Three Deductive Systems of Classical (or Boolean) Type Theory and Their Denotational-Semantic Completeness

Ken Akiba

Virginia Commonwealth University, Richmond, Virginia, USA
kakiba@vcu.edu

Abstract

Classical (or *Boolean*) type theory is a non-standard type theory that allows for the type inference $(A \to \bot) \to \bot \vdash A$, the type counterpart of the double-negation elimination rule, where A is any type and $\bot$ is absurdity type. This paper considers three deductive systems of classical type theory: (1) a λ-calculus in the natural deduction form called *λK-calculus*; (2) *classical double sequent calculus (LK2)*; and (3) $LK2$'s natural deduction version, *classical double natural deduction (NK2)*. A denotational semantics is given to each calculus. The semantics have a common domain structure, called *the infinitely nested Boolean structure*, which divides each type domain into infinitely many *ranks*, each of which forms a Boolean algebra. Absurdity type $\bot$ is identified with the type of truth values, and the notions of semantic consequence are also defined in terms of truth values. Each calculus is proved sound and complete with respect to its semantics.

In the process, this paper aims to give a definitive answer to the outstanding question of how the proof-annotated natural deduction and sequent calculus for classical propositional logic can be interpreted from the viewpoint of proofs (or programs)-as-terms (propositions (or formulas)-as-types, or the Curry-Howard correspondence).

Keywords: Type theory; Classical (or Boolean) type theory; λ-calculus; $\lambda\mu$-calculus; Natural deduction; Sequent calculus; Curry-Howard correspondence; Propositions (or formulas) as types; Proofs (or programs) as terms; Classical logic; Double-negation elimination; Infinitely nested Boolean structure; Type reduction; Denotational semantics; Completeness; Generalized quantifier theory

1 Introduction

Classical type theory is the type theory that allows for the type inference $(A \to \bot) \to \bot \vdash A$, where A is any type and $\bot$ is absurdity type. This rule is the type counterpart of the double-negation elimination rule (where $\neg A = A \to \bot$), the signature inference rule of classical logic. The standard logic of type inferences, minimal or intuitionistic logic, does not have this rule. Classical type theory, thus, is a non-standard type theory. However, the term 'classical type theory' can be used in other contexts to refer to the traditional type theory, say Church's or Curry's type theory. To distinguish it from such a type theory, classical type theory of our sense may also be called *Boolean* type theory.

The most famous and influential deductive system of classical type theory to date is Parigot's $\lambda\mu$-calculus [9]. $\lambda\mu$-calculus, however, is not entirely intuitive. If you start with each deduction rule in, say, the standard classical natural deduction or the standard classical sequent calculus, and prefixed each proposition in a proof with an encoding of its proof indicating how it was derived, the resulting system of proof-proposition pairs will not exactly be $\lambda\mu$-calculus. Among other things, $\lambda\mu$-calculus involves the distinction between λ- and μ-variables that cannot be made immediate sense of in this approach. In the next section of this paper, we present three calculi of classical type theory, two in natural deduction and one in sequent calculus, which are

all perfect in this respect. The first is called *λK-calculus*, the second is called *classical double sequent calculus (LK2)*, and the third is the natural deduction version of the second, called *classical double natural deduction (NK2)*. The last two are so called because their terms have types that are essentially of the same forms as the terms themselves (for instance, if term M is of type A and N of type B, then term $M \to N$ will be of type $A \to B$).

After Section 2, we will give denotational semantics to the three calculi. Per the Curry-Howard correspondence (see, e.g., [14]), the calculi ought to be understood as calculi of typed terms instead of (or as well as) those of proof-annotated propositions, and semantics are given accordingly. Those semantics have the common core, which is the same domain structure we call *the infinitely nested Boolean structure*. It divides each type domain into infinitely many *ranks*, each of which forms a Boolean algebra. Absurdity type $\bot$ is identified with the type of truth values. The double-negation elimination, as a type inference, then is interpreted as a move from type $(A \to \bot) \to \bot$, rank n, to type A, rank $n+1$. The notions of semantic consequence are defined also in terms of truth values, falsity in particular. In Section 4, each calculus is proved sound and complete with respect to its semantics.

Even though our formulation of classical λ-calculus as λK-calculus is not without its own merits, we are inclined to think that our formulation of classically-typed sequent calculus as classical double sequent calculus ($LK2$) is of particular significance. Unlike previous formulations of sequent calculus in terms of λ (such as [7], which deals only with minimal logic), our formulation is independent of λ-calculus. This seems more natural because unlike natural deduction, propositional sequent calculus does not involve subproofs and genuine variables. Put simply, while $[\![(\lambda x_A M_B)_{A\to B}]\!]_\rho$ is usually the total function that gives the value $[\![M_B]\!]_\rho$ for the argument $[\![x_A]\!]_\rho$ and other values $[\![(M[x_A := N_A])_B]\!]_\rho$s for the other arguments $[\![N_A]\!]_\rho$s accordingly, $[\![(M_A \to N_B)_{A\to B}]\!]_\rho$ is a (very) partial function that gives the value $[\![N_B]\!]_\rho$ (or something like it) for the argument $[\![M_A]\!]_\rho$ but is undefined for all the other arguments independent of $[\![M_A]\!]_\rho$. Whereas natural deduction is paired with λ-calculus and Hilbert-type axiomatic systems are with combinatory logic in the Curry-Howard correspondence, the third of the three main formalizations of logic (see, e.g., [15]), sequent calculus, has never been quite understood from that point of view. We hope that our interpretation of typed sequent calculus as having terms and types of essentially the same form is correct and contributes to a better understanding of sequent calculus in general from the viewpoint of the Curry-Howard correspondence.

2 λK, $LK2$, and $NK2$

λK-calculus, $LK2$, classical double sequent calculus, and its natural deduction version $NK2$ involve the same type system. As usual, there are two kinds of types: base (or atomic) types, denoted as X below, and function types. The types are defined thus:

$$A ::= X \mid \bot \mid A \to A$$

What is noteworthy here is the inclusion of $\bot$ as a (or *the*) type constant. $\bot$ is called *absurdity type*.

Throughout this paper, $\neg A$, whether it appears as a term or as a type, in the object language or metalanguage, is uniformly taken as the abbreviation of $A \to \bot$.

All three calculi we present are actually their $\{\to, \bot\}$-fragments. However, since they are classical calculi, other operators such as $\wedge$ and $\vee$ can be defined easily in terms of $\to$ and $\bot$ (e.g., $A \wedge B =_{df} \neg(A \to \neg B)$ and $A \vee B =_{df} \neg A \to B$). We won't consider quantifiers $\forall$ and $\exists$ in this paper.

2.1 λK-Calculus

λK-calculus is a straightforward classical extension of minimal logic in natural deduction, which can be understood from the propositions (or formulas)-as-types viewpoint as the standard simply-typed $\lambda\beta$-calculus in the natural deduction formulation. λK adds the double-negation elimination rule to the calculus. The deductive system closest to ours is probably Rehof and Sørensen's [11] λ_Δ-calculus.

2.1.1 Terms of λK

All terms of $\lambda K, LK2$, and $NK2$ are subscripted with their types. The following is the definition of a term M_A (of type A) in λK-calculus:

$$M_A ::= x_A \mid (M_{A\to B} M_A)_B \mid (\lambda x_A M_B)_{A\to B} \mid (\downarrow_{((A\to\bot)\to\bot)\to A} M_{(A\to\bot)\to\bot})_A$$

where x is a variable. Here the same Roman capitals in the subscripts in the same item are the same types. Since $A \to \bot$ can be abbreviated as $\neg\neg A$, $((A \to \bot) \to \bot) \to A$ in the last item can be abbreviated as $\neg\neg A \to A$. Furthermore, this subscript is often simply omitted in what follows.

2.1.2 Deduction Rules of λK

λK-calculus consists of the following five deduction rules:

$$\frac{M_{A\to B} \qquad N_A}{(M_{A\to B} N_A)_B}\,(Ap) \qquad \frac{\begin{array}{c}[x_A]\\ \vdots \\ M_B\end{array}}{(\lambda x_A M_B)_{A\to B}}\,(\lambda I) \qquad \frac{((\lambda x_A M_B)_{A\to B} N_A)_B}{(M[x_A := N_A])_B}\,(\lambda E)$$

$$\frac{M_{\neg\neg A}}{(\downarrow M_{\neg\neg A})_A}\,(\downarrow I) \qquad \frac{(N_{\neg A}(\downarrow M_{\neg\neg A})_A)_\bot}{(M_{\neg\neg A} N_{\neg A})_\bot}\,(\downarrow E)$$

$[x_A]$ means that the premise x_A may be discharged at the introduction of λx_A. $(M[x_A := N_A])_B$ is the same as M_B except that all occurrences of variable x_A in M_B is replaced with occurrences of term N_A.

For any set Γ of terms and any term N_B, $\Gamma \vdash_{\lambda K} N_B$ iff N_B is deducible from Γ by means of the above five rules.

λ-introduction (λI) and λ-elimination (λE) are more often called *abstraction* and *β-reduction*, respectively. Among the five rules, the first three, i.e., application (Ap), (λI), and (λE) are the standard deduction rules of the simply-typed $\lambda\beta$-calculus. (Ap) and (λI) can be considered the proof-annotated versions of the $\to$-elimination and -introduction rules in minimal logic. $(\downarrow I)$, from that viewpoint, is the double-negation $(\neg\neg)$ elimination rule.

As proof-annotated classical deduction rules, (λE) and $(\downarrow E)$ may seem to be useless, as their premises and conclusions have the same propositions (i.e., subscripts B and $\bot$, respectively) although, as λ-calculus they have obvious roles. However, even as proof-annotated classical deduction rules they have the role of *canceling* the previous use of (λI) and $(\downarrow I)$, respectively. More on this feature, see Subsection 2.4 below.

2.2 Classical Double Sequent Calculus ($LK2$)

Just as classical type theory, sequent calculus has largely been left aside in the discussion of the Curry-Howard correspondence (even though there are well-known calclui that deal with symmetry and duality of computation such as [1] and [4]). These two omissions are not unrelated. Among the three major systems of deduction, natural deduction, Hilbert-type (axiomatic) system, and sequent calculus, natural deduction corresponds to λ-calculus and Hilbert-type system corresponds to combinatory logic, but we have not yet found out what kind of type inference system sequent calculus corresponds to. But sequent calculus seems to be the most natural vehicle for classical inferences: classical inferences seem more natural than merely intuitionistic ones in the sequent calculus setting, for the calculus allows more than one succedent, thus creating the perfect symmetry between the antecedents and the succedents. In contrast, the intuitionistic system seems crippled, unnaturally restricting the succedents into having at most one member. So constructing a sequent calculus of classical type theory and giving semantics go a long way toward a more complete understanding of the Curry-Howard correspondence. $LK2$ should be considered in that context.

2.2.1 Terms of $LK2$

$$M_A ::= x_A \mid \bot_\bot \mid (M_A \to M_B)_{A\to B}$$

where x_A are variables (of type A). $\bot_\bot$ is constant $\bot$ of absurdity type $\bot$. Similarly, $\to$ is used here in two ways, as the function type constructor and as the term constructor. As you will see, this will not create any ambiguity.

2.2.2 Deduction Rules of $LK2$

Here we present $LK2$ as a *one-sided* sequent calculus: what is ordinarily a member of the succedent is signaled as an $\neg$-formula. Generally, a one-sided sequent calculus can be considered an upside-down version of semantic tableaux [3] or truth trees [13].

$$\frac{}{\Gamma,\ \bot_\bot}\,(\bot) \qquad \frac{}{\Gamma,\ M_A,\ (\neg M_A)_{\neg A}}\,(\neg)$$

$$\frac{\Gamma,\ (\neg M_A)_{\neg A} \qquad \Gamma,\ N_B}{\Gamma,\ (M_A \to N_B)_{A\to B}}\,(\to) \qquad \frac{\Gamma,\ M_A,\ (\neg N_B)_{\neg B}}{\Gamma,\ (\neg(M_A \to N_B)_{A\to B})_{\neg(A\to B)}}\,(\neg\to)$$

where Γ is a sequent[1] of terms. Then $\neg$ introduced by the definition $\neg A = A \to \bot$ will be the classical negation.

