\sum_{i=1}^n \mathbf{t}_i/x]$. By Lemma 3.13, σ' is adequate, hence

$$\pi_{\overline{(R_{jk})}_{k=1}^{h_j} \Rightarrow \tau}(\sigma\mathbf{r}[\sum_{i=1}^n \mathbf{t}_i/x])\vec{\mathbf{s}}_j = \pi_{\overline{(R_j)}_{\cdot=1}^{h_j} \Rightarrow \tau}(\sigma'\mathbf{r})\vec{\mathbf{s}}_j \in \mathsf{SN}$$

- Let $\mathbf{rs} : B$ as a consequence of $\mathbf{r} : A \Rightarrow B$, $\mathbf{s} : A$ and rule $(\Rightarrow_e)$. Let $\mathsf{can}(A) = \bigwedge_{i=1}^{n} \overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau$ and $\mathsf{can}(B) = \bigwedge_{j=1}^{o} \overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau$, then $\mathsf{can}(A \Rightarrow B) = \bigwedge_{j=1}^{m} \overline{(\overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau)}_{i=1}^{n} \Rightarrow \overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau$. By the induction hypothesis, if σ adequate, $\sigma\mathbf{r} \in [\![\mathsf{can}(A \Rightarrow B)]\!]$ and $\sigma\mathbf{s} \in [\![\mathsf{can}(A)]\!]$, that is, for $j = 1, \dots, m$, if for $i = 1, \dots, n$, $\mathbf{t}_{ji} \in \left[\!\!\left[\overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau\right]\!\!\right]$ and for $k = 1, \dots, p_k$, $\mathbf{u}_{jk} \in [\![R_{jk}]\!]$, then

$$\pi_{\overline{(\overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau)}_{i=1}^{n} \Rightarrow \overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{t}}_j\vec{\mathbf{u}}_j \in \mathsf{SN}$$

  Remark that

$$\pi_{\overline{(\overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau)}_{i=1}^{n} \Rightarrow \overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{t}}_j\vec{\mathbf{u}}_j \rightleftarrows^* \pi_{\overline{(\overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau)}_{i=1}^{n} \Rightarrow \overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau}(\sigma\mathbf{r})(\sum_{i=1}^{n}\mathbf{t}_{ji})\vec{\mathbf{u}}_j$$

  hence since $\sigma\mathbf{r} \in [\![\mathsf{can}(A \Rightarrow B)]\!]$, by Lemma 3.13, if $\sum_{i=1}^{n}\mathbf{t}_{ji} \in [\![\mathsf{can}(A)]\!]$, then $\pi_{\overline{(\overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau)}_{i=1}^{n} \Rightarrow \overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau}(\sigma\mathbf{r})(\sum_{i=1}^{n}\mathbf{t}_{ji})\vec{\mathbf{u}}_j \in \mathsf{SN}$. Since $\sigma\mathbf{s} \in [\![\mathsf{can}(A)]\!]$, we have that $\pi_{\overline{(\overline{(S_{ih})}_{h=1}^{m_i} \Rightarrow \tau)}_{i=1}^{n} \Rightarrow \overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau}(\sigma\mathbf{r})\sigma\mathbf{s}\vec{\mathbf{u}}_j$ is $\rightleftarrows^*$-equivalent to $\pi_{\overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau}((\sigma\mathbf{r})\sigma\mathbf{s})\vec{\mathbf{u}}_j = \pi_{\overline{(R_{jk})}_{k=1}^{p_j} \Rightarrow \tau}(\sigma(\mathbf{rs}))\vec{\mathbf{u}}_j \in \mathsf{SN}$, and so $\sigma(\mathbf{rs}) \in [\![\mathsf{can}(B)]\!]$.

