A Finite Axiomatization of Propositional Type Theory in Pure Lambda Calculus

Mark Kaminski and Gert Smolka
Programming Systems Lab, Saarland University July 10, 2008

ABSTRACT. We consider simply typed lambda terms obtained with a single base type B and two constants $\bot$ and $\to$, where B is interpreted as the set of the two truth values, $\bot$ as falsity, and $\to$ as implication. We show that every value of the full set-theoretic type hierarchy can be described by a closed term and that every valid equation can be derived from three axioms with β and η. In contrast to the established approach, we employ a pure lambda calculus where constants appear as a derived notion.

1 Introduction

Propositional type theory employs simply typed lambda terms obtained with a single base type B, which is interpreted as the set of the two truth values. A concrete propositional type theory fixes some logical constants and a deduction system. We require denotational and deductive completeness. Denotational completeness means that every value of the full set-theoretic type hierarchy can be denoted by a closed term. Deductive completeness means that every valid formula can be deduced. An example of a valid formula is $f(f(fx)) \equiv fx$ where $f : BB$ and $x : B$ are variables. Deciding the validity of formulas obtained with higher-order quantification requires nonelementary time [10].

The first propositional type theory was devised by Henkin [6]. As constants Henkin takes all identity predicates, which can easily express the propositional connectives and the quantifiers. His deductive system is a Hilbert system whose inference rules are β and replacement of equals with equals. Henkin proves denotational and deductive completeness. Henkin's deductive system has been simplified and generalized to general type theory by Andrews [2, 3].

Altenkirch and Uustalu [1] have devised a different propositional type theory. They take the simply typed lambda calculus with a single base type B, the constants $\mathit{false} : B$ and $\mathit{true} : B$, and the polymorphic conditional $\mathit{if}_\sigma : B\sigma\sigma$ known from programming languages. They obtain deductive completeness with β, η, and four axiom schemes. Their completeness proof is based on Berger and Schwichtenberg's [4] normalization-as-evaluation technique.

Like Altenkirch and Uustalu, we base our propositional type theory on the simply typed lambda calculus with a single base type B. However, we omit the polymorphic conditional and employ only two constants, falsity and implication. We show that falsity and implication suffice for denotational completeness and obtain deductive completeness with β, η, and three axioms. Our proof of deductive completeness uses ideas from Henkin's completeness proof. We are the first ones to obtain a complete propositional type theory with only finitely many constants (the polymorphic conditional counts as infinitely many constants).

As it comes to the lambda calculus, we work with a pure system where constants appear only as a derived notion. Besides β and η we only have conversion steps of the form $us = ut$ where $s = t$ is an axiom. This simplifies the standard approach [9], where axioms can be applied with capture below binders (ξ-rule). Application with capture amounts to an implicit universal quantification of the free variables of the axioms. Implicit quantification is essential for algebraic systems, but is unnecessary for higher-order systems, which can express universal quantification with functional equality. For instance, $\forall x.\ x + 0 = x$ can be expressed as $\lambda x.\, x + 0 = \lambda x.x$.

The paper is organized as follows. We start with the pure lambda calculus and basic conversion proofs. We then provide a sequent system and show that it yields the same theorems as the conversion system. The rules of the sequent system formulate important properties of the conversion system. We then obtain a propositional type theory by committing to one base type, two constants, and three axioms. Next we formulate propositional sequent rules that are admissible for the general sequent system and that subsume the usual natural deduction rules. We then prove denotational and deductive completeness.

2 Terms and Conversion

We assume familiarity with the simply typed lambda calculus (see, e.g., [7]). *Types* (σ, τ, ρ) are obtained from *base types* (α) according to $\sigma ::= \alpha \mid \sigma\sigma$. Think of $\sigma\tau$ as the type of functions from σ to τ. *Terms* (s, t, u) are obtained from *names* (x, y, z, f, g) according to $s ::= x \mid \lambda x.s \mid ss$. We assume a *typing relation* $s : \sigma$ satisfying the following properties:

1. For every term s there is at most one type σ such that $s : \sigma$.
2. For every type σ there are infinitely many names x such that $x : \sigma$.
3. For all x, s, σ, τ: $\lambda x.s : \sigma\tau \iff x : \sigma \wedge s : \tau$.
4. For all s, t, σ: $s\,t : \sigma \iff \exists\tau\colon\ s : \tau\,\sigma \wedge t : \tau$.