For any set $\mathbb{S}$ of sequents and any sequent Δ, $\mathbb{S} \vdash_{LK2} \Delta$ iff Δ is deducible from $\mathbb{S}$ by means of the above four rules.

What is remarkable about the four rules of $LK2$ above is that the proof part of each proof-proposition pair (or the term part of each term-type pair) in the rules has basically the same form as its proposition (type) part: hence 'double' and '2' in its name (recall Gentzen's [6] original classical sequent calculus was called LK). A proposition derived in the standard sequent calculus has its proof on its sleeves: we can tell how it was derived merely from the form of the proposition. $LK2$ is a formalization of this idea. It actually involves only two basic proof rules, $(\bot)$ and $(\to)$. The other two rules are combinations of these two basic rules.

We could add to the above rules the famous (Cut):

[1] Actually, just a set. This obviates the need of the so-called *structural* rules such as weakening, contraction, and exchange. However, throughout this paper we shall use the term 'sequents' for the sake of convenience.

$$\frac{\Gamma, M_A \qquad \Gamma, (\neg N_A)_{\neg A}}{\Gamma}\,(\text{Cut})$$

As is well known, however, (Cut) can be shown to be eliminable by a purely deductive consideration [6], and it is indeed eliminated in the aforementioned semantic tableaux and truth trees. Thus, we shall not include (Cut) in $LK2$.

2.3 Classical Double Natural Deduction ($NK2$)

$NK2$ is the natural deduction version of $LK2$. As such, again, each of its terms has the type of the same form.

2.3.1 Terms of $NK2$

The terms of $NK2$ is the same as those of $LK2$.

2.3.2 Deduction Rules of $NK2$

The deduction rules of $NK2$ are as follows:

$$\frac{\begin{array}{c}[M_A]\\ \vdots \\ N_B\end{array}}{(M_A \to N_B)_{A\to B}}\,(\to I) \qquad \frac{(M_A \to N_B)_{A\to B} \qquad M_A}{N_B}\,(\to E)$$

$$\frac{\bot_\bot}{X_A}\,(\bot E) \qquad \frac{(\neg(\neg M_A)_{\neg A})_{\neg\neg A}}{M_A}\,(\neg\neg E)$$

In $(\bot E)$, X_A, just like M_A and N_B, is not a term but a metavariable (or term schema) to be replaced with any term.

For any set Γ of terms and any term N_B, $\Gamma \vdash_{NK2} N_B$ iff N_B is deducible from Γ by means of the above four rules.

While the equivalence between $NK2$ and $LK2$ is obvious (since Gentzen's original NK and LK are equivalent), $NK2$ is a little difficult to interpret as a proof-annotated calculus. All the rules except $(\to I)$ seem to cancel and erase the previous proof records; however, unlike (λE) and $(\downarrow E)$ in λK, the propositions different from the premises are derived in the conclusions. The significance of $NK2$, thus, seems to reside solely in the fact that it is a natural deduction version of $LK2$. Consequently, $NK2$ will be treated very cursorily near the end of this paper.

2.4 Some Important Deductions in λK

Before we begin considering the semantics of $\lambda K, LK2$, and $NK2$ in the next section, let us examine a few deductions of λK which exhibit some important features.

(1)
$$\frac{\dfrac{\dfrac{\begin{array}{c}[x_A]\\ \vdots \\ M_B\end{array}}{(\lambda x_A M_B)_{A\to B}}\,(\lambda I) \qquad N_A}{((\lambda x_A M_B)_{A\to B} N_A)_B}\,(Ap)}{(M[x_A := N_A])_B}\,(\lambda E)$$

(2)
$$\frac{\dfrac{N_{\neg A} \qquad \dfrac{M_{\neg\neg A}}{(\downarrow M_{\neg\neg A})_A}\,(\downarrow I)}{(N_{\neg A}(\downarrow M_{\neg\neg A})_A)_\bot}\,(Ap)}{(M_{\neg\neg A} N_{\neg A})_\bot}\,(\downarrow E)$$

$$(3)\quad \dfrac{\dfrac{(N_{\neg A}(\downarrow(\lambda x_{\neg A} M_{\bot})_{\neg\neg A})_A)_{\bot}}{((\lambda x_{\neg A} M_{\bot})_{\neg\neg A} N_{\neg A})_{\bot}}\,(\downarrow E)}{(M[x_{\neg A} := N_{\neg A}])_{\bot}}\,(\lambda E)$$

$$(4)\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[y_{\neg B}]\quad \dfrac{[z_{A\to B}]\quad [v_A]}{(z_{A\to B} v_A)_B}\,(Ap)}{(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot}}\,(Ap)}{(\lambda z_{A\to B}(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot})_{\neg(A\to B)}}\,(\lambda I)\quad \dfrac{\dfrac{[x_{\neg(A\to B)}]}{\vdots}}{\dfrac{M_{\bot}}{(\lambda x_{\neg(A\to B)} M_{\bot})_{\neg\neg(A\to B)}}\,(\lambda I)}\,(\downarrow I)}{((\lambda z_{A\to B}(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot})_{\neg(A\to B)}(\downarrow(\lambda x_{\neg(A\to B)} M_{\bot})_{\neg\neg(A\to B)})_{A\to B})_{\bot}}\,(Ap)}{((\lambda x_{\neg(A\to B)} M_{\bot})_{\neg\neg(A\to B)}(\lambda z_{A\to B}(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot})_{\neg(A\to B)})_{\bot}}\,(\downarrow E)}{(M[x_{\neg(A\to B)} := (\lambda z_{A\to B}(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot})_{\neg(A\to B)}])_{\bot}}\,(\lambda E)}{(\lambda y_{\neg B}(M[x_{\neg(A\to B)} := (\lambda z_{A\to B}(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot})_{\neg(A\to B)}])_{\bot})_{\neg\neg B}}\,(\lambda I)}{(\downarrow(\lambda y_{\neg B}(M[x_{\neg(A\to B)} := (\lambda z_{A\to B}(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot})_{\neg(A\to B)}])_{\bot})_{\neg\neg B})_B}\,(\downarrow I)}{(\lambda x_A(\downarrow(\lambda y_{\neg B}(M[x_{\neg(A\to B)} := (\lambda z_{A\to B}(y_{\neg B}(z_{A\to B} v_A)_B)_{\bot})_{\neg(A\to B)}])_{\bot})_{\neg\neg B})_B)_{A\to B}}\,(\lambda I)$$

Proofs (1) and (2) show how proofs involving $\to$ and $\neg\neg$ are normalized in λK. Again, (λE) and $(\downarrow E)$ do not derive new propositions but cancel the old, redundant proofs involving (λI) and $(\downarrow I)$ within the deductive system. Proofs (3) and (4) correspond to what Parigot [9] calls *renaming* and *structural reduction*, which, together with β-reduction (λE), constitute the three reduction rules of his $\lambda\mu$-calculus. In particular, (4) cancels the initial derivation of $A \to B$ by double-negation elimination on $\neg\neg(A \to B)$ on the left-hand side and replaces it with a derivation involving double-negation elimination on $\neg\neg B$. By repeating this procedure, we can always end up with proofs involving double-negation elimination only on double negated *atomic* propositions. This plays an essential role in Prawitz's [10] normalization of classical proofs.

3 The Domain Structure

3.1 The Infinitely Nested Boolean Structure

In the remainder of this paper, we shall consider the denotational semantics of the three calculi, λK, $LK2$, and $NK2$, presented above. The question is: what sort of structure can the three deductive systems be thought to represent? The most important feature of our semantics is the structure of the domains common to all the semantics. It is called *the infinitely nested Boolean structure*. In this structure, each type domain is divided into infinitely many *ranks* (≥ 0). For any type A, $D^0_A \subset D^1_A \subset D^2_A \subset ...$, where the superscripts indicate ranks, and the whole domain for type A, $D_A = \bigcup D^n_A$. The domain D^n_A of type A, rank n, has operators $-^n_A, \sqcap^n_A$, and $\sqcup^n_A$, constituting a Boolean algebra. We need this complex structure because when the denotation of $M_{\neg\neg A}$ is in the original domain of type $\neg\neg A$, $D^0_{\neg\neg A}$, the denotation of $(\downarrow M_{\neg\neg A})_A$ cannot be in the original domain of type A, D^0_A, as even a simple cardinality consideration may reveal; so it must be expanded into D^1_A. But then the added members of D^1_A must land on an expansion of D^1_A, D^2_A, etc. Furthermore, since $D^0_{A\to B}$ must also be expanded into $D^1_{A\to B}$, $D^2_{A\to B}$, etc., for the same reason, how those added members of type A and type $A \to B$ ought to interact with

one another must be dealt with carefully. The key results are summarized in Subsubsection 3.1.3.

More specifically, we define the type domains and their structures for our semantics by mathematical induction on the complexity of types.

In what follows, any object a properly in D_A, i.e., object a such that $a \in D_A^n$ but $a \notin D_A^{n-1}$, is indicated as a_A^n. Any object that is in D_A^n but may not be properly so is indicated as $a_A^{\leq n}$.

This is not essential, but for the sake of simplicity we take absurdity type $\bot$ to be the sole base (or atomic) type of our model. We also take the function type $A \to B$ to be the type of total *or partial* (not just total) functions from the objects of type A to the objects of type B. For the semantics of λK, this inclusion of partial functions is actually not necessary, but for the semantics of $LK2$ and $NK2$, it is necessary. In addition to their different uses in the formal (object) languages, '$\bot$' and '$\to$' will also be used in the metalanguage describing the semantics.

3.1.1 Base

- The sole base type: $\bot$.
- $D_\bot^0 = \{\mathsf{T}_\bot^0, \mathsf{F}_\bot^0\}$.
- For any $p_\bot^0$ and $q_\bot^0$, $(-_\bot^0 p_\bot^0)_\bot^0$, $(p_\bot^0 \sqcap_\bot^0 q_\bot^0)_\bot^0$, and $(p_\bot^0 \sqcup_\bot^0 q_\bot^0)_\bot^0$ are defined simply as the ordinary negation, conjunction, and disjunction:

$p_\bot^0$	$(-_\bot^0 p_\bot^0)_\bot^0$
$\mathsf{T}_\bot^0$	$\mathsf{F}_\bot^0$
$\mathsf{F}_\bot^0$	$\mathsf{T}_\bot^0$

$\sqcap_\bot^0$	$\mathsf{T}_\bot^0$	$\mathsf{F}_\bot^0$
$\mathsf{T}_\bot^0$	$\mathsf{T}_\bot^0$	$\mathsf{F}_\bot^0$
$\mathsf{F}_\bot^0$	$\mathsf{F}_\bot^0$	$\mathsf{F}_\bot^0$

$\sqcup_\bot^0$	$\mathsf{T}_\bot^0$	$\mathsf{F}_\bot^0$
$\mathsf{T}_\bot^0$	$\mathsf{T}_\bot^0$	$\mathsf{T}_\bot^0$
$\mathsf{F}_\bot^0$	$\mathsf{T}_\bot^0$	$\mathsf{F}_\bot^0$

- $D_\bot^1$ has 16 members: $\mathsf{T}_\bot^0, \mathsf{F}_\bot^0, (-_\bot^1 \mathsf{T}_\bot^0)_\bot^1, (-_\bot^1 \mathsf{F}_\bot^0)_\bot^1, (\mathsf{T}_\bot^0 \sqcap_\bot^1 \mathsf{F}_\bot^0)_\bot^1, (\mathsf{T}_\bot^0 \sqcup_\bot^1 \mathsf{F}_\bot^0)_\bot^1$, etc., where $(-_\bot^1(-_\bot^1 \mathsf{T}_\bot^0)_\bot^1)_\bot^1 = \mathsf{T}_\bot^0, (\mathsf{T}_\bot^0 \sqcap_\bot^1 \mathsf{T}_\bot^0)_\bot^1 = \mathsf{T}_\bot^0, ((-_\bot^1 \mathsf{T}_\bot^0)_\bot^1 \sqcap_\bot^1 (-_\bot^1 \mathsf{F}_\bot^0)_\bot^1)_\bot^1 = (-_\bot^1(\mathsf{T}_\bot^0 \sqcup_\bot^1 \mathsf{F}_\bot^0)_\bot^1)_\bot^1$, etc.; that is, $-_\bot^1, \sqcap_\bot^1$, and $\sqcup_\bot^1$ are Boolean operators connecting $\mathsf{T}_\bot^0$ and $\mathsf{F}_\bot^0$.
- $D_\bot^2$, analogously, consists of the members of $D_\bot^1$ plus Boolean operators $-_\bot^2, \sqcap_\bot^2$, and $\sqcup_\bot^2$, connecting those members.
- $D_\bot^m$, generally, consists of the members of $D_\bot^{m-1}$ plus Boolean operators $-_\bot^m, \sqcap_\bot^m$, and $\sqcup_\bot^m$,[1] connecting those members.
- $D_\bot = \bigcup_m D_\bot^m$.