- Let $\mathbf{r} + \mathbf{s} : A \wedge B$ as a consequence of $\mathbf{r} : A$, $\mathbf{s} : B$ and rule $(\wedge_i)$. By the induction hypothesis, $\forall\sigma$ adequate, $\sigma\mathbf{r} \in [\![\mathsf{can}(A)]\!]$ and $\sigma\mathbf{s} \in [\![\mathsf{can}(B)]\!]$, hence by Lemma 3.13, $\sigma\mathbf{r} + \sigma\mathbf{s} \in [\![\mathsf{can}(A \wedge B)]\!]$. Notice that $\sigma\mathbf{r} + \sigma\mathbf{s} = \sigma(\mathbf{r} + \mathbf{s})$.

- Let $\pi_A(\mathbf{r}) : A$ as a consequence of $\mathbf{r} : A \wedge B$ and rule $(\wedge_{e_n})$. By the induction hypothesis, $\forall\sigma$ adequate, $\sigma\mathbf{r} \in [\![\mathsf{can}(A \wedge B)]\!]$. Let $\mathsf{can}(A) = \bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$ and $\mathsf{can}(B) = \bigwedge_{i=k+1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$, then $\sigma\mathbf{r} \in [\![\mathsf{can}(A \wedge B)]\!]$ means that $\forall i$, if $\forall j$, $\mathbf{s}_{ij} \in [\![S_{ij}]\!]$, then $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}$.

  We need to prove that

$$\pi_{\overline{(S_{1j})}_{j=1}^{m_1} \Rightarrow \tau}(\pi_{\bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\sigma\mathbf{r}))\vec{\mathbf{s}}_i \rightleftarrows \pi_\tau(\pi_{\bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_i) \in \mathsf{SN}$$

  By Lemma 3.14, it suffices to prove $\pi_{\bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}$. If $k = 1$, then we are done. In other case, we proceed by induction on the sum of the number of steps to reach the normal form of $\sigma\mathbf{r}$ and $\vec{\mathbf{s}}_i$. The possible reductions fired from $\pi_{\bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_1} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_i$ are:

  - reducing one of $\sigma\mathbf{r}, \mathbf{s}^1, \dots, \mathbf{s}^j$, then the induction hypothesis applies,
  - $\mathbf{r}'\vec{\mathbf{s}}_i$, with $\mathbf{r}' : \bigwedge_{i=1}^{k} \overline{(S_{ij})}_{j=1}^{m_1} \Rightarrow \tau$ and $\sigma\mathbf{r} = \mathbf{r}' + \mathbf{t}$ or just $\sigma\mathbf{r} = \mathbf{r}'$. Since $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_i \rightleftarrows \pi_\tau(\sigma\mathbf{r}\vec{\mathbf{s}}_i) \in \mathsf{SN}$ which is equal either to $\pi_\tau((\mathbf{r}'+\mathbf{t})\vec{\mathbf{s}}_i) \in \mathsf{SN}$, then we have $\pi_\tau(\mathbf{r}'\vec{\mathbf{s}}_i + \mathbf{t}\vec{\mathbf{s}}_i) \in \mathsf{SN}$, or to $\pi_\tau(\mathbf{r}'\vec{\mathbf{s}}_i) \in \mathsf{SN}$, in any case we can conclude $\mathbf{r}'\vec{\mathbf{s}}_i \in \mathsf{SN}$.

– Any other reduction involving first using $\textsc{dist}_{ee}$-rule, are analogous to the previous case.

- Let $\pi_A(\mathbf{r}) : A$ as a consequence of $\mathbf{r} : A$ and rule $(\wedge_{e_1})$. By the induction hypothesis $\sigma\mathbf{r} \in [\![\mathsf{can}(A)]\!]$, that is, if $\mathsf{can}(A) = \bigwedge_{i=1}^{n} \overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau$, for all i, if for all j, $\mathbf{s}_{ij} \in [\![\mathsf{can}(S_{ij})]\!]$, then $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}$. Notice that since $\sigma\mathbf{r} : A$, we have $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\sigma\mathbf{r})\vec{\mathbf{s}}_i \rightleftarrows^* \pi_\tau((\sigma\mathbf{r})\vec{\mathbf{s}}_i)$, hence $(\sigma\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}$, so $\pi_A(\sigma\mathbf{r})\vec{\mathbf{s}}_i \in \mathsf{SN}$, which implies $\pi_{\overline{(S_{ij})}_{j=1}^{m_i} \Rightarrow \tau}(\pi_A(\sigma\mathbf{r}))\vec{\mathbf{s}}_i \in \mathsf{SN}$. ☐

Now we can prove strong normalisation as a corollary of Lemma 3.15.