A term σ is *well-typed* if there is a type σ such that $s : \sigma$. We will only consider well-typed terms. We use Λ to denote the set of all well-typed terms. We omit parentheses according to $\sigma\tau\rho \rightsquigarrow \sigma(\tau\rho)$ and $stu \rightsquigarrow (st)u$.

An *equation* is a pair $s = t$ of two terms s, t of the same type. We use e as a metavariable for single equations and A as a metavariable for finite sets of equations. Since equations are pairs, A is a binary relation on Λ. We use $\mathcal{N}s$, $\mathcal{N}e$ and $\mathcal{N}A$ to denote the sets of names that *occur free* in s, e and A, respectively. *Contexts* are obtained according to $C ::= [] \mid \lambda x.C \mid C\,s \mid s\,C$. The notation $C[s]$ describes the term obtained by replacing the hole $[]$ of C with s (capturing is ok). We assume a *substitution operation* s^x_t that yields for s, x, t a term that can be obtained from s by *capture-free substitution* of t for x, possibly after renaming of local names.

We define three *reduction relations*:

$$\begin{aligned}
\to_\beta &:= \{\,(C[(\lambda x.s)t],\, C[s^x_t]) \mid C[(\lambda x.s)t] \in \Lambda\,\} \\
\to_\eta &:= \{\,(C[\lambda x.sx],\, C[s]) \mid C[\lambda x.sx] \in \Lambda \wedge x \notin \mathcal{N}s\,\} \\
\to_A &:= \{\,(us, ut) \mid us \in \Lambda \wedge (s,t) \in A\,\}
\end{aligned}$$

The elements of the reduction relations are called *reduction steps*. We define *A-conversion* $\sim_A$ as the least equivalence relation on Λ that contains $\to_\beta$, $\to_\eta$, and $\to_A$. *λ-conversion* $\sim_\lambda$ is obtained as $\sim_\emptyset$. We say that s, t are *A-convertible* [*λ-convertible*] if $s \sim_A t$ [$s \sim_\lambda t$]. A *β-conversion step* is an equation $s = t$ such that $s \to_\beta t$ or $t \to_\beta s$. The definition of *η- and A-conversion steps* is analogous. A *basic conversion proof of $s = t$ with A* is a tuple $(s_1, \ldots, s_n)$ such that $s_1 = s$, $s_n = t$, and $s_i = s_{i+1}$ is a β-, η- or A-conversion step for all $i \in \{1, \ldots, n-1\}$.

PROPOSITION 1.1. *$s \sim_A t$ iff there exists a basic conversion proof of $s = t$ with A.*

PROPOSITION 1.2 (α-equivalence). *$\lambda x.s \sim_\lambda \lambda y.s^x_y$ if $\lambda x.s \in \Lambda$, $y \notin \mathcal{N}(\lambda x.s)$, and x and y have the same type.*

Proof. Follows with the basic conversion proof $(\lambda x.s,\ \lambda y.(\lambda x.s)y,\ \lambda y.s^x_y)$ consisting of an η- and a β-step. ■

Our definition of $\to_A$ is non-standard. At first glance, $\to_A$ may seem too weak to yield first-order rewriting. However, if we quantify the equational

axioms with λ, we get all we need. This can be seen from the following basic conversion proof of $f(y+0) = fy$ with $\{\lambda x.x{+}0 = \lambda x.x\}$:

$$\begin{aligned} f(y+0) &= f((\lambda x.x{+}0)y) && \beta \\ &= (\lambda g.f(gy))(\lambda x.x{+}0) && \beta \\ &= (\lambda g.f(gy))(\lambda x.x) && \lambda x.x{+}0 = \lambda x.x \\ &= f((\lambda x.x)y) && \beta \\ &= fy && \beta \end{aligned}$$

Our pure lambda calculus with A-conversion has the same expressive power as the lambda calculus with constants commonly used in programming language theory [9]. We prefer the pure calculus since it is technically simpler. In particular, it does not provide implicit quantification of the free variables of equational axioms. In the pure system, the distinction between variables and constants is not hard-wired but is obtained as a derived notion, as will be seen in § 5.

