The Minimal Relevant Logic and the Call-by-Value Lambda Calculus*

S. van Bakel, M. Dezani-Ciancaglini, U. de’Liguoro, Y. Motohama
Department of Computing, Imperial College London, UK
Dipartimento di Informatica, Università degli Studi di Torino, Italy

Abstract

The minimal relevant logic $B^+$, seen as a type discipline, includes an extension of Curry types known as the intersection type discipline. We will show that the full logic $B^+$ gives a type assignment system which produces a model of Plotkin’s call-by-value $\lambda$-calculus.

1 Introduction

The logical system $B^+$ arose from Meyer and Routley’s investigation on the negation-free entailment logic [12]. In their approach, $B^+$ turns out as the minimal relevant logic, which is complete with respect to a variant of Kripke models, called the positive model structures.

Independently, and with quite different aims, an extension of the Curry type assignment system with “intersection” types was introduced in [3] by Coppo and Dezani. The same authors, together with Barendregt, were able to prove in [2] that, provided a suitable axiomatisation of the subtype relation $A \leq B$, the set of filters over types is a $\lambda$-model. As a consequence, a completeness theorem for the intersection types assignment system was established.

As remarked by Meyer in [8], it can be recognised that $A \leq B$ holds in the type theory of [2] if and only if, “on translation”, $A \rightarrow B$ is a theorem of $B^+$. However, the language of $B^+$ has one more connective than intersection types, namely disjunction. Extensions of the intersection type discipline with a “union” type constructor have been pursued, by some of the present authors together with others, both in the framework of classical $\lambda$-calculus and in that of some “parallel” and non-deterministic extensions of $\lambda$-calculus: see [1, 5, 4]. From this investigations it turns out that strong systems of union types do not give filter $\lambda$-models, while weaker systems allow for a satisfactory logical analysis of some extended $\lambda$-calculi.

Here we will follow a different path. Instead of extending the calculus, we will consider Plotkin’s call-by-value $\lambda$-calculus [9], whose syntax is the same as that of the $\lambda K$-calculus, but the $\beta$-rule is replaced by a restricted form:

\[ \beta_v \quad (\lambda x.M)N \rightarrow M[N/x], \quad \text{if } N \text{ is not an application.} \]

The idea is that a term that is an application needs to be further evaluated before it can be passed on as an argument. By rule $(\beta_v)$, in contrast to the classical $\beta$-rule, substitution is delayed until the evaluation of the argument $N$ reaches a value, namely either a variable or an abstraction. Because of this, the call-by-value $\lambda$-calculus has been proposed and studied as the abstract counterpart of call-by-value lazy programming languages, reflecting closer

---

* Partially supported by NATO Collaborative Research Grant CRG 970285 ‘Extended Rewriting and Types’.
the actual practice of language implementation. In fact, the subtleties of parameter passing
can be caught by the formal calculus, and analyzed using both proof theoretical and model
theoretical tools in an elegant way [6, 10, 11].

What we will show here is that \( \mathcal{B}^+ \) is the type theory of the call-by-value \( \lambda \)-calculus, in the
following sense: we will consider the full language of the logic \( \mathcal{B}^+ \), including disjunction,
and take the \( \mathcal{B}^+ \) axioms and rules as defining a subtype relation. Then we will consider a
type assignment system which is essentially that of [2], but using the extended type syntax
and subtyping relation hinted above, with a key restriction on the types that can be assumed
for variables in a basis. It turns out that, within this system, types are preserved under
\( \beta\nu \)-conversion. Moreover, the filter structure of the type theory is an instance of call-by-value
syntactical \( \lambda \)-model (a generalisation, [6, 10, 11] of the notion of syntactical \( \lambda \)-model due to
Hindley and Longo [7]).

2 The minimal relevant logic \( \mathcal{B}^+ \) as a type discipline

The minimal relevant logic \( \mathcal{B}^+ \) is a propositional calculus. To see \( \mathcal{B}^+ \) as a type discipline,
we interpret propositions as types: the constant for truth is interpreted as a constant type \( \nu \),
whose intended meaning is “the type of values”; implication as the arrow type connective \( \to \),
conjunction and disjunction as intersection (\( \land \)) and union type connectives (\( \lor \)), respectively.
Then we define a preorder relation \( \leq \) such that \( A \leq B \) if and only if \( A \to B \) is a theorem of
\( \mathcal{B}^+ \).

Definition 2.1 (The set \( L \) of formulae) The set \( L \) of formulae is defined by

i) a denumerable set \( \mathcal{P} \nu \) of propositional variables, ranged over by \( a,b,c \) possibly with
subscripts;

ii) \( \nu \in L \) (a type constant);

iii) \( A,B \in L \Rightarrow (A \to B),(A \land B),(A \lor B) \in L \).

Notation. To avoid using too many parentheses, we assume that \( \land \) and \( \lor \) take precedence
over \( \to \), and that \( \to \) associates to the right.

Definition 2.2 (The minimal relevant logic \( \mathcal{B}^+ \) with \( \leq \)) \( \mathcal{B}^+ \) is the logic on the language \( L \)
defined by the following axioms and rules:

\[
\begin{align*}
(A1) & : A \leq \nu \\
(A2) & : A \leq A \\
(A3) & : A \leq A \land A \\
(A4) & : A \land B \leq A, A \land B \leq B \\
(A5) & : A \lor A \leq A \\
(A6) & : A \leq A \lor B, B \leq A \lor B \\
(A7) & : (A \to B) \land (A \to C) \leq A \to B \land C \\
(A8) & : (A \to C) \land (B \to C) \leq A \lor B \to C \\
(A9) & : A \land (B \lor C) \leq (A \land B) \lor (A \land C) \\
(R1) & : A \leq B, B \leq C \Rightarrow A \leq C \text{ (transitivity)} \\
(R2) & : A \leq B, C \leq D \Rightarrow A \land C \leq B \land D \text{ (\( \land \)-monotonicity)} \\
(R3) & : A \leq B, C \leq D \Rightarrow A \lor C \leq B \lor D \text{ (\( \lor \)-monotonicity)} \\
(R4) & : A \leq B \Rightarrow B \leq C \leq A \to C \text{ (suffixing)} \\
(R5) & : B \leq C \Rightarrow A \to B \leq A \to C \text{ (prefixing)}.
\end{align*}
\]

We define \( \sim \) as the symmetric closure of \( \leq \). The relation \( \sim \) enjoys the following properties:
i) \( \sim \) is a congruence relation.

ii) \( A \land B \sim B \land A, \ A \lor B \sim B \lor A. \)

iii) \( (A \land B) \land C \sim A \land (B \land C), \ (A \lor B) \lor C \sim A \lor (B \lor C). \)

iv) \( A \land (B \lor C) \sim (A \land B) \lor (A \land C), \ A \lor (B \land C) \sim (A \lor B) \land (A \lor C) \) (distributivity).

v) \( (A \rightarrow B) \land (A \rightarrow C) \sim (A \rightarrow B \land C). \)

vi) \( (A \rightarrow C) \land (B \rightarrow C) \sim (A \lor B \rightarrow C). \)

To further investigate properties of the system \( B^+ \), it is useful to associate to each type both its conjunctive and disjunctive normal form. First let us define inductively the following subsets of \( L \).