Theorem 3.16 (Strong normalisation)**.** *If $\boldsymbol{r} : A$, then $\boldsymbol{r} \in \mathsf{SN}$.*

Proof. If $\mathbf{r} : A$, by Lemma 3.15, for all σ adequate, $\sigma\mathbf{r} \in [\![\mathsf{can}(A)]\!]$. Take $\sigma = \mathit{identity}$, and notice that it is adequate (cf. proof of Lemma 3.11), then $\sigma\mathbf{r} = \mathbf{r} \in [\![\mathsf{can}(A)]\!]$, which by Lemma 3.12, is in SN. ☐

3.2. Characterisation of Typed Closed Normal Forms

In this section, we give a characterisation of typed closed normal forms (Theorem 3.18), for which we need the following auxiliary result.

Lemma 3.17. *If $\boldsymbol{r} : A \wedge B$ and $FV(\boldsymbol{r}) = \emptyset$, then $\pi_A(\boldsymbol{r})$ reduces using at least one π_n reduction.*

Proof. We proceed by structural induction on $\mathbf{r}$.

- If $\mathbf{r} = \lambda x^C.\mathbf{s}$ then $A \equiv C \Rightarrow A'$ and $B \equiv C \Rightarrow B'$, with $\mathbf{s} : A' \wedge B'$. So, $\pi_{C\Rightarrow A'}(\lambda x^C.\mathbf{s}) \rightleftarrows \lambda x^C.\pi_{A'}(\mathbf{s})$, which by the induction hypothesis reduces using at least one π_n reduction.

- If $\mathbf{r} = \mathbf{r}_1\mathbf{r}_2$ then $\mathbf{r}_1 : C \Rightarrow (A \wedge B)$, so $\pi_A(\mathbf{r}_1\mathbf{r}_2) \rightleftarrows \pi_{C\Rightarrow A}(\mathbf{r}_1)\mathbf{r}_2$. We conclude with the induction hypothesis.

- If $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2$ then if $\mathbf{r}_1 : A$ and $\mathbf{r}_2 : B$, hence $\pi_A(\mathbf{r}) \hookrightarrow_{\pi_n} \mathbf{r}_1$. In other case—say $\mathbf{r}_1 : A \wedge B_1$ and $\mathbf{r}_2 : B_2$, with $B \equiv B_1 \wedge B_2$— then, by the induction hypothesis, $\pi_A(\mathbf{r}_1)$ reduces using at least one π_n reduction, and so $\pi_A(\mathbf{r}_1 + \mathbf{r}_2)$ does the same.

- If $\mathbf{r} = \pi_C(\mathbf{s})$, then $C \equiv A \wedge B$ and $\mathbf{s} : A \wedge B \wedge D$, so by the induction hypothesis, $\pi_C(\mathbf{s})$ reduces using at least one π_n reduction, hence $\pi_A(\pi_C(\mathbf{s}))$ does the same. ☐

Theorem 3.18 (Characterisation of typed closed normal forms)**.** *If $\boldsymbol{r} : A$ and $FV(\boldsymbol{r}) = \mathrm{Red}(\boldsymbol{r}) = \emptyset$, then $\boldsymbol{r} \rightleftarrows^* \sum_{i=1}^{n} \lambda x^{A_i}.\boldsymbol{s}_i$.*

Proof. We proceed by structural induction on $\mathbf{r}$.

- If $\mathbf{r} = \lambda x^A.\mathbf{s}$, then we are done.