A context C *captures* a name x if the hole of C is in the scope of a binder λx. A context C is *admissible for A* if it does not capture any name in $\mathcal{N}A$.

PROPOSITION 1.3 (Compatibility). *If $s \sim_A t$ and C is admissible for A, then $C[s] \sim_A C[t]$.*

Proof. It suffices to show the property for single reduction steps. It is easy to verify that it holds for β- and η-steps. Let $us \to_A ut$ and C be admissible for A. Then we can choose a name x such that $(\lambda x.C[ux])s \to_\beta C[us]$ and $(\lambda x.C[ux])t \to_\beta C[ut]$. Hence $C[us] \sim_A C[ut]$. ■

A *generalized A-conversion step* is an equation $s = t$ such that either $s \sim_\lambda t$ or there exist an equation $(u{=}v) \in A$ and an A-admissible context C such that $s \sim_\lambda C[u]$ and $C[v] \sim_\lambda t$. A *generalized conversion proof of $s = t$ with A* is a tuple $(s_1, \ldots, s_n)$ such that $s_1 = s$, $s_n = t$, and $s_i = s_{i+1}$ is a generalized A-conversion step for all $i \in \{1, \ldots, n-1\}$.

PROPOSITION 1.4. *$s \sim_A t$ iff there exists a generalized conversion proof of $s = t$ with A.*

Generalized conversion proofs can be much shorter than basic conversion proofs. For instance, there is a generalized conversion proof of $f(y{+}0) = fy$ with $\{\lambda x.x{+}0 = \lambda x.x\}$ consisting of a single step. Both basic and generalized conversion proofs have their uses. The existence of basic conversion proofs is helpful when we show general properties of A-conversion.

Triv $\dfrac{}{A,\, e \vdash e}$ Weak $\dfrac{A' \vdash e}{A \vdash e}\ A' \subseteq A$ Cut $\dfrac{A \vdash e' \qquad A, e' \vdash e}{A \vdash e}$

Sub $\dfrac{A \vdash e}{A^x_s \vdash e^x_s}$

Ref $\dfrac{}{A \vdash s = s}$ Sym $\dfrac{A \vdash s = t}{A \vdash t = s}$ Trans $\dfrac{A \vdash s = t \qquad A \vdash t = u}{A \vdash s = u}$

Inst $\dfrac{A \vdash s_1 = s_2}{A \vdash t_1 = t_2}$ $t_1 \sim_\lambda C[s_1]$ and $C[s_2] \sim_\lambda t_2$ and C admissible for A

Figure 1.1. A Sequent-Based Proof System

3 A Sequent System

The rules in Figure 1.1 define a sequent-based proof system for equations. We will show that $A \vdash e$ iff there is a conversion proof of e with A. Thus the sequent rules show how conversion proofs can be combined into more complex conversion proofs. In fact, the rules formulate important properties of A-conversion. The rules Triv, Weak, Cut, and Sub express basic properties that are common for sequent systems. Note that Cut boosts conversion proofs since it allows conversion with respect to lemmas (i.e., provable equations). The rules Ref, Sym, and Trans account for the fact that conversion is an equivalence relation. The rule Inst (Instantiation) is the workhorse of the system. It incorporates λ-equivalence and closure under admissible contexts. Here is an instance of Inst that shows its power:

$$\frac{A \vdash \lambda xy.\, x + y = \lambda xy.\, y + x}{A \vdash s + t = t + s}$$

Inst also subsumes the rules

ζ $\dfrac{A \vdash sx = tx}{A \vdash t = s}\ x \notin \mathcal{N}(A \cup \{s{=}t\})$ ξ $\dfrac{A \vdash s = t}{A \vdash \lambda x.s = \lambda x.t}\ x \notin \mathcal{N}A$

that generalize well-known properties of λ-conversion.