**Definition 2.3 (Stratification of \( L \))**  \( T_\rightarrow, T_\lor, T_\land, T_\land\lor, T_\lor\land \subseteq L \) are inductively defined:

\[(T_\rightarrow): \ v \in T_\rightarrow; \ a \in T_\rightarrow, \ for \ all \ propositional \ variables \ a \]

\( A \in T_\land, B \in T_\lor \Rightarrow A \rightarrow B \in T_\rightarrow \)

\( (T_\lor): \ A \in T_\rightarrow \Rightarrow A \in T_\lor \)

\( A, B \in T_\lor \Rightarrow A \lor B \in T_\lor \)

\( (T_\land): \ A \in T_\rightarrow \Rightarrow A \in T_\land \)

\( A, B \in T_\land \Rightarrow A \land B \in T_\land \)

\( (T_\land\lor): \ A \in T_\lor \Rightarrow A \in T_\land\lor \)

\( A, B \in T_\land\lor \Rightarrow A \land B \in T_\land\lor \)

\( (T_\lor\land): \ A \in T_\land \Rightarrow A \in T_\lor\land \)

\( A, B \in T_\lor\land \Rightarrow A \lor B \in T_\lor\land. \)

We will now introduce maps from arbitrary types belonging to \( L \) into their conjunctive/disjunctive normal forms in \( T_\land\lor \) and \( T_\lor\land \), respectively.

**Definition 2.4** The maps \( m_{\land\lor}: L \rightarrow T_\land\lor \) and \( m_{\lor\land}: L \rightarrow T_\lor\land \) are as follows defined by simultaneously induction on the structure of formulæ:

i) \( m_{\land\lor}(A) = m_{\lor\land}(A) = A \) if \( A \in \mathcal{P}V \cup \{v\} \).

ii) If \( m_{\land\lor}(A) = \bigvee_{i \in I} A_i \) (where each \( A_i \in T_\land \)) and \( m_{\lor\land}(B) = \bigwedge_{j \in J} B_j \) (where each \( B_j \in T_\lor \)) then

\[
m_{\land\lor}(A \rightarrow B) = m_{\lor\land}(A \rightarrow B) = \bigwedge_{i \in I} \bigvee_{j \in J} (A_i \rightarrow B_j).
\]

iii) \( m_{\land\lor}(A \land B) = m_{\land\lor}(A) \land m_{\land\lor}(B) \), and, if \( m_{\land\lor}(A) = \bigvee_{i \in I} A_i \) and \( m_{\lor\land}(B) = \bigwedge_{j \in J} B_j \) then

\[
m_{\land\lor}(A \land B) = \bigvee_{i \in I} \bigwedge_{j \in J} (A_i \land B_j).
\]

iv) \( m_{\land\lor}(A \lor B) = m_{\land\lor}(A) \lor m_{\land\lor}(B) \), and, if \( m_{\land\lor}(A) = \bigwedge_{i \in I} A_i \) and \( m_{\lor\land}(B) = \bigvee_{j \in J} B_j \) then

\[
m_{\land\lor}(A \lor B) = \bigwedge_{i \in I} \bigvee_{j \in J} (A_i \lor B_j).
\]

We shall prove, in the Appendix, that \( A \sim m_{\land\lor}(A) \sim m_{\lor\land}(A) \) for all \( A \).

The Lindenbaum algebra of \( B^+ \), i.e. the quotient of \( L \) under \( \sim \), is ordered by the relation \([A]_\sim \leq [B]_\sim \iff [A]_\sim \leq [B]_\sim \) (where \([A]_\sim \) is the equivalence class of \( A \)). As an ordered set it is a distributive lattice, where \([A \land B]_\sim \) and \([A \lor B]_\sim \) are respectively the meet and the join of \([A]_\sim \) and \([B]_\sim \), and \([v]_\sim \) is the top.

Co-prime elements in a lattice, here called \( \lor \)-prime elements, play a role in the representation theorems of distributive lattices. In our setting, \( \lor \)-prime formulæ will turn out to be types of closed terms, \( \beta_\lor \)-convertible with values. We introduce them here and defer the analysis of their connection to values to the next sections.
Definition 2.5 A formula A is $\lor$-prime iff $A \leq B \lor C \Rightarrow A \leq B$ or $A \leq C$.

Theorem 2.6 (Properties of $\lor$-prime formulas)

i) $\lor$ and any propositional variable are $\lor$-prime.

ii) If $m_{\lor}(A) = \lor_{i \in I} A_i$ then $A_i$ is $\lor$-prime, for all $i \in I$.

iii) The formula $\land_{i \in I} (A_i \rightarrow B_i)$ is $\lor$-prime for all (finite) $I$ and formulas $A_i, B_i$.

iv) $B \rightarrow C \not\sim \lor$ for all $B, C$.

v) $\land_{i \in J} (A_i \rightarrow B_i) \leq C \rightarrow D$ and $C$ is $\lor$-prime imply $C \leq \land_{i \in J} A_i$ and $\land_{i \in J} B_i \leq D$ for some $J \subseteq I$.

The proof of Theorem 2.6 is reported in the Appendix. The condition ‘$C$ is $\lor$-prime’ in statement (v) of the above theorem, is necessary. A counter-example is axiom (A8).

3 The type assignment system

Type assignment systems are formal systems deriving statements of the form $M : A$, where $M$ (the subject) is a pure $\lambda$-term and $A$ (the predicate) is a type (here a formula). The intuitive meaning is that types are properties of terms, hence $M : A$ means “$M$ has property $A$”. As $M$ may contain free variables, assumptions about the types of such variables may be used in the derivation, and are collected into bases (also called contexts): $x_1 : A_1, \ldots, x_n : A_n$, which we consider as finite sets of statements (assumptions) such that $x_i \not\equiv x_j$ if $i \neq j$.

Bases are ranged over by $\Gamma$, and we write $\Gamma, x : A$ as abbreviation of $\Gamma \cup \{x : A\}$, under the assumption that $x$ does not occur in $\Gamma$. Then the general form of statements derivable in a type assignment system is $\Gamma \vdash M : A$, meaning “$M$ has type $A$ if the variables in $\Gamma$ have the types listed in $\Gamma$ itself”.