- If $\mathbf{r} = \mathbf{r}_1\mathbf{r}_2$, then $\mathbf{r}_1 : B \Rightarrow A$, $\mathbf{r}_2 : B$ and $FV(\mathbf{r}_1) = \text{Red}(\mathbf{r}_1) = \emptyset$. So, by the induction hypothesis $\mathbf{r}_1 \rightleftarrows^* \sum_{i=1}^n \lambda x^{A_i}.\mathbf{s}_i$, hence $\mathbf{r}_1\mathbf{r}_2 \rightleftarrows^* (\sum_{i=1}^n \lambda x^{A_i}.\mathbf{s}_i)\mathbf{r}_2 \rightleftarrows^* \sum_{i=1}^n (\lambda x^{A_i}.\mathbf{s}_i)\mathbf{r}_2 \hookrightarrow_\beta^* \sum_{i=1}^n \mathbf{s}_i[\mathbf{r}_1/x]$, and therefore $\text{Red}(\mathbf{r}) \neq \emptyset$.

- If $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2$, then for $j = 1, 2$, $\mathbf{r}_j : A'_j$ and $FV(\mathbf{r}_j) = \text{Red}(\mathbf{r}_j) = \emptyset$, so by the induction hypothesis $\mathbf{r}_1 \rightleftarrows^* \sum_{i=1}^n \lambda x^{A_i}.\mathbf{s}_i$ and $\mathbf{r}_2 \rightleftarrows^* \sum_{i=n+1}^m \lambda x^{A_i}.\mathbf{s}_i$, so $\mathbf{r} \rightleftarrows^* \sum_{i=1}^m \lambda x^{A_i}.\mathbf{s}_i$.

- If $\mathbf{r} = \pi_A(\mathbf{s})$, then $\mathbf{s} : A \wedge B$ (notice that $\mathbf{s}$ cannot have type A because $\text{Red}(\pi_A(\mathbf{s})) = \emptyset$). So, by Lemma 3.17, $\text{Red}(\pi_A(\mathbf{s})) \neq \emptyset$. □

4. Computing with our Calculus

4.1. Pairs (and lists)

Because the symbol $+$ is associative and commutative, our calculus does not contain the usual notion of pairs. However it is possible to encode a deterministic projection, even if we have more than one term of the same type. An example, although there are various possibilities, is given in the following table:

Standard	Encoding
$\langle \mathbf{r}, \mathbf{s} \rangle : A \wedge A$	$\lambda x^{\mathbb{1}}.\mathbf{r} + \lambda x^{\mathbb{2}}.\mathbf{s} : \mathbb{1} \Rightarrow A \wedge \mathbb{2} \Rightarrow A$
$\pi_1 \langle \mathbf{r}, \mathbf{s} \rangle$	$\pi_{\mathbb{1} \Rightarrow A}(\lambda x^{\mathbb{1}}.\mathbf{r} + \lambda x^{\mathbb{2}}.\mathbf{s}) y^{\mathbb{1}}$

where types $\mathbb{1}$ and $\mathbb{2}$ are any two different types. This example uses free variables, but it is easy to close it, e.g. use $\lambda y.y$ instead of $y^{\mathbb{1}}$ in the second line.

Moreover, this technique is not limited to pairs. Due to the associativity nature of $+$, the encoding can be easily extended to lists.

4.2. A deterministic subsystem

In the previous section we have seen how to encode a pair, transforming the non-deterministic projection into a deterministic one via an encoding. Another possibility, is to remove the non-deterministic behaviour of this calculus by dropping the isomorphisms (1) and (2), as well as rules COMM and ASSO. Despite that such a modification would simplify the calculus—indeed, the projection can be taken as the standard projection—the resulting calculus would still count with distribution of application over conjunction and currification, two interesting features for a language. The former allows to execute a function only partially, when not all its results are needed. The latter can also be used to optimise programs when there are multiple calls to the same function, but one of its arguments is fixed.