3.1.2 Induction Step

Now that we have defined $-_\bot^m, \sqcap_\bot^m$, and $\sqcup_\bot^m$, we proceed to define $-_A^m, \sqcap_A^m$, and $\sqcup_A^m$ for any type A. We start with $-_{\neg\bot}^m, \sqcap_{\neg\bot}^m$, and $\sqcup_{\neg\bot}^m$ (recall $\neg\bot = \bot \to \bot$). These are defined as follows, where the term 'undefined' is abbreviated as U.

[1] Strictly speaking, $-_\bot^m$ belongs to type $\neg\bot$ $(= \bot \to \bot)$ and $\sqcap_\bot^m$ and $\sqcup_\bot^m$ belong to type $\bot \to \neg\bot$ $(= \bot \to (\bot \to \bot))$, so they should be denoted as $-_{\neg\bot}^m, \sqcap_{\bot\to\neg\bot}^m$, and $\sqcup_{\bot\to\neg\bot}^m$, respectively. But we simplify the notation. A similar simplification will be employed throughout this paper.

- If $n \leq m$, then for any $x^n_\bot, y^{\leq m}_\bot, y^{\leq m}_{1\bot}$, and $y^{\leq m}_{2\bot}$,

$$\begin{array}{rcl}
x^n_\bot \xrightarrow[(-^m_{\neg\bot} f^{\leq m}_{\neg\bot})^m_{\neg\bot}]{} (-^m_\bot y^{\leq m}_\bot)^m_\bot & \Leftrightarrow & x^n_\bot \xrightarrow[f^{\leq m}_{\neg\bot}]{} y^{\leq m}_\bot \\
x^n_\bot \xrightarrow[(-^m_{\neg\bot} f^{\leq m}_{\neg\bot})^m_{\neg\bot}]{} \mathsf{U} & \Leftrightarrow & x^n_\bot \xrightarrow[f^{\leq m}_{\neg\bot}]{} \mathsf{U} \\
x^n_\bot \xrightarrow[(f^{\leq m}_{1\neg\bot} \sqcap^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}]{} (y^{\leq m}_{1\bot} \sqcap^m_\bot y^{\leq m}_{2\bot})^m_\bot & \Leftrightarrow & x^n_\bot \xrightarrow[f^{\leq m}_{1\neg\bot}]{} y^{\leq m}_{1\bot} \text{ and } x^n_\bot \xrightarrow[f^{\leq m}_{2\neg\bot}]{} y^{\leq m}_{2\bot} \\
x^n_\bot \xrightarrow[(f^{\leq m}_{1\neg\bot} \sqcap^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}]{} \mathsf{U} & \Leftrightarrow & x^n_\bot \xrightarrow[f^{\leq m}_{1\neg\bot}]{} \mathsf{U} \text{ or } x^n_\bot \xrightarrow[f^{\leq m}_{2\neg\bot}]{} \mathsf{U} \\
x^n_\bot \xrightarrow[(f^{\leq m}_{1\neg\bot} \sqcup^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}]{} (y^{\leq m}_{1\bot} \sqcup^m_\bot y^{\leq m}_{2\bot})^m_\bot & \Leftrightarrow & x^n_\bot \xrightarrow[f^{\leq m}_{1\neg\bot}]{} y^{\leq m}_{1\bot} \text{ and } x^n_\bot \xrightarrow[f^{\leq m}_{2\neg\bot}]{} y^{\leq m}_{2\bot} \\
x^n_\bot \xrightarrow[(f^{\leq m}_{1\neg\bot} \sqcup^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}]{} y^{\leq m}_{1\bot} & \Leftarrow & x^n_\bot \xrightarrow[f^{\leq m}_{1\neg\bot}]{} y^{\leq m}_{1\bot} \text{ and } x^n_\bot \xrightarrow[f^{\leq m}_{2\neg\bot}]{} \mathsf{U} \\
x^n_\bot \xrightarrow[(f^{\leq m}_{1\neg\bot} \sqcup^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}]{} y^{\leq m}_{2\bot} & \Leftarrow & x^n_\bot \xrightarrow[f^{\leq m}_{1\neg\bot}]{} \mathsf{U} \text{ and } x^n_\bot \xrightarrow[f^{\leq m}_{2\neg\bot}]{} y^{\leq m}_{2\bot} \\
x^n_\bot \xrightarrow[(f^{\leq m}_{1\neg\bot} \sqcup^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}]{} \mathsf{U} & \Leftrightarrow & x^n_\bot \xrightarrow[f^{\leq m}_{1\neg\bot}]{} \mathsf{U} \text{ and } x^n_\bot \xrightarrow[f^{\leq m}_{2\neg\bot}]{} \mathsf{U}
\end{array}$$

- If $m < n$, then for any $g^m_{\neg\bot}$, including $(-^m_{\neg\bot} f^{\leq m}_{\neg\bot})^m_{\neg\bot}, (f^{\leq m}_{1\neg\bot} \sqcap^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}$, and $(f^{\leq m}_{1\neg\bot} \sqcup^m_{\neg\bot} f^{\leq m}_{2\neg\bot})^m_{\neg\bot}$,

$$\begin{array}{rcl}
(-^n_\bot x^{\leq n}_\bot)^n_\bot \xrightarrow[g^m_{\neg\bot}]{} (-^n_\bot y^{\leq n}_\bot)^n_\bot & \Leftrightarrow & x^n_\bot \xrightarrow[g^m_{\neg\bot}]{} y^n_\bot \\
(-^n_\bot x^{\leq n}_\bot)^n_\bot \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} & \Leftrightarrow & x^{\leq n}_\bot \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} \\
(x^{\leq n}_{1\bot} \sqcap^n_\bot x^{\leq n}_{2\bot})^n_\bot \xrightarrow[g^m_{\neg\bot}]{} (y^{\leq n}_{1\bot} \sqcap^n_\bot y^{\leq n}_{2\bot})^n_\bot & \Leftrightarrow & x^{\leq n}_{1\bot} \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{1\bot} \text{ and } x^{\leq n}_{2\bot} \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{2\bot} \\
(x^{\leq n}_{1\bot} \sqcap^n_\bot x^{\leq n}_{2\bot})^n_\bot \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} & \Leftrightarrow & x^{\leq n}_{1\bot} \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} \text{ or } x^{\leq n}_{2\bot} \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} \\
(x^{\leq n}_{1\bot} \sqcup^n_\bot x^{\leq n}_{2\bot})^n_\bot \xrightarrow[g^m_{\neg\bot}]{} (y^{\leq n}_{1\bot} \sqcup^n_\bot y^{\leq n}_{2\bot})^n_\bot & \Leftrightarrow & x^{\leq n}_{1\bot} \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{1\bot} \text{ and } x^{\leq n}_{2\bot} \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{2\bot} \\
(x^{\leq n}_{1\bot} \sqcup^n_\bot x^{\leq n}_{2\bot})^n_\bot \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{1\bot} & \Leftarrow & x^{\leq n}_{1\bot} \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{1\bot} \text{ and } x^{\leq n}_{2\bot} \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} \\
(x^{\leq n}_{1\bot} \sqcup^n_\bot x^{\leq n}_{2\bot})^n_\bot \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{2\bot} & \Leftarrow & x^{\leq n}_{1\bot} \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} \text{ and } x^{\leq n}_{2\bot} \xrightarrow[g^m_{\neg\bot}]{} y^{\leq n}_{2\bot} \\
(x^{\leq n}_{1\bot} \sqcup^n_\bot x^{\leq n}_{2\bot})^n_\bot \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} & \Leftrightarrow & x^{\leq n}_{1\bot} \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U} \text{ and } x^{\leq n}_{2\bot} \xrightarrow[g^m_{\neg\bot}]{} \mathsf{U}
\end{array}$$

Now that $-^m_{\neg\bot}, \sqcap^m_{\neg\bot}$, and $\sqcup^m_{\neg\bot}$ have been defined, we now define $-^m_{A\to B}, \sqcap^m_{A\to B}$, and $\sqcup^m_{A\to B}$ for any type $A \to B$, assuming that $-^m_A, \sqcap^m_A, \sqcup^m_A, -^m_B, \sqcap^m_B$, and $\sqcup^m_B$ have been defined.

- If $n \leq m$, then for any $x^n_A, y^{\leq m}_B, y^{\leq m}_{1B}$, and $y^{\leq m}_{2B}$,

$$\begin{array}{rcl}
x^n_A \xrightarrow[(-^m_{A\to B} f^{\leq m}_{A\to B})^m_{A\to B}]{} (-^m_B y^{\leq m}_B)^m_B & \Leftrightarrow & x^n_A \xrightarrow[f^m_{A\to B}]{} y^{\leq m}_B \\
x^n_A \xrightarrow[(-^m_{A\to B} f^{\leq m}_{A\to B})^m_{A\to B}]{} \mathsf{U} & \Leftrightarrow & x^n_A \xrightarrow[f^{\leq m}_{A\to B}]{} \mathsf{U} \\
x^n_A \xrightarrow[(f^{\leq m}_{1A\to B} \sqcap^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}]{} (y^{\leq m}_{1B} \sqcap^m_B y^{\leq m}_{2B})^m_B & \Leftrightarrow & x^n_A \xrightarrow[f^{\leq m}_{1A\to B}]{} y^{\leq m}_{1B} \text{ and } x^n_A \xrightarrow[f^{\leq m}_{2A\to B}]{} y^{\leq m}_{2B} \\
x^n_A \xrightarrow[(f^{\leq m}_{1A\to B} \sqcap^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}]{} \mathsf{U} & \Leftrightarrow & x^n_A \xrightarrow[f^{\leq m}_{1A\to B}]{} \mathsf{U} \text{ or } x^n_A \xrightarrow[f^{\leq m}_{2A\to B}]{} \mathsf{U} \\
x^n_A \xrightarrow[(f^{\leq m}_{1A\to B} \sqcup^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}]{} (y^{\leq m}_{1B} \sqcup^m_B y^{\leq m}_{2B})^m_B & \Leftrightarrow & x^n_A \xrightarrow[f^{\leq m}_{1A\to B}]{} y^{\leq m}_{1B} \text{ and } x^n_A \xrightarrow[f^{\leq m}_{2A\to B}]{} y^{\leq m}_{2B} \\
x^n_A \xrightarrow[(f^{\leq m}_{1A\to B} \sqcup^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}]{} y^{\leq m}_{1B} & \Leftarrow & x^n_A \xrightarrow[f^{\leq m}_{1A\to B}]{} y^{\leq m}_{1B} \text{ and } x^n_A \xrightarrow[f^{\leq m}_{2A\to B}]{} \mathsf{U} \\
x^n_A \xrightarrow[(f^{\leq m}_{1A\to B} \sqcup^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}]{} y^{\leq m}_{2B} & \Leftarrow & x^n_A \xrightarrow[f^{\leq m}_{1A\to B}]{} \mathsf{U} \text{ and } x^n_A \xrightarrow[f^{\leq m}_{2A\to B}]{} y^{\leq m}_{2B} \\
x^n_A \xrightarrow[(f^{\leq m}_{1A\to B} \sqcup^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}]{} \mathsf{U} & \Leftrightarrow & x^n_A \xrightarrow[f^{\leq m}_{1A\to B}]{} \mathsf{U} \text{ and } x^n_A \xrightarrow[f^{\leq m}_{2A\to B}]{} \mathsf{U}
\end{array}$$