We observe that, although the basic motivation for having bases is to handle the types of the free variables in the subject $M$, it is not the case that, if $\Gamma \vdash M : A$ is derivable, then the subjects of statements in $\Gamma$ are exactly the free variables of $M$. Indeed, on the one hand $\Gamma$ may contain assumptions about variables not occurring free in $M$; on the other hand, because of rule (v), $M$ may contain free variables which are not in $\Gamma$.

Definition 3.1 (The type assignment system)

i) A statement is an expression of the form $M : A$, where $M$ is a term (subject) and $A$ is a type (predicate).

ii) An assumption is a statement whose subject is a term variable.

iii) A basis is a set of assumptions with distinct variables as subjects whose predicates are $\lor$-prime types.

iv) A statement $M : A$ is derivable from a basis $\Gamma$, notation $\Gamma \vdash M : A$, if $\Gamma \vdash M : A$ can be proved using the following axioms and inference rules:
\[(Ax): \frac{\Gamma, x: A \vdash x: A}{\Gamma \vdash M : \nu} \text{ (if } M \text{ is a variable or an abstraction)}\]

\[(\rightarrow I): \frac{\Gamma, x: A \vdash B}{\Gamma \vdash \lambda x. M : A \rightarrow B}\]

\[(\rightarrow E): \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}\]

\[(\land I): \frac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \land B}\]

\[(\leq): \frac{\Gamma \vdash M : A}{\Gamma \vdash M : B} (A \leq B)\]

This system differs from that in [2] not just because of the syntax of types and of the definition of the \(\leq\) relation, but also because of the restrictions (3.1(iii) on assumptions and of rule (\(v\)).

If we allow assumptions with predicates which are not \(\lor\)-prime, then typing is not preserved under \(\beta_\nu\)-reduction. In fact, suppose that \(A, B\) and \(C\) are unrelated formulae (distinct propositional variables, say). Then we can deduce both \(x : A \rightarrow C, y : A \land B \vdash xy : C\) and \(x : B \rightarrow C, y : A \land B \vdash xy : C\), hence

\[y : A \land B \vdash \lambda x. xy : ((A \rightarrow C) \rightarrow C) \land ((B \rightarrow C) \rightarrow C) \sim [A \rightarrow C] \lor (B \rightarrow C) \rightarrow C,\]

by rules \((\rightarrow I), (\land I)\) and \((\leq)\). Suppose that we relax the restriction on the assumptions, and consider \(z : (A \rightarrow C) \lor (B \rightarrow C)\). Then by rule \((\rightarrow E)\) and the admissibility of weakening we have:

\[z : (A \rightarrow C) \lor (B \rightarrow C), y : A \land B \vdash (\lambda x. xy)z : C.\]

Now \((\lambda x. xy)z\) \(\beta_\nu\)-reduces to \(zy\), but \(z : (A \rightarrow C) \lor (B \rightarrow C), y : A \land B \not\vdash zy : C\). This is an easy consequence of the Generation Lemma below (Lemma 3.4).

A second remark concerns rule \((v)\). Compared with [2], this rule is close to rule \((\omega)\):

\[(\omega): \frac{\Gamma \vdash M : \omega}{\Gamma \vdash M : \omega}\]

of [2], but for the restriction on the form of \(M\). In fact, in the present system it is not true that any term has a type: a typical un-typeable term is \(\Omega \equiv (\lambda x. xx)(\lambda x. xx)\). As a matter of fact, in our system only terms which are \(\beta_\nu\)-convertible to values have a type in the empty basis, hence (also) type \(\nu\). This is a basic choice: as terms will be discriminated according to the types we can assign to them, the fact that “divergent” closed terms (which are not \(\beta_\nu\)-convertible to a value) have no type will identify all of them.

In contrast, in [2] any term has a type, at least the “universal type” \(\omega\) (trivially) and also type \(\omega \rightarrow \omega\), because of the axiom \(\omega \leq \omega \rightarrow \omega\). Had we introduced here the universal type \(\omega\), serious problems would immediately have arisen. Indeed, one would have, e.g., that \(\vdash (\lambda xy. y)\Omega : A \rightarrow A\). But, once the inequation \(\omega \leq \omega \rightarrow \omega\) has been dropped, we expect, as it happens in all standard models of call-by-value \(\lambda\)-calculus, that arrow types (in the empty basis) characterize terms evaluating to a value (see, e.g. [5]). But then \((\lambda xy. y)\Omega\) should be itself \(\beta_\nu\)-convertible with a value, which is not.

The next lemmas are routine, but essential for the technical development: the essence is that variables can be substituted by terms having the same type, and that, under certain conditions, the rules of the system can be reversed.
Lemma 3.2 (Syntactical properties)

i) (weakening) If \( \Gamma \vdash M : A \) then \( \Gamma, x : B \vdash M : A \).

ii) (thinning) If \( \Gamma \vdash M : A \) then \( \Gamma[M/x \vdash M : A] \), where \( \Gamma[M = \{ x : A \in \Gamma | x \in FV(M) \} \).

iii) (substitution) If \( \Gamma, x : A \vdash M : B \) and \( \Gamma \vdash N : A \) then \( \Gamma \vdash M[N/x] : B \).

Proof: All statements above are proved by induction on derivations. In particular (iii) is proved by induction on the derivation of \( \Gamma, x : A \vdash M : B \). We only observe that, if the last rule is \((v)\), then two cases are possible, according to the form of \( M \):

(iii): Case \( M \equiv y \): if \( y \equiv x \) then \( y[N/x] \equiv N \) and we have, using the hypothesis \( \Gamma \vdash N : A \) and rule \((\leq)\), that \( \Gamma \vdash N : v \); otherwise, if \( y \not\equiv x \) then \( y[N/x] \equiv y \), and \( \Gamma \vdash y : v \) by rule \((v)\).

(ii): Case \( M \equiv \lambda y. M' \): then \( (\lambda y. M')[N/x] \equiv \lambda z. M'[y/z][N/x] \) (where \( z \) is a fresh variable) which is an abstraction; therefore \( \Gamma \vdash M[N/x] : v \) is an instance of rule \((v)\).

When restricting to \( \lor \)-prime formulae, some useful properties concerning the type assign-
ment system clearly reflect properties of natural deduction systems for propositional logic.

Lemma 3.3

i) The following rule is admissible:

\[
(\leq)_L : \frac{\Gamma, x : A \vdash M : B}{\Gamma, x : C \vdash M : B} \quad (C \leq A \text{ and } C \text{ is } \lor\text{-prime})
\]

ii) If \( M : B \) is derived from \( M : A_1 \cdots M : A_n \) only by using rules \((\land I)\) and \((\leq)\), then \( A_1 \land \cdots \land A_n \leq B \).

Proof: (i) By induction on the derivation of \( \Gamma, x : A \vdash M : B \).

(ii) By induction on derivations.

The Generation Lemma allows for a reverse reading of the rules. In particular, it establishes that, if the subject of the conclusion is an application or an abstraction, then, under suitable hypotheses, we have information on types which can be given to its immediate subterms.