4.3. Booleans

The Example 2.5 on booleans, actually overlooks an interesting fact: If $A \equiv B$, then both $\mathbf{T}$ and $\mathbf{F}$ behaves as a non-deterministic projector. Indeed, $\mathbf{Trs} \hookrightarrow^* \mathbf{r}$, but also $(\lambda x^A.\lambda y^B.x)\mathbf{rs} \rightleftarrows (\lambda x^A.\lambda y^A.x)\mathbf{rs} \rightleftarrows (\lambda x^A.\lambda y^A.x)(\mathbf{r}+\mathbf{s}) \rightleftarrows (\lambda x^A.\lambda y^A.x)(\mathbf{s}+\mathbf{r}) \rightleftarrows (\lambda x^A.\lambda y^A.x)\mathbf{sr} \hookrightarrow^* \mathbf{s}$.

Similarly, $\mathbf{Frs} \hookrightarrow^* \mathbf{s}$ and also $\mathbf{Frs} \rightsquigarrow^* \mathbf{r}$. Hence, $A \Rightarrow A \Rightarrow A$ is not suitable to encode the type Bool. The type $A \Rightarrow A \Rightarrow A$ has only one term in the underlying equational theory.

Fortunately, there are ways to construct types with more than one term. Let $[\mathbf{t}]^A = \lambda z^A.\mathbf{t}$ and $\{\mathbf{t}\}^A = \mathbf{t}\lambda z^A.z$. The type $((A \Rightarrow A) \Rightarrow B) \Rightarrow B \Rightarrow B$ has the following two different terms: $\mathbf{tt} := \lambda x^B.\lambda y^{(A\Rightarrow A)\Rightarrow B}.x$ and $\mathbf{ff} := \lambda x^{(A\Rightarrow A)\Rightarrow B}.\lambda y^B.\{x\}^A$.Hence, it is possible to encode an if-then-else conditional expression in the following way: $\mathsf{If}\ \mathbf{c}\ \mathsf{then}\ \mathbf{r}\ \mathsf{else}\ \mathbf{s} := \mathbf{cr}[\mathbf{s}]^{\mathsf{A}\Rightarrow\mathsf{A}}$. So, $\mathbf{ttr}[\mathbf{s}]^{A\Rightarrow A} \hookrightarrow^* \mathbf{r}$, while $\mathbf{ffr}[\mathbf{s}]^{A\Rightarrow A} \rightleftarrows^* \mathbf{ff}[\mathbf{s}]^{A\Rightarrow A}\mathbf{r} \hookrightarrow^* \{[\mathbf{s}]^{A\Rightarrow A}\}^A \hookrightarrow \mathbf{s}$.

5. Conclusions, Discussions and Future Work

In this paper we defined a proof system for propositional logic with an associative and commutative conjunction, and a distributive implication with respect to it, where equivalent propositions get the same proofs.

5.1. Related Work

5.1.1. Relation with other non-deterministic calculi

As a consequence of the commutativity of conjunction, the projection in our calculus is not position-oriented but type-oriented, which entails a non-deterministic projection where if a proposition has two possible proofs, the projection of its conjunction can output any of them. For example, if $\mathbf{r}$ and $\mathbf{s}$ are two possible proofs of A, then $\pi_A(\mathbf{r}+\mathbf{s})$ will output either $\mathbf{r}$ or $\mathbf{s}$.

In several works (cf. [21, §3.4] for a survey), the non-determinism is modelled by two operators: The first is normally written $+$, and instead of distributing over application, it actually makes the non-deterministic choice. Hence $(\mathbf{r}+\mathbf{s})\mathbf{t}$ reduces either to $\mathbf{rt}$ or to $\mathbf{st}$ [9]. The second one, denoted by $\|$, does not make the choice, and therefore $(\mathbf{r} \parallel \mathbf{s})\mathbf{t}$ reduces to $\mathbf{rt} \parallel \mathbf{st}$ [10]. One way to interpret these operators is that the first one is a non-deterministic one, while the second is the parallel composition. Another common interpretation is that $+$ is a *may-convergent* non-deterministic operator, where type systems ensure that at least one branch converges (i.e. terminates), while $\|$ is a *must-convergent* non-deterministic operator, where both branches are meant to converge [7, 15]. In our setting, the $+$ operator behaves like $\|$, and an extra operator (π_A) induces the non-deterministic choice. The main point is that this construction arose naturally as a consequence of considering the isomorphisms between types as an equivalence relation. Our type system ensures the termination of all the branches (Theorem 3.16), therefore ensuring must-convergence.