- If $m < n$, then for any $g^m_{A\to B}$, including $(-^m_{A\to B} f^{\leq m}_{A\to B})^m_{A\to B}$, $(f^{\leq m}_{1A\to B} \sqcap^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}$, and $(f^{\leq m}_{1A\to B} \sqcup^m_{A\to B} f^{\leq m}_{2A\to B})^m_{A\to B}$,

$$
\begin{array}{rcl}
(-^n_A x^{\leq n}_A)^n_A \xrightarrow[g^m_{A\to B}]{} (-^n_B y^{\leq n}_B)^n_B & \Leftrightarrow & x^{\leq n}_A \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_B \\
(-^n_A x^{\leq n}_A)^n_A \xrightarrow[g^m_{A\to B}]{} \mathsf{U} & \Leftrightarrow & x^{\leq n}_A \xrightarrow[g^m_{A\to B}]{} \mathsf{U} \\
(x^{\leq n}_{1A} \sqcap^n_A x^{\leq n}_{2A})^n_A \xrightarrow[g^m_{A\to B}]{} (y^{\leq n}_{1B} \sqcap^n_B y^{\leq n}_{2B})^n_B & \Leftrightarrow & x^{\leq n}_{1A} \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{1B} \text{ and } x^{\leq n}_{2A} \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{2B} \\
(x^{\leq n}_{1A} \sqcap^n_A x^{\leq n}_{2A})^n_A \xrightarrow[g^m_{A\to B}]{} \mathsf{U} & \Leftrightarrow & x^{\leq n}_{1A} \xrightarrow[g^m_{A\to B}]{} \mathsf{U} \text{ or } x^{\leq n}_{2A} \xrightarrow[g^m_{A\to B}]{} \mathsf{U} \\
(x^{\leq n}_{1A} \sqcup^n_A x^{\leq n}_{2A})^n_A \xrightarrow[g^m_{A\to B}]{} (y^{\leq n}_{1B} \sqcup^n_B y^{\leq n}_{2B})^n_B & \Leftrightarrow & x^{\leq n}_{1A} \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{1B} \text{ and } x^{\leq n}_{2A} \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{2B} \\
(x^{\leq n}_{1A} \sqcup^n_A x^{\leq n}_{2A})^n_A \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{1B} & \Leftarrow & x^{\leq n}_{1A} \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{1B} \text{ and } x^{\leq n}_{2A} \xrightarrow[g^m_{A\to B}]{} \mathsf{U} \\
(x^{\leq n}_{1A} \sqcup^n_A x^{\leq n}_{2A})^n_A \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{2B} & \Leftarrow & x^{\leq n}_{1A} \xrightarrow[g^m_{A\to B}]{} \mathsf{U} \text{ and } x^{\leq n}_{2A} \xrightarrow[g^m_{A\to B}]{} y^{\leq n}_{2B} \\
(x^{\leq n}_{1A} \sqcup^n_A x^{\leq n}_{2A})^n_A \xrightarrow[g^m_{A\to B}]{} \mathsf{U} & \Leftrightarrow & x^{\leq n}_{1A} \xrightarrow[g^m_{A\to B}]{} \mathsf{U} \text{ and } x^{\leq n}_{2A} \xrightarrow[g^m_{A\to B}]{} \mathsf{U}
\end{array}
$$

Since all function types of form $A \to B$ are built up from types A and B, $-^m_A, \sqcap^m_A$, and $\sqcup^m_A$ have now been defined for any type A.

3.1.3 Summary

As a consequence of the above, so long as all the relevant functions are total, the following equations hold. (As is usually the case in the literature, the function application '$f(x)$' is simplified as 'fx' throughout this paper, omitting the parentheses.)

- If $n \leq m$, then for any $h, k \leq m$,

$$
\begin{array}{rcl}
((-^m_{A\to B} f^h_{A\to B})^m_{A\to B} x^n_A)^m_B & = & (-^m_B (f^h_{A\to B} x^n_A)^{\max(h,n)}_B)^m_B \\
((f^h_{1A\to B} \sqcap^m_{A\to B} f^k_{2A\to B})^m_{A\to B} x^n_A)^m_B & = & ((f^h_{1A\to B} x^n_A)^{\max(h,n)}_B \sqcap^m_B (f^m_{2A\to B} x^n_A)^{\max(k,n)}_B)^m_B \\
((f^h_{1A\to B} \sqcup^m_{A\to B} f^k_{2A\to B})^m_{A\to B} x^n_A)^m_B & = & ((f^h_{1A\to B} x^n_A)^{\max(h,n)}_B \sqcup^m_B (f^k_{2A\to B} x^n_A)^{\max(k,n)}_B)^m_B
\end{array}
$$

- If $m < n$, then for any $h, k \leq n$,

$$
\begin{array}{rcl}
(g^m_{A\to B}(-^n_A x^h_A)^n_A)^n_B & = & (-^n_B (g^m_{A\to B} x^h_A)^{\max(m,h)}_B)^n_B \\
(g^m_{A\to B}(x^h_{1A} \sqcap^n_A x^k_{2A})^n_A)^n_B & = & ((g^m_{A\to B} x^h_{1A})^{\max(m,h)}_B \sqcap^n_B (g^m_{A\to B} x^k_{2A})^{\max(m,k)}_B)^n_B \\
(g^m_{A\to B}(x^h_{1A} \sqcup^n_A x^k_{2A})^n_A)^n_B & = & ((g^m_{A\to B} x^h_{1A})^{\max(m,h)}_B \sqcup^n_B (g^m_{A\to B} x^k_{2A})^{\max(m,k)}_B)^n_B
\end{array}
$$

Put simply, when function $f^m_{A\to B}$ involving the main operator of rank m is applied to object x^n_A involving the main operator of rank n, then (so long as the result is defined) the former distributes over the latter if m is larger than or equal to n; the latter distributes over the former if n is larger than m.

In fact, we can treat 'undefined' U as if it were a special value (not the absence of one) and declare that the above equations hold universally (so, e.g., one side is U iff the other side is U, etc.), instead of making disclaimers about undefined cases. That's in fact the convention we adopt in what follows, often omitting disclaimers.

3.2 Theorems and Examples

3.2.1 Theorems

Theorem 1. *The total functions in each D^n_A form a Boolean algebra.*

Proof Routine. □

Definition 1. For any type A and rank n,

$$\mathsf{T}^n_A =_{df} (p^{\leq n}_A \sqcup^n_A (-^n_A p^{\leq n}_A)^n_A)^n_A \quad \textit{and} \quad \mathsf{F}^n_A =_{df} (p^{\leq n}_A \sqcap^n_A (-^n_A p^{\leq n}_A)^n_A)^n_A.$$

Theorem 2. *For any $f^m_{A\to B}$ and x^n_A, assuming that $(f^m_{A\to B}x^n_A)^{\max(m,n)}_B$ is defined,*

1. *If $n \leq m$,*

$$(f^m_{A\to B}x^n_A)^m_B = \mathsf{T}^m_B \;\Leftrightarrow\; f^m_{A\to B} = \mathsf{T}^m_{A\to B} \quad \textit{and} \quad (f^m_{A\to B}x^n_A)^m_B = \mathsf{F}^m_B \;\Leftrightarrow\; f^m_{A\to B} = \mathsf{F}^m_{A\to B}.$$

2. *If $m < n$,*

$$(f^m_{A\to B}x^n_A)^n_B = \mathsf{T}^n_B \;\Leftrightarrow\; x^n_A = \mathsf{T}^n_A \quad \textit{and} \quad (f^m_{A\to B}x^n_A)^n_B = \mathsf{F}^n_B \;\Leftrightarrow\; x^n_A = \mathsf{F}^n_A.$$

Proof From the results of Subsection 3.1.3 and Definition 1. □

3.2.2 Examples

To give the reader a better sense of how the operators distribute depending on their ranks, we here present a few examples.

Example 1.

1. (a) $((-^n_{A\to B}f^{n-1}_{A\to B})^n_{A\to B}(x^n_{1A}\sqcap^n_A x^n_{2A})^n_A)^n_B$
$= (-^n_B(f^{n-1}_{A\to B}(x^n_{1A}\sqcap^n_A x^n_{2A})^n_A)^n_B)^n_B$
$= (-^n_B((f^{n-1}_{A\to B}x^n_{1A})^n_B \sqcap^n_B (f^{n-1}_{A\to B}x^n_{2A})^n_B)^n_B)^n_B.$

   (b) $((-^n_{A\to B}f^{n-1}_{A\to B})^n_{A\to B}(x^n_{1A}\sqcap^{n+1}_A x^n_{2A})^{n+1}_A)^{n+1}_B$
$= ((-^n_{A\to B}f^{n-1}_{A\to B})^n_{A\to B}x^n_{1A})^n_B \sqcap^{n+1}_B ((-^n_{A\to B}f^{n-1}_{A\to B})^n_{A\to B}x^n_{2A})^n_B)^{n+1}_B$
$= ((-^n_B(f^{n-1}_{A\to B}x^n_{1A})^n_B)^n_B \sqcap^{n+1}_B (-^n_B(f^{n-1}_{A\to B}x^n_{2A})^n_B)^n_B)^{n+1}_B.$

2. (a) $([(f^0_{A\to\neg B}\sqcap^1_{A\to\neg B}g^0_{A\to\neg B})^1_{A\to\neg B}(-^1_A a^0_A)^1_A]^1_{\neg B}\ [x^0_B\sqcup^2_B y^0_B]^2_B)^2_\perp$
$= ([(-^1_\perp((f^0_{A\to\neg B}a^0_A)^0_{\neg B}x^0_B)^0_\perp)^1_\perp\sqcap^1_\perp(-^1_\perp((g^0_{A\to\neg B}a^0_A)^0_{\neg B}x^0_B)^0_\perp)^1_\perp]^1_\perp \sqcup^2_\perp [(-^1_\perp((f^0_{A\to\neg B}a^0_A)^0_{\neg B}y^0_B)^0_\perp)^1_\perp$
$\sqcap^1_\perp(-^1_\perp((g^0_{A\to\neg B}a^0_A)^0_{\neg B}y^0_B)^0_\perp)^1_\perp]^1_\perp)^2_\perp.$

   (b) $([f^0_{\neg B}\sqcap^1_{\neg B}g^0_{\neg B}]^1_{\neg B}\ [(-^1_{A\to B}a^0_{A\to B})^1_{A\to B}(x^0_A\sqcup^2_A y^0_A)^2_A]^2_B)^2_\perp$
$= ([(-^1_\perp(f^0_{\neg B}(a^0_{A\to B}x^0_A)^0_B)^0_\perp)^1_\perp\sqcap^1_\perp(-^1_\perp(g^0_{\neg B}(a^0_{A\to B}x^0_A)^0_B)^0_\perp)^1_\perp]^1_\perp \sqcup^2_\perp [(-^1_\perp(f^0_{\neg B}(a^0_{A\to B}y^0_A)^0_B)^0_\perp)^1_\perp$
$\sqcap^1_\perp(-^1_\perp(g^0_{\neg B}(a^0_{A\to B}y^0_A)^0_B)^0_\perp)^1_\perp]^1_\perp)^2_\perp.$