Lemma 3.4 (Generation Lemma)  

i) If \( \Gamma \vdash x : A \) and \( A \not\sim v \) then, for some \( B, x : B \in \Gamma \) and \( B \leq A \).

ii) \( \Gamma \vdash MN : A \) iff \( \Gamma \vdash M : B \rightarrow A \) and \( \Gamma \vdash N : B \) for some \( B \).

iii) If \( \Gamma \vdash \lambda x. M : A \) and \( A \not\sim v \) then \( \Gamma, x : B_i \vdash M : C_i \) and \( \land_{i \in I} (B_i \rightarrow C_i) \leq A \) for some \( I, B_i, C_i \).

iv) \( \Gamma \vdash \lambda x. M : B \rightarrow C \) and \( B \) is \( \lor\)-prime iff \( \Gamma, x : B \vdash M : C \).

Proof: i) Immediate, by Lemma 3.3 (ii).

ii) \((\Rightarrow)\) By induction on the derivation of \( \Gamma \vdash MN : A \). The only non-trivial case is when the last rule applied is \((\land I)\), i.e. \( A = A_1 \land A_2 \). Then

\[
\frac{\Gamma \vdash M : A_1 \quad \Gamma \vdash M : A_2}{\Gamma \vdash M : A_1 \land A_2} \quad (\land I)
\]

is the last step. By induction, there are \( B_1, B_2 \) such that \( \Gamma \vdash M : B_i \rightarrow A_i, \Gamma \vdash N : B_i \) for
\( i = 1, 2. \) Then

\[
\begin{array}{c}
\Gamma \vdash M : B_1 \rightarrow A_1 \\
\Gamma \vdash M : B_1 \land B_2 \rightarrow A_1 \\
\hline
\Gamma \vdash M : (B_1 \land B_2 \rightarrow A_1) \land (B_1 \land B_2 \rightarrow A_2) \\
\hline
\Gamma \vdash M : B_1 \land B_2 \rightarrow A_1 \land A_2 \\
\end{array}
\]

(\( \leq \)) By rule (\( \rightarrow \)).

\( iii \) The derivation of \( \Gamma \vdash \lambda x. M : A \) (\( A \not\sim \nu \)) has the shape:

\[
\begin{array}{c}
\Gamma, x : B_1 \vdash M : C_1 \\
\Gamma, x : B_1 \vdash M : C_1 \quad \ldots \quad \Gamma, x : B_1 \vdash M : C_n \\
\hline
\Gamma, x : B_1 \vdash \lambda x. M : B_1 \rightarrow C_1 \\
\Gamma, x : B_n \vdash M : C_n \\
\hline
\Gamma \vdash \lambda x. M : A
\end{array}
\]

then by Lemma 3.3 (ii)

\( \land_{i \in I} (B_i \rightarrow C_i) \land \nu \quad \ldots \quad \land \nu \leq A. \)

\( I \not= \emptyset \) because \( A \not\sim \nu \) and \( B_i \rightarrow C_i \leq \nu \) by (A1). Thus

\( \land_{i \in I} (B_i \rightarrow C_i) \leq A. \)

\( iv \) In (ii) above, as \( B \rightarrow C \not\sim \nu \) by Theorem 2.6 (iii), we can assume \( A = B \rightarrow C. \) Thus there exist \( I, B_i, C_i, \)

\( \Gamma, x : B_i \vdash M : C_i, \quad \land_{i \in I} (B_i \rightarrow C_i) \leq B \rightarrow C. \)

Since \( B \) is \( \lor \)-prime, by Theorem 2.6 (iv),

\( \exists J \subseteq I, B \leq \land_{i \in I} B_i \) and \( \land_{i \in J} C_i \leq C \implies \exists J \subseteq I [\forall i \in J, B_i \leq B_i \) and \( \land_{i \in J} C_i \leq C]. \)

Hence we have:

\[
\begin{array}{c}
\Gamma, x : B_i \vdash M : C_i \text{ and } B \leq B_i (\forall i \in J) \quad \text{and } B \text{ is } \lor \text{-prime} \\
\hline
\Gamma, x : B \vdash M : \land_{i \in J} C_i \quad (\land I) \\
\Gamma, x : B \vdash M : C \quad (\leq)
\end{array}
\]

The paper [1] considers a disjunction-elimination rule, which, as such, is not admissible in the present system. However, as in [5], a suitable restriction of it is admissible:

\[
\begin{array}{c}
\Gamma \vdash M, V / x : C \\
\Gamma, x : A \vdash M : C \\
\Gamma, x : B \vdash M : C \\
\hline
\Gamma \vdash V : A \lor B
\end{array}
\]

(V is a variable or an abstraction)

In fact, if \( V \equiv y \) is a variable that does not occur in \( \Gamma \), then \( A \lor B \sim \nu \) (by Lemma 3.4 (i)); since \( \nu \) is \( \lor \)-prime, both \( A, B \sim \nu \), and the thesis follows from the Substitution Lemma (Lemma 3.2 (iii)). If \( y : D \) is in \( \Gamma \) then \( D \) is \( \lor \)-prime and \( D \leq A \lor B \); therefore, by rule (\( \leq \)), either \( \Gamma \vdash y : A \) or \( \Gamma \vdash y : B. \) In both cases, the thesis follows by substitution.

Suppose, instead, that \( V \equiv \lambda y. M \). For \( A \lor B \sim \nu \), we reason as in the previous case. Otherwise, if \( A \lor B \not\sim \nu \), then, by Lemma 3.4 (iii), \( \Gamma, y : D_i \vdash M : E_i \) for \( i \in I \) (for some finite \( I \) and types \( D_i, E_i \)), such that \( \land_{i \in I} (D_i \rightarrow E_i) \leq A \lor B. \) By (\( \rightarrow \)) and (\( \land \)), \( \Gamma \vdash \lambda y. M : \land_{i \in I} (D_i \rightarrow E_i); \) on the other hand, we know that \( \land_{i \in I} (D_i \rightarrow E_i) \) is \( \lor \)-prime by Lemma 2.6 (v), hence either \( \Gamma \vdash \lambda y. M : A \) or \( \Gamma \vdash \lambda y. M : B \), and the result follows by substitution.

The restriction on the form of \( V \) is essential: notice that \( x : C \rightarrow A \lor B, y : C \vdash xy : A \lor B, \) but we cannot type, in the same basis, \( xy \) either by \( A \) or by \( B. \)
4 The call-by-value $\lambda$-calculus and its models

The type assignment system we have introduced does not respect the (unrestricted) $\beta$-rule, as it is the case, instead, for the system of [2]. More precisely, types are preserved neither under $\beta$-reduction, nor under $\beta$-expansion.