PROPOSITION 1.5. $s \sim_A t \iff A \vdash s = t$

Proof. The direction $\Rightarrow$ is straightforward since Ref, Sym, Trans and Inst can simulate basic conversion proofs. The other direction requires more work. For each of the rules one shows that one can obtain a generalized conversion proof for the conclusion if one has basic conversion proofs for the premises. Proposition 1.3 is helpful for Cut and Inst. ■

4 Interpretations

An *interpretation* is a function $\mathcal{I}$ that maps every type to a nonempty set and every name $x : \sigma$ to an element of $\mathcal{I}\sigma$. We will only consider interpretations that map a functional type $\sigma\tau$ to the set of all total functions from $\mathcal{I}\sigma$ to $\mathcal{I}\tau$. Every interpretation $\mathcal{I}$ can be extended uniquely to a function $\hat{\mathcal{I}}$ which maps every term $s : \sigma$ to an element of $\mathcal{I}\sigma$ and treats applications and abstractions as one would expect. An interpretation $\mathcal{I}$ *satisfies an equation* $s = t$ if $\mathcal{I}s = \mathcal{I}t$. An interpretation *satisfies* A if it satisfies every equation in A. We write $s \approx_A t$ if every interpretation that satisfies A also satisfies $s = t$.

PROPOSITION 1.6 (Soundness). $s \sim_A t \Longrightarrow s \approx_A t$

5 Propositional Type Theory

Figure 1.2 shows our axiomatization of propositional type theory. Technically, we fix a base type B and five names $\bot$, $\to$, x, y, and f and define P as the set containing the following three equations:

$$\begin{aligned} \lambda x.\ \top \to x &= \lambda x.\ x && \text{I}\top \\ \lambda fx.\ f\bot \to f\top \to fx &= \lambda fx.\ \top && \text{BCA} \\ \lambda xy.\ (x \equiv y) \to x &= \lambda xy.\ (x \equiv y) \to y && \text{Rep} \end{aligned}$$

Note that only $\bot$ and $\to$ occur free in P. We call $\bot$ and $\to$ *constants* and all other names *variables*. The variable constant distinction is exploited in Figure 1.2, where the axioms appear with implicit quantification. Note that in our system implicit quantification is a notational device and not a syntactic feature.

Convention. In the rest of the paper we will only consider types, terms, and equations that can be obtained with the single base type B. We use $\mathcal{V}s$ to denote the set of all variables that occur free in s. A term s is *closed* if $\mathcal{V}s = \emptyset$, and *open* otherwise.

PROPOSITION 1.7 (I$\bot$). $P \vdash \bot \to x = \top$

Proof. Here is a (generalized) conversion proof:

$$\begin{aligned} \bot \to x &= \bot \to \top \to x && \text{I}\top \\ &= \top && \text{BCA} \end{aligned}$$

■

Base Type B

Constants $\bot : B;\ \ \to : BBB$

Variables $x, y : B;\ \ f : BB$

Notations

$$\begin{array}{llll} \top & := & \bot \to \bot & \qquad s \vee t & := & (s \to t) \to t \\ \neg s & := & s \to \bot & \qquad s \wedge t & := & \neg(\neg s \vee \neg t) \\ & & & \qquad s \equiv t & := & (s \to t) \wedge (t \to s) \end{array}$$

Precedence $=,\ \ \equiv,\ \ \to,\ \ \vee,\ \ \wedge,\ \ \neg$

Parentheses $s \to t \to u \ \rightsquigarrow\ s \to (t \to u)$

Axioms

$$\begin{array}{rcll} \top \to x & = & x & \text{I}\top \\ f\bot \to f\top \to fx & = & \top & \text{BCA} \\ (x \equiv y) \to x & = & (x \equiv y) \to y & \text{Rep} \end{array}$$

Figure 1.2. Axiomatization of Propositional Type Theory

A *propositional interpretation* is an interpretation $\mathcal{I}$ such that $\mathcal{I}B = \{0,1\}$, $\mathcal{I}\bot = 0$ and $\mathcal{I}(\to)ab = \mathsf{if}\ a{=}0\ \mathsf{then}\ 1\ \mathsf{else}\ b$ for all $a, b \in \{0,1\}$.