   (c) $([f^0_{\neg B}\sqcap^2_{\neg B}g^0_{\neg B}]^1_{\neg B}\ [(-^1_{A\to B}a^0_{A\to B})^1_{A\to B}(x^0_A\sqcup^2_A y^0_A)^2_A]^2_B)^2_\perp$
$= ([(-^1_\perp(f^0_{\neg B}(a^0_{A\to B}x^0_A)^0_B)^0_\perp)^1_\perp\sqcup^2_\perp(-^1_\perp(f^0_{\neg B}(a^0_{A\to B}y^0_A)^0_B)^0_\perp)^1_\perp]^2_\perp \sqcap^2_\perp [(-^1_\perp(g^0_{\neg B}(a^0_{A\to B}x^0_A)^0_B)^0_\perp)^1_\perp$
$\sqcup^2_\perp(-^1_\perp(g^0_{\neg B}(a^0_{A\to B}y^0_A)^0_B)^0_\perp)^1_\perp]^2_\perp)^2_\perp.$

3. (a) $((-^n_{A\to B}f^n_{A\to B})^n_{A\to B}\mathsf{F}^{n+1}_A)^{n+1}_B = \mathsf{F}^{n+1}_B.$

   (b) $((-^n_{A\to B}f^n_{A\to B})^n_{A\to B}\mathsf{F}^n_A)^n_B = ((-^n_B(f^n_{A\to B}\mathsf{F}^n_A)^n_B)^n_B.$

4. (a) $(\mathsf{F}^n_{A\to B}(-^n_A x^n_A)^n_A)^n_B = \mathsf{F}^n_B$.
   (b) $(\mathsf{F}^n_{A\to B}(-^{n+1}_A x^n_A)^{n+1}_A)^{n+1}_B = (-^{n+1}_B(\mathsf{F}^n_{A\to B}x^n_A)^n_B)^{n+1}_B = (-^{n+1}_B\mathsf{F}^n_B)^{n+1}_B$.

In 2, the main structures are indicated as $[\cdot]$. It is a good specific example which illustrates the general features of conversions between objects in the domain. Any form of objects can ultimately be turned into objects in 'normal form', i.e., objects of rank 0 applied to one another constituting 'atoms', connected by the operators, $-, \sqcap$, and $\sqcup$. The structure of the original objects is retained in the structure of the atoms. The overall structures of the resulting objects, however, is determined not by the structures of the original objects but purely by the ranks of the operators. Specifically, higher-ranks always dominate and distribute, determining the overall structures. (Note, however, that in cases like $((f^n_{A\to B}\sqcap^n_{A\to B} g^n_{A\to B})^n_{A\to B}(x^n_A\sqcup^n_A y^n_A)^n_A)^n_B$, where the ranks are tied at n, the left-hand (functor) side always prevails and distributes over the right-hand (argument) side, i.e., $= ((f^n_{A\to B}(x^n_A\sqcup^n_A y^n_A)^n_A)^n_B\sqcap^n_B(g^n_{A\to B}(x^n_A\sqcup^n_A y^n_A)^n_A)^n_B)^n_B$.) Consequently, as in 3 and 4 above as well as Theorem 2, Ts and Fs prevail only when they are at the highest rank. This will later prove to be a very important feature.

3.3 Type Reduction

Again, we need the infinitely nested Boolean structure as the domain structure because double-negation elimination, a move from $\neg\neg A$ to A, is tantamount to type reduction from the propositions (or formulas)-as-types point of view. This move is interpreted as a move from $D^n_{\neg\neg A}$ for some n to D^{n+1}_A, i.e., a move from type $\neg\neg A$, rank n, to type A, rank $n+1$.

Let's consider relations between D^0_A, $D^0_{\neg A}$, $D^0_{\neg\neg A}$, and D^1_A for any type A. Any function $f^0_{\neg A}$ will project some members of D^0_A, say x^0_As, into $\mathsf{T}^0_\perp$, and some other members, say y^0_As, into $\mathsf{F}^0_\perp$, while the values of yet some other members, say z^0_As, may be undefined. That is,

$$\begin{array}{rcl} x^0_A & \longrightarrow & \mathsf{T}^0_\perp \\ y^0_A & \longrightarrow & \mathsf{F}^0_\perp \\ z^0_A & \underset{f^0_{\neg A}}{\longrightarrow} & \mathsf{U} \end{array}$$

where U, recall, is short for 'undefined'. Then define:

$$\{f^0_{\neg A}\}^{\leq 1}_A \ =_{df}\ ((\textstyle\bigsqcap^1_A x^0_A)^1_A\sqcap^1_A(\bigsqcap^1_A(-^1_A y^0_A)^1_A)^1_A)^1_A.$$

Similarly, any function $F^0_{\neg\neg A}$ will project some members of $D^0_{\neg A}$, say $f'^0_{\neg A}$s, into $\mathsf{T}^0_\perp$, and some other members, say $g'^0_{\neg A}$s, into $\mathsf{F}^0_\perp$, while the values of yet some other members, say $h'^0_{\neg A}$s, may be undefined. That is,

$$\begin{array}{rcl} f'^0_{\neg A} & \longrightarrow & \mathsf{T}^0_\perp \\ g'^0_{\neg A} & \longrightarrow & \mathsf{F}^0_\perp \\ h'^0_{\neg A} & \underset{F^0_{\neg\neg A}}{\longrightarrow} & \mathsf{U} \end{array}$$

Then define:

$$\{F^0_{\neg\neg A}\}^{\leq 1}_A \ =_{df}\ (\textstyle\bigsqcap^1_A(-^1_A\{g'^0_{\neg A}\}^{\leq 1}_A)^1_A)^1_A.$$

Example 2. Suppose, for the sake of simplicity, that D^0_A has only four members, a^0_A, b^0_A, c^0_A, and d^0_A. Suppose $F^0_{\neg\neg A}$ projects all and only the following three functions, $g^0_{1\neg A}, g^0_{2\neg A}$, and $g^0_{3\neg A}$, into $\mathsf{F}^0_\perp$:

$$\begin{array}{rcl} a^0_A, b^0_A & \longrightarrow & \mathsf{T}^0_\perp \\ c^0_A, d^0_A & \longrightarrow & \mathsf{F}^0_\perp \\ \textit{Nothing} & \underset{g^0_{1\neg A}}{\longrightarrow} & \mathsf{U} \end{array} \qquad \begin{array}{rcl} \textit{Nothing} & \longrightarrow & \mathsf{T}^0_\perp \\ a^0_A, b^0_A & \longrightarrow & \mathsf{F}^0_\perp \\ c^0_A, d^0_A & \underset{g^0_{2\neg A}}{\longrightarrow} & \mathsf{U} \end{array} \qquad \begin{array}{rcl} \textit{Nothing} & \longrightarrow & \mathsf{T}^0_\perp \\ b^0_A & \longrightarrow & \mathsf{F}^0_\perp \\ a^0_A, c^0_A, d^0_A & \underset{g^0_{3\neg A}}{\longrightarrow} & \mathsf{U} \end{array}$$

Then

$$\{F^0_{\neg\neg A}\}^{\leq 1}_A = -(a \sqcap b \sqcap -c \sqcap -d) \sqcap -(-a \sqcap -b) \sqcap -(-b)$$

where $\cdot^0_A$ and $\cdot^1_A$ are omitted from the letters and the operators, respectively, for the sake of clarity. Recall that the operators involved are Boolean. Thus, for instance, the second conjunct above is equivalent to $(a \sqcup b)^1_A$, and the third conjunct is equivalent to b^0_A.

Definition 2. *The type reduction function* for type $\neg\neg A$ at rank 0, $R{\downarrow}^0_{\neg\neg A \to A}$, is defined as

$$F^0_{\neg\neg A} \xrightarrow[R{\Downarrow}^0_{\neg\neg A \to A}]{} \{F^0_{\neg\neg A}\}^{\leq 1}_A$$

for any $F^0_{\neg\neg A}$.

Note that, unlike in most other cases, the superscript to the reduction function $R{\downarrow}$ does not indicate the function's own rank; instead it indicates the rank of the *reductee*, i.e., the object that is the origin of the reduction.

It is obvious, furthermore, that $R{\downarrow}^0_{\neg\neg A \to A}$ can easily be expanded into $R{\downarrow}^n_{\neg\neg A \to A}$ for any n and generalized into $R{\downarrow}_{\neg\neg A \to A}$ because both $D^n_{\neg\neg A}$ and D^n_A are just Boolean expansions of $D^{n-1}_{\neg\neg A}$ and D^{n-1}_A, respectively. Thus, we can obtain the general type reduction function $R{\downarrow}_{\neg\neg A \to A}$. Type reduction may also be called *type-lowering* (as in linguistics).

Theorem 3. *1. $R{\downarrow}_{\neg\neg A \to A}$ is an isomorphism.*

2. (Type Reduction) For any functions $F^n_{\neg\neg A}$ and $f^n_{\neg A}$,

$$(F^n_{\neg\neg A} f^n_{\neg A})^n_\perp = \mathsf{F}^n_\perp \;\Leftrightarrow\; (f^n_{\neg A}\{F^n_{\neg\neg A}\}^{\leq n+1}_A)^{\leq n+1}_\perp = \mathsf{F}^{n+1}_\perp.$$

3. For any A and n,

$$\{\mathsf{F}^n_{\neg\neg A}\}^{\leq n+1}_A = \mathsf{F}^{\leq n+1}_A.$$

Proof 1. Routine. $R{\downarrow}^0_{\neg\neg A \to A}$ is an isomorphism from $D^0_{\neg\neg A}$ to D^1_A, so is $R{\downarrow}^n_{\neg\neg A \to A}$ from $D^n_{\neg\neg A}$ to D^{n+1}_A for any $n > 0$. 2. From 1. 3. Routine. Both sides can be $\mathsf{T}^n_\perp, \mathsf{F}^n_\perp$, or undefined. □

The picture that emerges from all these is this:

$$\begin{array}{ccccccccccccccccccc}
\vdots & & & & & & & & & & & & & & & & & & \\
D_{\neg 5A} & \supseteq & \cdots & \supset & D^0_{\neg 5A} & & & & & & & & & & & & & & \vdots \\
 & & & & \Downarrow R{\downarrow} & & & & & & & & & & & & D^0_{\neg 4A} & \subset \cdots \subseteq & D_{\neg 4A} \\
D_{\neg 3A} & \supseteq & \cdots & \supset & D^1_{\neg 3A} & \supset & D^0_{\neg 3A} & & & & & & & & & & \Downarrow R{\downarrow} & & \\
 & & & & \Downarrow R{\downarrow} & & \Downarrow R{\downarrow} & & & & & & & & D^0_{\neg\neg A} & \subset & D^1_{\neg\neg A} & \subset \cdots \subseteq & D_{\neg\neg A} \\
D_{\neg A} & \supseteq & \cdots & \supset & D^2_{\neg A} & \supset & D^1_{\neg A} & \supset & D^0_{\neg A} & & & & & & \Downarrow R{\downarrow} & & \Downarrow R{\downarrow} & & \\
 & & & & & & & & & & D^0_A & \subset & D^1_A & \subset & D^2_A & \subset \cdots \subseteq & & & D_A
\end{array}$$

Fig. 1: Infinitely nested Boolean structure and type reduction

Here '¬3' means '¬¬¬', etc. The type and rank of each $R{\downarrow}$ are omitted but should be obvious. Since each R is one-to-one, its converse function exists. Also, the compound functions along the lines of ⇓ above will prove to be useful.