For example, one can deduce $(\lambda y. x y y)(u v) : C$ from the basis $\Gamma = \{ x : (A \rightarrow A \rightarrow C) \land (B \rightarrow B \rightarrow C), u : D \rightarrow A \lor B, v : D \}$ as follows:

\[
\begin{align*}
\Gamma \vdash & x : (A \rightarrow A \rightarrow C) \\
\Gamma \vdash & x : A \rightarrow A \\
\Gamma \vdash & y : A \\
\Gamma \vdash & xy : C \\
\Gamma \vdash & \lambda y. xy : A \rightarrow C \\
\Gamma \vdash & \lambda y. xy : B \rightarrow C \\
\Gamma \vdash & \lambda y. xy : (A \rightarrow C) \land (B \rightarrow C) \\
\Gamma \vdash & u : D \rightarrow A \lor B \\
\Gamma \vdash & v : D \\
\Gamma \vdash & (\lambda y. xy)(uv) : C
\end{align*}
\]

One cannot deduce $C$ from $\Gamma$ for the normal form $x(u v)(u v)$. Also one can deduce $A \rightarrow A$ for $\lambda y. y$, but this type cannot be deduced for $(\lambda x y y)((\lambda z. z z)(\lambda z. z z))$, which has no type at all.

Thus this type assignment system does not induce a $\lambda$-model. Instead it gives a model of the call-by-value $\lambda$-calculus. The call-by-value $\lambda$-calculus, as introduced by Plotkin [9], is obtained by restricting the $\beta$-rule to redexes whose argument is a value (i.e. a variable or an abstraction).

Definition 4.1 (Call-by-value $\lambda$-calculus) The set of values $Val \subseteq \Lambda$ is defined by

\[ Val = Var \cup \{ \lambda x. M | M \in \Lambda \} \]

where $Var$ is the set of term variables. The call-by-value $\beta$-reduction rule is

\[
(\beta_v) \ (\lambda x. M) N \rightarrow_v M[N/x] \text{ if } N \in Val.
\]

The contextual, reflexive, symmetric, and transitive closure of $\rightarrow_v$ is denoted by $=_{v}$.

Models of call-by-value $\lambda$-calculus are a generalisation of $\lambda$-models, in which a distinguished subset $\mathcal{V}$ of an applicative structure $\mathcal{D}$ is meant to interpret values: if one takes $\mathcal{V} = D$, the following definition immediately coincides with the notion of syntactical $\lambda$-model of [7]. A slightly different definition can be found in [6]; the present one is from [10].

Definition 4.2 (Models of call-by-value $\lambda$-calculus) A model of call-by-value $\lambda$-calculus is a structure $\mathcal{M} = \langle \mathcal{D}, \mathcal{V}, \cdot, [\cdot]^{\mathcal{M}} \rangle$, such that $\cdot$ is a binary operation on $\mathcal{D}$, called application (i.e. $\cdot$ is an applicative structure), $\mathcal{V} \subseteq \mathcal{D}$ and, for any environment $\rho : \mathcal{V} \rightarrow \mathcal{V}$, $[\cdot]^{\mathcal{M}}$ is a map from $\Lambda$ to $\mathcal{D}$ satisfying (writing simply $[M]_{\rho}$ for $[M]^{\mathcal{M}}_{\rho}$):

i) $[x]_{\rho} = \rho(x)$

ii) $[MN]_{\rho} = [M]_{\rho} \cdot [N]_{\rho}$

iii) $[\lambda x. M]_{\rho} \cdot d = [M]_{\rho[x:=d]}$ if $d \in \mathcal{V}$

iv) $[\forall x \in FV(M), \rho(x) = \rho'(x) \Rightarrow [M]_{\rho} = [M]_{\rho'}$.

v) $[\lambda x. M]_{\rho} = [\lambda y. M[y/x]]_{\rho}$ if $y \notin FV(M)$

vi) $[\forall d \in \mathcal{V}, [M]_{\rho[x:=d]} = [N]_{\rho[x:=d]} \Rightarrow [\lambda x. M]_{\rho} = [\lambda x. N]_{\rho}$
vii) \( M \in \text{Val} \Rightarrow [[M]]_{\rho} \in V \) for all \( \rho \).

The soundness of this definition is proved in [10]: we state this fact in the next lemma.

Lemma 4.3 If \( M =_\rho N \) then \( [[M]]^M_\rho = [[N]]^M_\rho \), for any model \( M \) and environment \( \rho \).

In analogy with [2], we will define a model using the set \( F \) of filters over \( L \), generated by the \( B^+ \) implication \( \leq \). The set of \( \lor \)-prime filters, as suggested by the remark at the end of the previous section, is meant as the set of values. There is, however, a slight mismatch between the present notion of filter and the standard one: we shall admit \( \emptyset \) as an element of \( F \).

As a matter of fact, we shall construct a model of the call-by-value \( \lambda \)-calculus out of the set of filters: the denotation of a term \( M \) is a filter, namely the set of all types (formulae) that can be assigned to \( M \). The fact that the empty set is considered as a filter is a consequence of our choice to use \( \nu \) as the type of values, and of the fact that closed terms which are not \( \beta_\nu \)-convertible to a value have no type at all.

Definition 4.4 (Filters)

i) A filter over \( B^+ \) is a set \( X \subseteq L \) such that
- if \( A \leq B \) and \( A \in X \) then \( B \in X \);
- if \( A, B \in X \), then \( A \land B \in X \).

ii) Let \( F \) be the set of all filters over \( B^+ \).

iii) A filter \( X \) over \( B^+ \) is a \( \lor \)-prime filter if \( A \lor B \in X \) implies either \( A \in X \) or \( B \in X \).

iv) Let \( P F \) be the set of \( \lor \)-prime filters on \( B^+ \) different from \( \emptyset \).

v) If \( X \subseteq L \), \( \uparrow X \) denotes the filter generated by \( X \).

vi) \( \uparrow A \) is short for \( \uparrow \{ A \} \).

\( \langle \mathcal{F}, \subseteq \rangle \) is a distributive lattice. As a domain it is algebraic, with finite (or compact) elements of the shape \( \uparrow A \). Now we observe that \( \uparrow B \) is \( \lor \)-prime if and only if \( B \) is \( \lor \)-prime; since any \( A \) is equivalent to a finite disjunction \( A_1 \lor \cdots \lor A_k \) of \( \lor \)-prime formulae, we conclude that any finite element of \( \mathcal{F} \) (namely any principal filter) factorizes into a finite intersection of \( \lor \)-prime filters: \( \uparrow (A_1 \lor \cdots \lor A_k) = \uparrow A_1 \cap \cdots \cap \uparrow A_k \).