5.1.2. Relation with the selective λ-calculus

In a work by Garrigue and Aït-Kaci [19], only the isomorphism

$$A \Rightarrow (B \Rightarrow C) \equiv B \Rightarrow (A \Rightarrow C). \tag{5}$$

has been treated, which is complete with respect to the function type. Our contribution with respect to this work is that we also consider the conjunction, and hence four isomorphisms. Notice that isomorphism (5), in our setting, is a consequence of currification and commutation, that is $A \wedge B \equiv B \wedge A$ and $(A \wedge B) \Rightarrow C \equiv A \Rightarrow B \Rightarrow C$.

Their proposal is the selective λ-calculus, a calculus including labellings to identify which argument is being used at each time. Moreover, by considering the Church encoding of pairs, isomorphism (5) implies isomorphism (1) (commutativity of $\wedge$). However their proposal is different to ours. In particular, we track the term by its type, which is a kind of labelling, but when two terms have the same type, then we leave the system to non-deterministically choose any proof. One of our main novelties is, indeed, the non-deterministic projector. However, we can also get back determinism, by encoding a labelling, or by dropping some of the isomorphisms, as discussed in the next section.

5.2. Future Work

5.2.1. Adding more connectives

A subtle question is how to add a neutral element of the conjunction, which will imply more isomorphisms, e.g. $A \wedge \top \equiv A$, $A \Rightarrow \top \equiv \top$ and $\top \Rightarrow A \equiv A$. Notice that within our system, $\top \Rightarrow \top \equiv \top$ would make it possible to derive $(\lambda x^\top.xx)(\lambda x^\top.xx) : \top$, however this term is not the classical Ω, it is typed by $\top$, and imposing some restrictions on the beta reduction, it could be forced not to reduce to itself but to discard its argument. For example: "If $A \equiv \top$, then $(\lambda x^A.\mathbf{r})\mathbf{s} \hookrightarrow \lambda x^A.\mathbf{r}$, in other case, do the standard beta-reduction".

5.2.2. Probabilistic and quantum computing

A second line is the probabilistic interpretation of the non-determinism in our calculus. In [14] a probability space over the set of non-deterministic execution traces is defined. This way, our calculus is transformed into a probabilistic calculus instead of just a non-deterministic one, providing an alternative way for more complex constructions. Moreover, the original motivation behind the linear algebraic extension of lambda calculus [3] and its *vectorial* type system [2] was to encode quantum computing on it by considering not only non-deterministic superpositions, but formal linear combinations of terms. A projection depending on scalars could lead to a measurement operator in a future design. This is a promising future direction we are willing to take.

Acknowledgement. We thank Emmanuel Beffara, Frédéric Blanqui, Emmanuel Polonowsky and Thomas Seiller for enlightening discussions.

[1] Arrighi, P., Díaz-Caro, A., 2012. A System F accounting for scalars. Logical Methods in Computer Science 8 (1:11).

[2] Arrighi, P., Díaz-Caro, A., Valiron, B., 2012. A type system for the vectorial aspects of the linear-algebraic lambda-calculus. In: Kashefi, E., Krivine, J., van Raamsdonk, F. (Eds.), Proceedings of DCM 2011. Vol. 88 of EPTCS. pp. 1–15.

[3] Arrighi, P., Dowek, G., 2008. Linear-algebraic λ-calculus: higher-order, encodings, and confluence. In: Voronkov, A. (Ed.), Proceedings of RTA 2008. Vol. 5117 of LNCS. pp. 17–31.

[4] Barendregt, H., 1984. The Lambda Calculus: Its Syntax and Semantics. North-Holland.

[5] Boudol, G., 1994. Lambda-calculi for (strict) parallel functions. Information and Computation 108 (1), 51–127.