Definition 3. For any type A, any m and n,

1. $R{\downarrow}^n_{\neg 2mA \to A} =_{df} R{\downarrow}^{n+m}_{\neg\neg A \to A} \circ \cdots \circ R{\downarrow}^{n+1}_{\neg 2(m-1)A \to \neg 2(m-2)A} \circ R{\downarrow}^n_{\neg 2mA \to \neg 2(m-1)A}$.

2. $R{\uparrow}^{n}_{\neg 2mA\to A}$ is the converse function of $R{\downarrow}^{n}_{\neg 2mA\to A}$.

For instance, as you can see in Fig. 1 above, $R{\downarrow}^{0}_{\neg 4A\to A} = R{\downarrow}^{1}_{\neg\neg A\to A} \circ R{\downarrow}^{0}_{\neg 4A\to\neg\neg A}$ and $R{\uparrow}^{0}_{\neg 4A\to A}$ is its converse. $R{\uparrow}$ may be called a *type-lifting* function. The objects connected to an object by $R{\downarrow}$ (and, thus, $R{\uparrow}$ as well) may be called the object's *clones*.

The following will prove to be significant in the semantics of $LK2$ and $NK2$.

Lemma 1. *Define* $f^0_{\neg A}$ *thus:*

$$\begin{array}{rcl} \textit{Nothing} & \longrightarrow & \mathsf{T}^0_\perp \\ e^0_A & \longrightarrow & \mathsf{F}^0_\perp \\ \textit{Everything else} & \underset{f^0_{\neg A}}{\longrightarrow} & \mathsf{U} \end{array}$$

where e^0_A *is one particular object. Accordingly,* $f^0_{\neg A}$ *is one particular (partial) function. Then define* $F^0_{\neg\neg A}$ *thus:*

$$\begin{array}{rcl} \textit{Nothing} & \longrightarrow & \mathsf{T}^0_\perp \\ f^0_{\neg A} & \longrightarrow & \mathsf{F}^0_\perp \\ \textit{Everything else} & \underset{F^0_{\neg\neg A}}{\longrightarrow} & \mathsf{U} \end{array}$$

Then $\{F^0_{\neg\neg A}\}^{\leq 1}_A = e^0_A$.

Proof From the definition of $\{F^0_{\neg\neg A}\}^{\leq 1}_A$. □

Theorem 4. *In the above case,*

$$F^0_{\neg\neg A} = \mathsf{F}^0_{\neg\neg A} \quad \Leftrightarrow \quad e^0_A = \mathsf{F}^0_A.$$

Proof From Theorem 3.3 and Lemma 1. □

Again, this generalizes to rank n in a straightforward fashion.

4 Semantics of λK, $LK2$, and $NK2$

We are now in a position to consider the semantics of all the three calculi, λK, $LK2$, and $NK2$. In each case, we shall give interpretations to the terms, define the notion of semantic consequence $\vDash$, and then prove the (soundness and) completeness of the calculus with respect to the notion.

Here the following comparison to the usual simple denotational semantics of classical propositional logic should help understand the basic idea of our semantics. In the usual semantics of classical propositional logic, the key notion is *truth*: the deductive rules are all and only rules that are *truth-preserving*. We modify this idea in two ways. First, since we use the general Boolean algebras and not the Boolean algebra involving only two values, we replace the notion of truth with that of *non-falsity*, i.e., any Boolean value except the bottom. Second, since we have a Boolean structure in each rank, values are relativized with ranks: non-falsity will be non-falsity *at some rank n*. With these two modifications in mind, we construct semantics along the lines of the usual semantics of classical propositional logic and show the completeness of the three calculi with respect them.

In all the semantics, the difference of interpretations consists solely in the difference of valuations (or assignments), ρs, to the variables; everything else is fixed. (The semantics are similar to the usual semantics of classical propositional logic in this respect.)

We give the same interpretation to the types involved in all the calculi.

Definition 4. 1. For any atomic type X, $[\![X]\!]_\rho = [\![\bot]\!]_\rho$ = absurdity type $\bot$ (for any ρ).

2. $[\![A \to B]\!]_\rho$ = the type of total or partial functions from D_A to D_B (for any ρ).

4.1 Semantics of λK

4.1.1 Interpretation of λK

The interpretation $[\![\cdot]\!]_{\rho[x \mapsto d]}$ is the same as the interpretation $[\![\cdot]\!]_\rho$ except that object d is assigned to variable x.

Definition 5. 1. $[\![x_A]\!]_\rho = \rho(x) \in D_A$.

2. If $[\![M_{A\to B}]\!]_\rho = f^m_{A\to B}$ and $[\![N_A]\!]_\rho = e^n_A$, then $[\![(M_{A\to B}N_A)_B]\!]_\rho = (f^m_{A\to B}e^n_A)^{\max(m,n)}_B$.

3. $[\![(\lambda x_A M_B)_{A\to B}]\!]_\rho = f^m_{A\to B}$ such that for any d^n_A, $(f^m_{A\to B}d^n_A)^{\max(m,n)}_B = [\![M_B]\!]_{\rho[x_A \mapsto d^n_A]}$.

4. If $[\![M_{\neg\neg A}]\!]_\rho = f^m_{\neg\neg A}$, then $[\![(\downarrow_{\neg\neg A\to A} M_{\neg\neg A})_A]\!]_\rho = (R{\downarrow}^m_{\neg\neg A\to A} f^m_{\neg\neg A})^{m+1}_A$.

Lemma 2. *Suppose* $[\![M_{A\to B}]\!]_\rho = f^m_{A\to B}$ *and* $[\![N_A]\!]_\rho = e^n_A$. *Then*

1. *If* $n \le m$,
$$[\![(M_{A\to B}N_A)_B]\!]_\rho = \mathsf{F}^m_B \quad \Leftrightarrow \quad [\![M_{A\to B}]\!]_\rho = \mathsf{F}^m_{A\to B}.$$

2. *If* $m < n$,
$$[\![(M_{A\to B}N_A)_B]\!]_\rho = \mathsf{F}^n_B \quad \Leftrightarrow \quad [\![N_A]\!]_\rho = \mathsf{F}^n_A.$$

Proof From Theorem 2 and Definition 5.2. □

Theorem 5. *1. For any* $M_{A\to B}$, N_A, m, *and* n, *for any* ρ,

$$[\![(M_{A\to B}N_A)_B]\!]_\rho = \mathsf{F}^{\max(m,n)}_B \quad \Leftrightarrow \quad [\![M_{A\to B}]\!]_\rho = \mathsf{F}^m_{A\to B} \ \&\ n \le m \quad OR \quad [\![N_A]\!]_\rho = \mathsf{F}^n_A \ \&\ m < n.$$

2. For any x_A, M_B, *and* N_A, *for any* ρ,

$$[\![((\lambda x_A M_B)_{A\to B}N_A)_B]\!]_\rho = [\![(M[x_A := N_A])_B]\!]_\rho.$$

3. For any $M_{\neg\neg A}$ *and* n, *for any* ρ,

$$[\![M_{\neg\neg A}]\!]_\rho = \mathsf{F}^n_{\neg\neg A} \quad \Leftrightarrow \quad [\![(\downarrow M_{\neg\neg A})_A]\!]_\rho = \mathsf{F}^{n+1}_A.$$

4. For any $M_{\neg\neg A}$ *and* $N_{\neg A}$, *for any* ρ,

$$[\![(N_{\neg A}(\downarrow M_{\neg\neg A})_A)_\bot]\!]_\rho = [\![(M_{\neg\neg A}N_{\neg A})_\bot]\!]_\rho.$$

Proof 1. From Lemma 2. 2. From Definition 5.3. 3. From Theorem 3.3 and Definition 5.4. 4. From Theorem 3.2 and Definition 5.4. □

4.1.2 Semantic Consequence in λK

Definition 6. For any set Γ of terms and any term N_B,

$$\Gamma \vDash_{\lambda K} N_B \quad \Leftrightarrow \quad \forall \rho(\forall M_A \in \Gamma.\exists_1 m.[\![M_A]\!]_\rho = f^m_A \neq \mathsf{F}^m_A \Rightarrow \exists_1 n.[\![N_B]\!]_\rho = f^n_B \neq \mathsf{F}^n_B)$$

where '$\exists_1$' reads 'there exists exactly one'. Here the basic notion of semantic consequence is that of *(necessary) non-falsity preservation*: (necessarily) if all premises are non-false (including undefined), then the conclusion ought to be non-false, too.

As a special case,

$$\vDash_{\lambda K} N_B \quad \Leftrightarrow \quad \forall \rho.\exists_1 n.[\![N_B]\!]_\rho = f^n_B \neq \mathsf{F}^n_B.$$

4.1.3 Completeness of λK

Theorem 5 shows that all the deduction rules of λK except (λI) are non-falsity preserving in both directions. λ-introduction (λI), or the so-called '(λ-)abstraction', is exceptional because it contains a subproof. So we (only) need to show for the completeness of λK that (λI) is also non-falsity preserving in both directions in the relevant sense, i.e., that the subproof is non-falsity preserving iff the conclusion is necessarily non-false.

Definition 7. Suppose $[\![M_A]\!]_\rho = e^n_A$. Then *the rank* of term M_A in valuation ρ,

$$RK([\![M_A]\!]_\rho) = n.$$

Definition 8. *The top variable* of term M_A in valuation ρ, $TV([\![M_A]\!]_\rho)$, is defined as follows.

1. $TV([\![x_A]\!]_\rho) = x_A$.
2. Suppose $[\![M_{A\to B}]\!]_\rho = f^m_{A\to B}$ and $[\![N_A]\!]_\rho = e^n_A$. Then
   (a) If $n \leq m$, $TV([\![(M_{A\to B}N_A)_B]\!]_\rho) = TV([\![M_{A\to B}]\!]_\rho)$.
   (b) If $m < n$, $TV([\![(M_{A\to B}N_A)_B]\!]_\rho) = TV([\![N_A]\!]_\rho)$.
3. (a) If $TV([\![M_B]\!]_\rho) \neq x_A$, $TV([\![(\lambda x_A M_B)_{A\to B}]\!]_\rho) = TV([\![M_B]\!]_\rho)$.
   (b) If $TV([\![M_B]\!]_\rho) = x_A$, $TV([\![(\lambda x_A M_B)_{A\to B}]\!]_\rho) = SV([\![M_B]\!]_\rho)$,
   where *the second-to-the-top variable*, SV, is defined in an analogous, obvious manner.
   (Here the idea is to exclude bound variables from rank comparisons.)
   (c) If all variables in M_B are bound by λ, then $TV([\![M_B]\!]_\rho)$ = undefined.
4. $TV([\![(\downarrow M_{\neg\neg A})_A]\!]_\rho) = TV([\![M_{\neg\neg A}]\!]_\rho)$.

The rank of $TV([\![M_A]\!]_\rho)/SV([\![M_A]\!]_\rho)$ is called *the top/second(-to-the-top) rank*.

Lemma 3. *For any term M_A and valuation ρ,*

$$RK([\![M_A]\!]_\rho) = RK(TV([\![M_A]\!]_\rho)).$$

Proof Routine. □

Theorem 6. *For any set Γ of terms, variable x_A, and term M_B,*

$$\Gamma, x_A \vDash_{\lambda K} M_B \quad \Leftrightarrow \quad \Gamma \vDash_{\lambda K} (\lambda x_A M_B)_{A\to B}.$$

Proof In this proof we often omit subscripts if they are obvious or irrelevant.