\( \mathcal{F} \) is also a solution of the domain equation

\[ D \triangleleft [D \to \bot D]_{\bot} \]

in the category of algebraic lattices, where \( [D \to \bot D]_{\bot} \) denotes the lifted space of strict continuous functions from \( D \) to \( D \). In fact \( \emptyset \) and \( \uparrow \nu \) are respectively the bottom of \( \mathcal{F} \) and of \( [\mathcal{F} \to \bot \mathcal{F}] \). Moreover \( \uparrow (A \to B) \) represents the strict step function \( f_{\uparrow A, \uparrow B} \), where as usual

\[ f_{a,b}(d) = \text{if } a \sqsubseteq d \text{ then } b \text{ else } \bot. \]

So, we can define the projection pair \( \langle F, G \rangle \), where \( F : \mathcal{F} \to [\mathcal{F} \to \bot \mathcal{F}]_{\bot} \), \( G : [\mathcal{F} \to \bot \mathcal{F}]_{\bot} \to \mathcal{F} \), and \( F \circ G = \text{Id}_{[\mathcal{F} \to \bot \mathcal{F}]_{\bot}} \), as follows:

\[ F(X) = \bigsqcup \{ f_{\uparrow A, \uparrow B} \mid A \to B \in X \} \]

\[ G(f) = \uparrow \{ A \to B \mid B \in f(\uparrow A) \} \cup \uparrow \nu. \]

Due to the presence of union types, we do not have an isomorphism between \( \mathcal{F} \) and \( [\mathcal{F} \to \bot \mathcal{F}]_{\bot} \), since different filters represent the same function. For example, \( \uparrow (A \to B) \lor (C \to D) \sqsubseteq \uparrow A \land C \to B \lor D \) and the inclusion is proper, but \( F(\uparrow (A \to B) \lor (C \to D)) = F(\uparrow \)
Lemma 4.6 i) $X \cdot Y = \{ A \mid \exists B \in Y, B \rightarrow A \in X \}$.

Observe that $X \cdot \emptyset = \emptyset \cdot Y = \emptyset$.

Lemma 4.6 ii) $X \cdot Y \in \mathcal{F}$ for all $X, Y \in \mathcal{F}$.

Proof: i) First $X \cdot Y$ is upward closed: if $A \in X \cdot Y$ and $A \leq B$ then $C \rightarrow A \in X$, for some $C \in Y$. Then, by (R5), $C \rightarrow A \leq C \rightarrow B$, hence $C \rightarrow B \in X$, which is upward closed, so $B \in X \cdot Y$.

We now show that $X \cdot Y$ is closed under $\land$: let $A, B \in X \cdot Y$. By definition, there exist $A', B'$ such that $A', B' \in Y$, $A' \rightarrow A, B' \rightarrow B \in X$. By (R4), $A' \rightarrow A \leq (A' \land B') \rightarrow A$, and, similarly, $B' \rightarrow B \leq (A' \land B') \rightarrow B$. By (A7), $((A' \land B') \rightarrow A) \land ((A' \land B') \rightarrow B) \leq (A' \land B') \rightarrow (A \land B) \in X$, since $X$ is a filter. But $A' \land B' \in Y$, since $Y$ is a filter too, so we conclude that $A \land B \in X \cdot Y$.

ii) $(\Leftarrow)$ Let $A \lor B \in X$, then $C \leq A \lor B$, for some $\lor$-prime $C \in X$. Then immediately $C \leq A$ or $C \leq B$, so $A \in X$ or $B \in X$ since $X$ is upward closed.

$(\Rightarrow)$ By Proposition A.3, $B \sim m_{\lor}(B) = \bigvee_{i \in I} B_i$, where the $B_i$'s are $\lor$-prime by Theorem 2.6 (ii). As $X$ is $\lor$-prime, $B_i \in X$ for some $i \in I$, and clearly $B_i \leq B$.

The basic idea of the next definition is to interpret terms into $\mathcal{F}$ using the Leibnitzian principle for which objects are identified with the set of their properties (here formulae, or types).

Definition 4.7 (Term interpretation)

i) A basis $\Gamma$ agrees with an environment $\rho : \text{Var} \rightarrow \mathcal{P} \mathcal{F}$ (notation $\Gamma \models \rho$) iff $x : B \in \Gamma$ implies $B \in \rho(x)$.

ii) The interpretation of $\lambda$-terms induced by $\vdash$ is defined by

$$[[M]]^\mathcal{F}_\rho = \{ A \in L \mid \exists \Gamma \models \rho, \Gamma \vdash M : A \}.$$ 

The mapping $[[M]]^\mathcal{F}_\rho$ is actually an interpretation from $\Lambda$ to $\mathcal{F}$, as stated in the next Lemma.

Lemma 4.8 $[[M]]^\mathcal{F}_\rho \in \mathcal{F}$, for any $M \in \Lambda$ and environment $\rho : \text{Var} \rightarrow \mathcal{P} \mathcal{F}$.

Proof: By rules $(\leq)$ and $(\land)$.

The main fact we establish now about $\mathcal{F}$ is that, given the above definitions of application and interpretation, it is a (filter) model of the call-by-value $\lambda$-calculus.

Theorem 4.9 $\mathcal{M}_0 = \langle \mathcal{F}, \mathcal{P} \mathcal{F}, \cdot, [[\cdot]]^\mathcal{F}_\rho \rangle$ is a model of call-by-value $\lambda$-calculus.

Proof: By checking that all conditions of Definition 4.2 are satisfied.

i) If $A \in [[x]]^\mathcal{F}_\rho$, then $\Gamma \vdash x : A$ for some $\Gamma$ such that $\Gamma \models \rho$. If $A \not\equiv \emptyset$ (otherwise the thesis follows from the fact that $\rho(x) \neq \emptyset$) then $x : B \in \Gamma$ for some $B \leq A$, by Lemma 3.4 (i). This implies that $B \in \rho(x)$ and, therefore, also that $A \in \rho(x)$.

On the other hand, if $A \in \rho(x)$, then $\rho \models \{ x : A \}$ and $x : A \vdash x : A$. 