[6] Bruce, K. B., Di Cosmo, R., Longo, G., 1992. Provable isomorphisms of types. Mathematical Structures in Computer Science 2 (2), 231–247.

[7] Bucciarelli, A., Ehrhard, T., Manzonetto, G., 2012. A relational semantics for parallelism and non-determinism in a functional setting. Annals of Pure and Applied Logic 163 (7), 918–934.

[8] Coquand, T., Huet, G., 1988. The calculus of constructions. Information and Computation 76 (2–3), 95–120.

[9] de'Liguoro, U., Piperno, A., 1995. Non deterministic extensions of untyped λ-calculus. Information and Computation 122 (2), 149–177.

[10] Dezani-Ciancaglini, M., de'Liguoro, U., Piperno, A., 1998. A filter model for concurrent λ-calculus. SIAM Journal on Computing 27 (5), 1376–1419.

[11] Di Cosmo, R., 1995. Isomorphisms of types: from λ-calculus to information retrieval and language design. Progress in Theoretical Computer Science. Birkhauser.

[12] Di Cosmo, R., 2005. A short survey of isomorphisms of types. Mathematical Structures in Computer Science 15 (5), 825–838.

[13] Díaz-Caro, A., Dowek, G., 2013. Non determinism through type isomorphism. In: Kesner, D., Viana, P. (Eds.), Proceedings of LSFA 2012. Vol. 113 of EPTCS. pp. 137–144.

[14] Díaz-Caro, A., Dowek, G., 2014. The probability of non-confluent systems. In: Ayala-Rincn, M., Bonelli, E., Mackie, I. (Eds.), Proceedings of DCM 2013. Vol. 144 of EPTCS. pp. 1–15.

[15] Díaz-Caro, A., Manzonetto, G., Pagani, M., 2013. Call-by-value non-determinism in a linear logic type discipline. In: Artemov, S., Nerode, A. (Eds.), Proceedings of LFCS 2013. Vol. 7734 of LNCS. pp. 164–178.

[16] Dowek, G., Hardin, T., Kirchner, C., 2003. Theorem proving modulo. Journal of Automated Reasoning 31 (1), 33–72.

[17] Dowek, G., Jiang, Y., 2011. On the expressive power of schemes. Information and Computation 209, 1231–1245.

[18] Dowek, G., Werner, B., 2003. Proof normalization modulo. The Journal of Symbolic Logic 68 (4), 1289–1316.

[19] Garrigue, J., Aït-Kaci, H., 1994. The typed polymorphic label-selective λ-calculus. In: Proceedings of POPL 1994. ACM SIGPLAN. pp. 35–47.

[20] Geuvers, H., Krebbers, R., McKinna, J., Wiedijk, F., 2010. Pure type systems without explicit contexts. In: Crary, K., Miculan, M. (Eds.), Proceedings of LFMTP 2010. Vol. 34 of EPTCS. pp. 53–67.

[21] Manzonetto, G., 2008. Models and theories of lambda calculus. Ph.D. thesis, Università Ca'Foscari (Venice) and Université Paris Diderot (Paris 7).

[22] Martin-Löf, P., 1984. Intuitionistic type theory. Studies in proof theory. Bibliopolis.

[23] Nipkow, T., 1990. A critical pair lemma for higher-order rewrite system and its application to λ^*. In: Proceedings of the First Workshop on Logical Frameworks. pp. 361–376.

[24] Pagani, M., Ronchi Della Rocca, S., 2010. Linearity, non-determinism and solvability. Fundamental Informaticae 103 (1–4), 173–202.

[25] Park, J., Seo, J., Park, S., Lee, G., 2014. Mechanizing metatheory without typing contexts. Journal of Automated Reasoning 52 (2), 215–239.

[26] Rittri, M., 1990. Retrieving library identifiers via equational matching of types. In: Proceedings of CADE 1990. Vol. 449 of LNCS. pp. 603–617.

[27] The Univalent Foundations Program, 2013. Homotopy Type Theory: Univalent Foundations of Mathematics. http://homotopytypetheory.org/book, Institute for Advanced Study.