($\Rightarrow$) Suppose $\Gamma \not\vDash_{\lambda K} (\lambda x_A M_B)_{A\to B}$. Then there is a valuation ρ such that for every $N \in \Gamma$, $[\![N]\!]_\rho = f^m \neq \mathsf{F}^m$ for some m, but $[\![(\lambda x_A M_B)_{A\to B}]\!]_\rho = \mathsf{F}^n$ for some n. So either (1) $TV([\![M_B]\!]_\rho) \neq x_A$ and $[\![M_B]\!]_\rho = \mathsf{F}^n$; or (2) $TV([\![M_B]\!]_\rho) = x_A$ and $SV([\![M_B]\!]_\rho) = \mathsf{F}^n$. But in case (1), there is a variant of ρ, say ρ', which is the same as ρ except that it assigns to x_A a rank n object $\neq \mathsf{F}^n$. So $[\![x_A]\!]_{\rho'} = f^n \neq \mathsf{F}^n$ and $[\![M_B]\!]_{\rho'} = \mathsf{F}^n$. In case (2), there is a variant of ρ, say $\rho*$, which is the same as ρ except that it assigns to x_A a rank m object $\neq \mathsf{F}^m$, where m is a rank lower than the rank of the second-to-the-top variable. So $[\![x_A]\!]_{\rho*} = f^m \neq \mathsf{F}^m$ and $[\![M_B]\!]_{\rho*} = \mathsf{F}^m$. Therefore, either way, $\Gamma, x_A \not\vDash_{\lambda K} M_B$.

($\Leftarrow$) Suppose $\Gamma, x_A \not\vDash_{\lambda K} M_B$. Then there is a valuation ρ such that for every $N \in \Gamma$, $[\![N]\!]_\rho = f^k \neq \mathsf{F}^k$ for some k and $[\![x_A]\!]_\rho = f^m \neq \mathsf{F}^m$ for some m but $[\![M_B]\!]_\rho = \mathsf{F}^n$ for some n. In this case, $TV([\![M_B]\!]_\rho) = \mathsf{F}^n \neq x_A$. So $TV([\![(\lambda x_A M_B)_{A\to B}]\!]_\rho) = \mathsf{F}^n$. Thus, $[\![(\lambda x_A M_B)_{A\to B}]\!]_\rho = \mathsf{F}^n$. Therefore, $\Gamma \not\vDash_{\lambda K} (\lambda x_A M_B)_{A\to B}$. □

In contrast, (λI) is obviously not truth-preserving; for instance, while $x_A \vdash_{\lambda K} x_A$ is truth-preserving, $(\lambda x_A.x_A)_{A\to A}$ is not necessarily true.

Theorem 7. *(Completeness of λK) For any set Γ of terms and any term N_B,*

$$\Gamma \vdash_{\lambda K} N_B \quad \Leftrightarrow \quad \Gamma \vDash_{\lambda K} N_B.$$

Proof Theorem 5.1–4 and Theorem 6 show, respectively, that $(Ap), (\lambda E), (\downarrow I), (\downarrow E)$, and (λI) are non-falsity preserving in the sense of Definition 6 not only from the top to the bottom but in both directions. □

4.2 Semantics of $LK2$

4.2.1 Interpretation of $LK2$

Definition 9. 1. $[\![x_A]\!]_\rho = \rho(x) \in D_A$.

2. $[\![\bot_\bot]\!]_\rho = \mathsf{F}^n_\bot$ for some n.

3. Suppose $[\![M_A]\!]_\rho = d^n_A$ and $[\![N_B]\!]_\rho = e^m_B$. Then

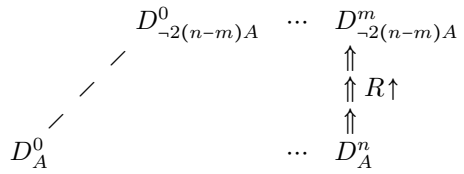
   (a) If $n \leq m$, $[\![(M_A \to N_B)_{A\to B}]\!]_\rho = f^m_{A\to B}$ such that $(f^m_{A\to B} d^n_A)^m_B = e^m_B$.

   (b) If $m < n$, $[\![(M_A \to N_B)_{A\to B}]\!]_\rho = f^m_{\neg 2(n-m)A\to B} \circ R{\uparrow}^m_{A\to\neg 2(n-m)A}$ such that $((f^m_{\neg 2(n-m)A\to B} \circ R{\uparrow}^m_{A\to\neg 2(n-m)A}) d^n_A)^m_B = e^m_B$.

   (c) In both (a) and (b), the relevant function is otherwise undefined unless its values are determined as a consequence of the above definitions. (As an example of a value determined as a consequence of the above definitions, consider if $n \leq m < k$ and $x^k_A = (-^k_A d^n_A)^k_A$; then $(f^m_{A\to B} x^k_A)^k_B = (-^k_B (f^m_{A\to B} d^n_A)^m_B)^k_B = (-^k_B e^m_B)^k_B$.)

Item 3, especially (b), may call for an explanation. Here the basic idea is that $[\![(M_A \to N_B)_{A\to B}]\!]_\rho$ is a (very) partial function that gives the value $[\![N_B]\!]_\rho$ for the argument $[\![M_A]\!]_\rho$ (i.e., $[\![(M_A \to N_B)_{A\to B}]\!]_\rho [\![M_A]\!]_\rho = [\![N_B]\!]_\rho$) as it does in (a), but this idea does not work straightforwardly if $m < n$. In that case, however, we can type-lift $[\![M_A]\!]_\rho = d^n_A$ to its clone at

type $\neg 2(n-m)A$, rank m, and apply the same idea.



Thus, still $[\![(M_A \to N_B)_{A\to B}]\!]_\rho[\![M_A]\!]_\rho = [\![N_B]\!]_\rho$. (a) could be subsumed under the same pattern if the identity function were considered an (extreme or vacuous form of) type-lifting function $R\uparrow$ and everything were a clone of itself, although we won't do so here.

Recall $(\neg M_A)_{\neg A} = (M_A \to \bot_\bot)_{\neg A}$ and $[\![\bot_\bot]\!]_\rho = \mathsf{F}^n_\bot$ for some n.

Lemma 4. *Suppose* $[\![M_A]\!]_\rho = d^n_A$ *and* $[\![(\neg M_A)_{\neg A}]\!]_\rho = f^m_{\neg A}$. *Then*

$$[\![(\neg M_A)_{\neg A}]\!]_\rho[\![M_A]\!]_\rho = \mathsf{F}^{\max(m,n)}_\bot.$$

Proof From Theorem 3.3 and Definition 9. □

Theorem 8. *1. For any* $M_A, m,$ *and* n, *for any* ρ,

$$[\![(\neg M_A)_{\neg A}]\!]_\rho = \mathsf{F}^m_{\neg A} \;\&\; n \le m \quad OR \quad [\![M_A]\!]_\rho = \mathsf{F}^n_A \;\&\; m < n.$$

2. For any M_A *and* n, *for any* ρ,

$$[\![(\neg(\neg M_A)_{\neg A})_{\neg\neg A}]\!]_\rho = \mathsf{F}^n_{\neg\neg A} \;\Leftrightarrow\; [\![M_A]\!]_\rho = \mathsf{F}^n_A.$$

Proof 1. From Theorem 5.1 and Lemma 4. 2. From Theorem 4 and Definition 9.3. □

Needless to say, F is not a partial but a total function.

4.2.2 Semantic Consequence in $LK2$

Definition 10. For any set $\mathbb{S}$ of sequences and any sequence Δ,

$$\mathbb{S} \vDash_{LK2} \Delta \quad\Leftrightarrow\quad \forall\rho.\forall\Gamma \in \mathbb{S}.\exists M_A \in \Gamma.\exists_1 m.[\![M_A]\!]_\rho = \mathsf{F}^m_A \Rightarrow \forall\sigma.\exists N_B \in \Delta.\exists_1 n.[\![N_B]\!]_\sigma = \mathsf{F}^n_B.$$

As a special case,

$$\vDash_{LK2} \Delta \quad\Leftrightarrow\quad \forall\sigma.\exists N_B \in \Delta.\exists_1 n.[\![N_B]\!]_\sigma = \mathsf{F}^n_B.$$

Here the basic notion of semantic consequence is that of *inconsistency preservation*: if all premise sets are inconsistent, then the conclusion set ought to be inconsistent, too, where a set is inconsistent iff at least one of its members must be false.

4.2.3 Completeness of $LK2$

Theorem 9. *(Completeness of LK2) For any set* $\mathbb{S}$ *of sequences and any sequence* Δ,

$$\mathbb{S} \vdash_{LK2} \Delta \quad\Leftrightarrow\quad \mathbb{S} \vDash_{LK2} \Delta.$$

Proof Consider the four rules of $LK2$. The $(\bot)$ and $(\neg)$ rules obviously pose no problem. We only need to consider the $(\to)$ and $(\neg\to)$ rules and show the equivalence between the premise(s) and the conclusion of each rule. In what follows, if '$n <$' appears more than once in one proof, they denote the same m such that $n < m$.

- Show $\forall\rho(\exists k.[\![(\neg M_A)_{\neg A}]\!]_\rho = \mathsf{F}^k_A \;\&\; \exists m.[\![N_B]\!]_\rho = \mathsf{F}^m_B) \;\Rightarrow\; \forall\sigma.\exists n.[\![(M_A \to N_B)_{A\to B}]\!]_\sigma = \mathsf{F}^n_{A\to B}$.

  Suppose for some σ, $[\![(M_A \to N_B)_{A\to B}]\!]_\sigma = f^n_{A\to B} \neq \mathsf{F}^n_{A\to B}$. Then either $[\![M_A]\!]_\sigma = \mathsf{F}^{n<}_A$ or $[\![N_B]\!]_\sigma \neq \mathsf{F}^m_B$ for any m. Then, for some ρ such that $[\![\bot_\bot]\!]_\rho = \mathsf{F}^n_\bot$, $[\![(\neg M_A)_{\neg A}]\!]_\rho = g^k_{\neg A} \neq \mathsf{F}^n_{\neg A}$ for any k or $[\![N_B]\!]_\rho \neq \mathsf{F}^m_B$ for any m. Hence the target proposition by contraposition.

- Show $\forall\sigma.\exists n.[\![(M_A \to N_B)_{A\to B}]\!]_\sigma = \mathsf{F}^n_{A\to B} \;\Rightarrow\; \forall\rho(\exists k.[\![(\neg M_A)_{\neg A}]\!]_\rho = \mathsf{F}^k_A \;\&\; \exists m.[\![N_B]\!]_\rho = \mathsf{F}^m_B)$.

  Suppose for some ρ, (1) $[\![(\neg M_A)_{\neg A}]\!]_\rho = f^k_{\neg A} \neq \mathsf{F}^k_{\neg A}$ or (2) $[\![N_B]\!]_\rho = g^m_B \neq \mathsf{F}^m_B$. (1) Then $[\![M_A]\!]_\rho = \mathsf{F}^{k<}_A$. So, for some σ such that $[\![M_A]\!]_\sigma = \mathsf{F}^{k<}_A$ and $[\![N_B]\!]_\sigma = h^k_B$, $[\![(M_A \to N_B)_{A\to B}]\!]_\sigma = i^k_{A\to B} \neq \mathsf{F}^k_{A\to B}$. (2) Trivially, for some σ such that $[\![M_A]\!]_\sigma = j^m_A$, $[\![(M_A \to N_B)_{A\to B}]\!]_\sigma \neq \mathsf{F}^m_{A\to B}$. Hence, either way, the target proposition by contraposition.

- Show $\forall\rho(\exists k.[\![M_A]\!]_\rho = \mathsf{F}^k_A \text{ or } \exists m.[\![(\neg N_B)_{\neg B}]\!]_\rho = \mathsf{F}^m_B) \;\Rightarrow\; \forall\sigma.\exists n.[\![(\neg(M_A \to N_B)_{A\to B})_{\neg(A\to B)}]\!]_\sigma = \mathsf{F}^n_{\neg(A\to B)}$.