10
ii) Immediate by Lemma 3.4 (ii).

iii) Let $X \in \mathcal{P} \mathcal{F}$, then

$$A \in [\lambda x.M]_{\rho}^F \cdot X \Rightarrow \exists B \in X. B \rightarrow A \in [\lambda x.M]_{\rho}^F$$

$$\Rightarrow \exists C \in X. C \land-\text{prime} \& C \rightarrow A \in [\lambda x.M]_{\rho}^F$$

$$\Rightarrow \exists C \in X. \Gamma \land-\text{prime} \& \Gamma \models \rho \& \Gamma \vdash \lambda x.M : C \rightarrow A$$

$$\Rightarrow \exists C \in X. \Gamma. x : C \models \rho[x := X] \& \Gamma, x : C \vdash M : A$$

$$\Rightarrow A \in [M]_{\rho[x := X]}^F$$

Vice versa

$$A \in [M]_{\rho[x := X]}^F \Rightarrow \exists B, \Gamma. \Gamma, x : B \models \rho[x := X] \& \Gamma, x : B \vdash M : A$$

$$\Rightarrow \exists B \in X. \Gamma. \Gamma \models \rho \& \Gamma \vdash \lambda x.M : B \rightarrow A$$

$$\Rightarrow \exists B \in X. B \rightarrow A \in [\lambda x.M]_{\rho}$$

$$\Rightarrow A \in [\lambda x.M]_{\rho} \cdot X.$$  

iv) vi), vii) Easy.

vii) If $M \equiv x \in \text{Var}$, then $[M]_{\rho}^F = \rho(x) \in \mathcal{P} \mathcal{F}$, because $\rho$ is a mapping from $\text{Var}$ to $\mathcal{P} \mathcal{F}$. Otherwise, suppose that $M \equiv \lambda x.M'$: then by Lemma 4.6 (ii), it suffices to prove that

$$B \in [\lambda x.M'][\rho] \Rightarrow \exists C \in [\lambda x.M'][\rho]. C \leq B \land C \text{ is } \land-\text{prime}.$$  

Assume $B \in [\lambda x.M'][\rho]$, i.e. $\exists \Gamma \models \rho, \Gamma \vdash \lambda x.M' : B$. The case $B \not\sim v$ is trivial; let us suppose that $B \sim v$. By Lemma 3.4 (iii) we have that, for some $I, B_i$ and $C_i$,

$$\forall i \in I. \Gamma, x : B_i \vdash M' : C_i \text{ and } \land_{i \in I} (B_i \rightarrow C_i) \leq B.$$  

Then $\land_{i \in I} (B_i \rightarrow C_i) \in [\lambda x.M'][\rho]$ by $(\rightarrow I)$ and $(\land I)$, where $\land_{i \in I} (B_i \rightarrow C_i)$ is $\land$-prime by Theorem 2.6 (ii).

5 Conclusion

The fact that the system $B+$ gives naturally a type assignment system for the call-by-value $\lambda$-calculus is a pleasant surprise. Indeed, something more should be true about our system. First we strongly conjecture that an approximation theorem holds: given the right notion of approximant in the case of call-by-value $\lambda$-calculus (see e.g. [6]), we expect that the set of types that can be assigned to any term is exactly the set of all types that can be assigned to its approximate normal forms. Then some leading ideas of our construction, such as the fact that closed terms which can be typed by $v$ are exactly the convergent “programs”, would have a clear and elegant proof.

There are, however, some open questions. First, we do not have a completeness theorem for the $B+$ based type assignment system along the lines of [2]. A deeper analysis of the correspondence between the filter model construction and the semantics of relevant logics is still on demand, and, we guess, should lead to a better understanding even of the results we presently have.

References

Appendix A  Properties of $\leq$

Specialisations of $\leq$ to the sets $T_i$ are now introduced, whose definition exploits the syntactical form of the types in $T_i$.

Definition A.1 $\leq_i \subseteq T_i \times T_i (i = \rightarrow, \forall, \land, \lor, \land, \lor, \land)$ are the least preorders such that

$(\leq_\rightarrow)$: $A \leq_\rightarrow B \iff$ either $A = B$ or $B = \nu$ or $A = A_1 \rightarrow A_2, B = B_1 \rightarrow B_2$ and $B_1 \leq_\land A_1, A_2 \leq_\lor B_2$

$(\leq_\land)$: $\land_{i \in I} A_i \leq_\land \land_{j \in J} B_j$ (where $A_i, B_j \in T_\rightarrow$) $\iff \forall i \exists j \in J, A_i \leq_\rightarrow B_j$

$(\leq_\lor)$: $\lor_{i \in I} A_i \leq_\lor \lor_{j \in J} B_j$ (where $A_i, B_j \in T_\land$) $\iff \forall j \exists i \in I, A_i \leq_\land B_j$

$(\leq_\land)$: $\land_{i \in I} A_i \leq_\land \land_{j \in J} B_j$ (where $A_i, B_j \in T_\lor$) $\iff \forall j \exists i \in I, A_i \leq_\lor B_j$

$(\leq_\land)$: $\lor_{i \in I} A_i \leq_\lor \lor_{j \in J} B_j$ (where $A_i, B_j \in T_\land$) $\iff \forall i \exists j \in J, A_i \leq_\land B_j$.

Lemma A.2 $\leq_i (i = \rightarrow, \forall, \land, \lor, \land, \lor, \land)$ are reflexive and transitive.

Proof: By induction on the definition of $\leq_i$.

The following proposition states that conjunctive/disjunctive normal forms are logically equivalent to their counterimages under $m_{\land\lor}()$ and $m_{\lor\land}()$, and that the specialised relations $\leq_i$ are actually restrictions of $\leq$ to the sets $T_i$ respectively.

Property A.3 For all $A, B \in L$:

i) $A \sim m_{\land\lor}(A) \sim m_{\lor\land}(A)$.

ii) $A, B \in T_i, A \leq_i B \Rightarrow A \leq B$ for $i = \rightarrow, \forall, \land, \lor, \land, \lor, \land$.

iii) $A \leq B \iff m_{\land\lor}(A) \leq_{\land\lor} m_{\lor\land}(B) \iff m_{\lor\land}(A) \leq_{\lor\land} m_{\land\lor}(B)$.

Proof: i) By induction on $A$. E.g. if $A = B \rightarrow C$ then, by induction hypothesis, we have $B \sim m_{\lor\land}(B) = \lor_{i \in I} B_i$ and $C \sim m_{\land\lor}(C) = \land_{j \in J} C_j$, so that, by repeated uses of (A7), (A8), (R4) and (R5) we conclude that

$$B \rightarrow C \sim \lor_{i \in I} B_i \rightarrow \land_{j \in J} C_j \sim \land_{i \in I} \land_{j \in J} (B_i \rightarrow C_j) \sim m_{\land\lor}(B \rightarrow C) = m_{\lor\land}(B \rightarrow C).$$
Proof: i) Let

\[ m_{\land \lor}(A) \leq m_{\land \lor}(B) \Rightarrow \forall j \in J \forall i \in I \forall p \in I_i \exists q \in J_j, A_{ip} \leq B_{jq}, \]

where \( m_{\land \lor}(A) = \land_{i \in I} A_i, m_{\land \lor}(A_i) = \lor_{p \in I_i} A_{ip} \), and \( m_{\land \lor}(B) = \land_{i \in J} B_i, m_{\land \lor}(B_i) = \lor_{q \in J_i} B_{iq} \).

Similarly,

\[ m_{\land \lor}(C) \leq m_{\land \lor}(D) \Rightarrow \forall l \in L \exists k \in K \forall r \in K_k \exists s \in L_l, C_{kr} \leq D_{ls}, \]

where \( m_{\land \lor}(C) = \land_{k \in K} C_k, m_{\land \lor}(C_k) = \lor_{l \in L} C_{lk} \) and \( m_{\land \lor}(D) = \land_{l \in L} D_l, m_{\land \lor}(D_l) = \lor_{s \in L} D_{ls} \).