  Suppose for some σ, $[\![(\neg(M_A \to N_B)_{A\to B})_{\neg(A\to B)}]\!]_\sigma = f^n_{\neg(A\to B)} \neq \mathsf{F}^n_{\neg(A\to B)}$. Then $[\![(M_A \to N_B)_{A\to B}]\!]_\sigma = \mathsf{F}^{n<}_{A\to B}$. Then, for some ρ such that $[\![N_B]\!]_\rho = \mathsf{F}^{n<}_B$ and $[\![\bot_\bot]\!]_\rho = \mathsf{F}^n_\bot$, $[\![M_A]\!]_\rho = f^n_A \neq \mathsf{F}^n_A$ and $[\![(\neg N_B)_{\neg B}]\!]_\rho = g^n_{\neg B} \neq \mathsf{F}^n_{\neg B}$. Hence the target proposition by contraposition.

- Show $\forall\sigma.\exists n.[\![(\neg(M_A \to N_B)_{A\to B})_{\neg(A\to B)}]\!]_\sigma = \mathsf{F}^n_{\neg(A\to B)} \;\Rightarrow\; \forall\rho(\exists k.[\![M_A]\!]_\rho = \mathsf{F}^k_A \text{ or } \exists m.[\![(\neg N_B)_{\neg B}]\!]_\rho = \mathsf{F}^m_B)$.

  Suppose for some ρ, $[\![M_A]\!]_\rho = f^k_A \neq \mathsf{F}^k_A$ and $[\![(\neg N_B)_{\neg B}]\!]_\rho = g^m_{\neg B} \neq \mathsf{F}^m_{\neg B}$. Then $[\![N_B]\!]_\rho = \mathsf{F}^{m<}_B$. Thus, for some σ, $[\![(M_A \to N_B)_{A\to B}]\!]_\sigma = \mathsf{F}^{m<}_{A\to B}$ and $[\![\bot_\bot]\!] = \mathsf{F}^m_\bot$, where $k \leq m$. Therefore, $[\![(\neg(M_A \to N_B)_{A\to B})_{\neg(A\to B)}]\!]_\sigma \neq \mathsf{F}^m_{\neg(A\to B)}$. Hence the target proposition by contraposition. □

4.3 Semantics of $NK2$

4.3.1 Interpretation of $NK2$

The interpretation of the terms of $NK2$ is the same as that of the terms of $LK2$.

4.3.2 Semantic Consequence in $NK2$

The notion of semantic consequence in $NK2$ is basically the same as that in λK, i.e., that of non-falsity preservation (even though, of course, the interpretations involved are different).

Definition 11. For any set Γ of terms and any term N_B,

$$\Gamma \vDash_{NK2} N_B \quad\Leftrightarrow\quad \forall\rho(\forall M_A \in \Gamma.\exists_1 m.[\![M_A]\!]_\rho = f^m_A \neq \mathsf{F}^m_A \Rightarrow \exists_1 n.[\![N_B]\!]_\rho = f^n_B \neq \mathsf{F}^n_B).$$

As a special case,

$$\vDash_{NK2} N_B \quad\Leftrightarrow\quad \forall\rho.\exists_1 n.[\![N_B]\!]_\rho = f^n_B \neq \mathsf{F}^n_B.$$

4.3.3 Completeness of $NK2$

$NK2$ is obviously deductively equivalent to $LK2$; thus, given the completeness of $LK2$, proving the completeness of $NK2$ is routine.

Theorem 10. *(Equivalence between LK2 and NK2) For any set* Γ *of terms and any term* N_B*,*

$$\vdash_{LK2} \Delta \quad \Leftrightarrow \quad \Delta \vdash_{NK2} \bot_\bot.$$

Proof From the equivalence between Gentzen's original LK and NK. □

Theorem 11. *(Completeness of NK2) For any set* Γ *of terms and any term* N_B*,*

$$\Gamma \vdash_{NK2} N_B \quad \Leftrightarrow \quad \Gamma \vDash_{NK2} N_B.$$

Proof From Theorem 9 and Theorem 10,

$$\vDash_{LK2} \Delta \quad \Leftrightarrow \quad \vdash_{LK2} \Delta \quad \Leftrightarrow \quad \Delta \vdash_{NK2} \bot_\bot.$$

But from Definition 6 and Definition 11 it is routine to show

$$\Delta \vDash_{NK2} \bot_\bot \quad \Leftrightarrow \quad \vDash_{LK2} \Delta.$$

Further generalization is routine. □

5 Conclusion and Future Work

We have shown that the three deductive systems of classical type theory, $\lambda K, LK2$, and $NK2$, are (sound and) complete with respect to the semantics with the common domain structure, the infinitely nested Boolean structure. Absurdity type $\bot$ is identified with the type of truth values, and, generally, the notion of truth values, truth and, in particular, falsity, has turned out to play a crucial role in classical type theory.

We now have a better understanding of the Curry-Howard correspondence for classical propositional logic:

Classical propositional logic	Classically-typed deductive systems $\lambda K, LK2$ &$NK2$
Propositions	Types
Proofs	Terms
Implication $\rightarrow$ (in propositions)	Function types
Absurdity $\bot$ (in propositions)	Type of truth values (infinitely Boolean-expanded)
Double-negation elimination	Type reduction $R\downarrow$
$[\![(M_A \rightarrow N_B)_{A\rightarrow B}]\!]_\rho$	Partial function from $[\![M_A]\!]_\rho$ to $[\![N_B]\!]_\rho$

For the normalization of classical proofs, recall Subsection 2.4. One of the most interesting things about the Curry-Howard correspondence for classical logic is that it is not simply an extension of the correspondence for intuitionistic logic. In the latter, absurdity type $\bot$ is the empty (or uninhabited) type, type with no members, whereas in the former, it is the type of truth values, and it plays a crucial, anchoring role in the infinitely nested Boolean structure.

In our model, we assumed for the sake of convenience that the only base type is absurdity type $\bot$. It should be obvious, however, that the model can be expanded in a straightforward fashion to include other base types. In such models, rank 0 of those types contains the usual members of the types, but the Boolean structure starts at rank 1; all the non-zero ranks are tied to the ranks of type $\bot$. So type $\bot$ is indispensable.

Obvious questions to answer in the immediate future are: What is the plausible *untyped* deductive system of classical type theory? and What is the structure like the system is complete with? Can the untyped version of λK be the system we are looking for? And what is the

appropriate semantics for it? There are impressive correspondences between the semantics we have offered and Scott's [12] D_∞ model of the untyped λ-calculus. Where Scott uses complete partial orders, we use Boolean structures. Can his idea for D_∞ models be transferred to the semantics of the untyped version of classical type theory? We are inclined to think so and hope to show it in a near future. Note also that the untyped versions of $LK2$ and $NK2$ are simply the usual sequent calculus and natural deduction for classical propositional logic, Gentzen's original LK and NK. So we should be able to give (perhaps unnecessarily complicated but interesting) denotational semantics of classical propositional logic, too.

Another obvious question to answer is about the relation between our semantics and syntactic or procedural (especially, continuation passing style) treatments of classical type theory such as [5]. Also, what is the relation between our calculi and other well-known classical calculi such as Parigot's [9] $\lambda\mu$-calculus and Curien and Herbelin's [4] $\bar{\lambda}\mu\tilde{\mu}$-calculus? How can they be understood from our semantic viewpoint?

Another obvious project is an expansion of classical type theory dealing with universal and existential quantification, i.e., where the type inferences are those of predicate logic. Here, just like in the minimal/intuitionistic cases, predication should be considered type construction and universal quantification should be considered a creation of *product* types, but since, unlike in the minimal/intuitionistic cases, the existential quantifier is definable with the universal quantifier, a simplification is possible.

A little longer-term project concerns the use of classical type theory in the semantics of natural language. While classical type theory may remain as a mere curiosity in the study of programming languages, there is much hope that it can be used more widely in the semantics of natural languages. In the Montague [8]/generalized-quantifier [2] theory, quantifiers such as *all Presidents* and *some Presidents*, conjoinments such as *Bush and Obama* and *Bush or Obama*, and even proper names such as *Bush* and *Obama* are considered to denote objects of type $(e \to t) \to t$ instead of e, where e is the type of individuals (ind) and t is the type of truth values (bool). $[\![\textit{Bush is a Republican}]\!] = \mathsf{T}$ not because $[\![\textit{is a Republican}]\!][\![\textit{Bush}]\!] = \mathsf{T}$ but $[\![\textit{Bush}]\!][\![\textit{is a Republican}]\!] = \mathsf{T}$. This is an odd theory as far as proper names are concerned, but to make proper names type e, we need a reduction rule $(e \to t) \to t \vdash e$. Our investigation into classical type theory began when we realized that this rule was just an instance of the double-negation elimination rule $(A \to \bot) \to \bot \vdash A$ if $\bot$ is identified with t.

The current type theory for natural language is based on the Montague/generalized-quantifier theory, and as a result, type assignments become very complicated very easily. We are inclined to think that there may be a way to simplify the theory by using classical type theory with type reductions. Another, related, issue is about scope ambiguities in natural language. *Every man loves some woman* or *Adam and Bob love Carol or Diane* can be understood in two ways, depending on whether the conjunction (or the universal quantifier) has scope over the disjunction (or the existential quantifier) or vice versa. However, the grammatical structure of the sentence seems only one way. How can we resolve this apparent conflict? Our suggested answer is by assigning different ranks to the operators involved. Generally, we can separate scope structures of natural language sentences from their grammatical structures by taking the former to be determined by ranks while the latter to be determined by types. As these examples should indicate, there seem to be a lot of things we can do in the semantics of natural language by using classical type theory.

References

[1] F. Barbanera and S. Berardi. A symmetric lambda calculus for classical program extraction. *Information and Computation* 125 (1996): 103–117.

[2] J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy* 4 (1981), 159–219.

[3] E. W. Beth. Semantic entailment and formal derivability. *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde*, new series 18 (1955): 309–42.

[4] P.-L. Curien and H. Herbelin. The duality of computation. In *ICFP '00 Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*, 233–243. ACM, 2000.

[5] P. de Groote. A CPS-translation of the $\lambda\mu$-calculus. In S. Tison (ed.), *Colloquium on Trees in Algebra and Programming (Lecture Notes in Computer Science 787)*, 85–99. Springer, 1994.

[6] G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift* 39 (1935): 176–210, 405–431.

[7] H. Herbelin. A λ-calculus structure isomorphic to Gentzen-style sequent calculus structure. In L. Pacholski and J. Tiuryn (eds.) *Computer Science Logic. 8th Workshop, CSL'94 (Lecture Notes in Computer Science 933)*, 61–75. Springer, 1995.

[8] R. Montague. The proper treatment of quantification in ordinary English. In J. Hintikka, J. Moravcsik, and P. Suppes (eds.), *Approaches to Natural Language*, 221–242. Reidel, 1973.

[9] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In A. Voronkov (ed.), *Logic Programming and Automated Reasoning: International Conference, LPAR '92 (Lecture Notes in Artificial Intelligence 624)*, 190–201. Springer, 1992.

[10] D. Prawitz. *Natural Deduction*. Almqvist and Wiksell, 1965.

[11] N. J. Rehof and M. H. Sørensen. The λ_Δ-calculus. In M. Hagiya and J. C. Mitchell (eds.), *Theoretical Aspects of Computer Software: International Symposium TACS '94 (Lecture Notes in Computer Science 789)*, 516–542. Springer, 1994.

[12] D. S. Scott. Continuous lattices. In F. W. Lawvere (ed.), *Toposes, Algebraic Geometry and Logic (Lecture Notes in Mathematics 274)*, 97–136. Springer, 1972.

[13] R. M. Smullyan. *First-Order Logic*. Dover, 1968.

[14] M. H. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard Isomorphism*. Elsevier, 2006.

[15] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, 2nd edn. Cambridge University Press, 1996.