Then we have

\[ \forall j \in J, l \in L [ \exists i \in I \forall p \in I_i \exists q \in J_j, A_{ip} \leq B_{jq} \land \exists k \in K \forall r \in K_k \exists s \in L_l, C_{kr} \leq D_{ls} ] \]

\[ \Rightarrow \forall j \in J, l \in L [ \exists i \in I, k \in K, \lor_{p \in I_i} A_{ip} \lor_{r \in K_k} \land_{s \in L_l} C_{rk} \leq \lor_{q \in J_j} B_{jq} \lor_{s \in L_l} D_{ls} ] \]

\[ \Rightarrow \forall j \in J, l \in L [ \exists i \in I, k \in K, A_i \land C_k \leq \land \lor_{q \in J_j} B_{jq} \lor_{s \in L_l} D_{ls} ] \]

\[ \Rightarrow \land_{i \in I} \land_{k \in K} (A_i \lor C_k) \leq m_{\land \lor}(A \land C) \land_{i \in I} \land_{k \in K} (B_j \lor D_l) \]

\[ \Rightarrow m_{\land \lor}(A \land C) \leq m_{\land \lor}(B \lor D). \]

We come eventually to the proof of Theorem 2.6.

**Theorem A.4 (Properties of \( \lor \)-prime formulas)**

i) \( \forall \) and any propositional variable are \( \lor \)-prime.

ii) The formula \( \land_{i \in I} (A_i \rightarrow B_i) \) is \( \lor \)-prime for all (finite) \( I \) and formulas \( A_i, B_i \).

iii) If \( m_{\lor \land}(A_i) = \lor_{i \in I} A_i \) then \( A_i \) is \( \lor \)-prime, for all \( i \in I \).

iv) \( B \rightarrow C \neq \forall \) for all \( B, C \).

v) \( \land_{i \in I} (A_i \rightarrow B_i) \leq C \rightarrow D \) and \( C \) is \( \lor \)-prime imply \( C \leq \land_{i \in I} A_i \) and \( \land_{i \in I} B_i \leq D \) for some \( J \subseteq I \).

Proof: i) Let \( A \in \mathcal{PV} \cup \{ \forall \} \), and suppose that \( A \leq B \lor C \). Then \( m_{\lor \land}(A) = A \), while

\[ m_{\lor \land}(B \lor C) = \lor_{j \in J} B_j \lor \land_{k \in K} C_k, \]

where \( m_{\lor \land}(B) = \lor_{j \in J} B_j \) and \( m_{\lor \land}(C) = \land_{k \in K} C_k \). By Proposition A.3 (iii), we have \( A \leq \land \lor_{j \in J} B_j \lor \land_{k \in K} C_k \), so that, by Definition A.1, \( A \leq \land B_j \) or \( A \leq \land C_k \) for some \( j, k \); then \( A \leq B_j \leq B \) or \( A \leq C_k \leq C \) by Proposition A.3 (iii).

ii) By Proposition A.3 (iii) we have:

\[ \land_{i \in I} (A_i \rightarrow B_i) \leq C \lor D \Leftrightarrow m_{\lor \land}(\land_{i \in I} (A_i \rightarrow B_i)) \leq m_{\lor \land}(C \lor D). \]

Now \( m_{\lor \land}(\land_{i \in I} (A_i \rightarrow B_i)) \) is a conjunction of arrows, namely a formula with no disjunction at the top level; on the other hand \( m_{\lor \land}(C \lor D) \) has the form \( \lor_{k \in K} C_k \lor \land_{l \in L} D_l \). By definition of \( \leq_{\lor \land} \), we immediately have that \( m_{\lor \land}(\land_{i \in I} (A_i \rightarrow B_i)) \leq \land C_k \) or \( m_{\lor \land}(\land_{i \in I} (A_i \rightarrow B_i)) \leq \land D_l \), for some \( k, l \); therefore the thesis follows by Proposition A.3 (i) and (ii).

iii) Parts (i) and (ii) imply that any formula in \( T_{\land} \) is \( \lor \)-prime: hence the thesis follows by the definition of \( m_{\lor \land}(\cdot) \).

iv) By contra-position. Suppose that \( \forall \leq B \rightarrow C \); then, by Proposition A.3 (iii), \( \forall \leq \land \lor m_{\lor \land}(B \rightarrow C) = \land_{i \in I, j \in J} (B_i \rightarrow C_j) \). This implies that there exist \( i \) and \( j \) such that \( \forall \leq B_i \rightarrow C_j \), which is not.

v) Let first compute:

\[ m_{\lor \land}(\land_{i \in I} (A_i \rightarrow B_i)) = \land_{i \in I} [ \land_{k \in K} \land_{l \in L} (A_{ik} \rightarrow B_{lj}) ], \]
where $\text{m}_\wedge(A_i) = \bigvee_{h \in H_i} A_{i,h}$, and $\text{m}_\wedge(B_i) = \wedge_{l \in L_i} B_{i,l}$. On the other hand suppose that $\text{m}_\wedge(C \rightarrow D) = \wedge_{p \in P} \wedge_{q \in Q} (C_p \rightarrow D_q)$, where $\text{m}_\wedge(C) = \bigvee_{p \in P} C_p$, and $\text{m}_\wedge(D) = \wedge_{q \in Q} D_q$. By Proposition A.3 (iii) and the definition of $\leq_{\wedge\vee}$ we have

$$\forall p \in P, q \in Q \exists i \in I, h \in H_i, l \in L_i, C_p \leq \wedge A_{i,h} \wedge B_{i,l} \leq \vee D_q.$$  

By Proposition A.3 (i), $C \sim \bigvee_{p \in P} C_p$: hence, since $C$ is $\vee$-prime, there exists $p \in P$ such that $C \leq C_p$. Choose one such $p$ and, for any $q \in Q$, define

$$J_q = \{i | \exists i \in I, h \in H_i, l \in L_i, C_p \leq \wedge A_{i,h} \wedge B_{i,l} \leq \vee D_q\},$$

which is non-empty by the above statement. Finally, we take $J = \bigcup_{q \in Q} J_q$. Now, for all $i \in J$, there exists $\exists i \in H_i$ such that $C_p \leq A_{i,h} \wedge A_i$: therefore $C \leq C_p \leq \wedge_{i \in J} A_i$.

To conclude, for all $q \in Q$ there is $i \in J_q$ and $l \in L_i$ such that $B_i \leq B_{i,l} \leq D_q$: then $\wedge_{i \in J} B_i \leq D_q$ for all $q$, and, therefore, $\wedge_{i \in J} B_i \leq \wedge_{q \in Q} D_q \sim D$. 

14