PROPOSITION 1.8. *Every propositional interpretation satisfies P.*

An equation $s = t$ is *propositionally valid* if $\hat{\mathcal{I}}s = \hat{\mathcal{I}}t$ for every propositional interpretation $\mathcal{I}$. We write $s \approx_2 t$ if $s = t$ is propositionally valid. An example of a propositionally valid equation is $f(f(fx)) = fx$ where $f : BB$ and $x : B$ (to verify validity, consider all 4 functions $\{0,1\} \to \{0,1\}$).

PROPOSITION 1.9 (Semantic Completeness). $s \approx_P t \iff s \approx_2 t$

Proof. The direction $\Rightarrow$ holds by Proposition 1.8. For the other direction we assume that $\mathcal{I}$ is an interpretation that satisfies P. We show $\mathcal{I}s = \mathcal{I}t$. Case Analysis.

Let $\mathcal{I}\bot = \mathcal{I}\top$. Then $\mathcal{I}(\lambda x.x) = \mathcal{I}(\lambda x.\top)$ by I$\bot$ (Proposition 1.7) and Axiom I$\top$. Hence $\mathcal{I}B$ is a singleton. Thus all types denote singletons and hence $\mathcal{I}s = \mathcal{I}t$.

Let $\mathcal{I}\bot \neq \mathcal{I}\top$. Because of I$\bot$ and I$\top$ it suffices to show that $\mathcal{I}B \subseteq \{\mathcal{I}\bot, \mathcal{I}\top\}$. Suppose for contradiction $a \in \mathcal{I}B - \{\mathcal{I}\bot, \mathcal{I}\top\}$, and let b be a function from $\mathcal{I}B \to \mathcal{I}B$ such that $b(\mathcal{I}\bot) = b(\mathcal{I}\top) = \mathcal{I}\top$ and $ba = a$. Instantiating BCA and I$\top$ with a and b yields $a = \mathcal{I}\top$, thus contradicting

the assumption. ■

Note that so far we have only used the axioms I$\top$ and BCA. The axiom Rep is only needed for deductive completeness. Brown [5] has constructed a general model that shows that P without Rep is deductively weaker than P. An alternative to Rep that yields the same deductive power is $\lambda xy.\, x \vee y = \lambda xy.\, y \vee x$.

A *propositional term* is a term that can be obtained according to $s ::= x \mid s \to s$ where $x : B$ is a name. A *propositional equation* is an equation $s = t$ where s, t are propositional terms. A *tautology* is a propositional equation $s = t$ such that $s \approx_2 t$. Note that the axioms I$\top$ and Rep are tautologies.

LEMMA 1.10. *Let s be a closed propositional term. Then:*

1. *$s \sim_P \bot$ or $s \sim_P \top$.*
2. *If $s \approx_2 \top$, then $s \sim_P \top$.*

Proof. A closed propositional term contains no other names but $\bot$ and $\to$. We show (1) by induction on the size of s. If $s = \bot$, then $s \sim_P \bot$ by Ref. If $s = s_1 \to s_2$, then $s_1 \sim_P \bot$ or $s_1 \sim_P \top$ and $s_2 \sim_P \bot$ or $s_2 \sim_P \top$ by induction. Now the claim follows with I$\top$ and I$\bot$. Claim (2) follows by (1) and Soundness. ■

A term s is *β-normal* if there exists no term t such that $s \to_\beta t$. It is well-known [7] that for every well-typed term s there exists a β-normal term t such that $s \sim_\lambda t$ and $\mathcal{N}t \subseteq \mathcal{N}s$.

PROPOSITION 1.11.
Every closed and β-normal term $s : B$ is propositional.

Proof. By induction on the size of s. Let $s : B$ be β-normal and closed. Then $s = x s_1 \dots s_n$ where $s_1, \dots, s_n$ are closed and β-normal. Since s is closed, either $x = \bot$ or $x = \to$. If $x = \bot$, then $n = 0$ and hence s is propositional. If $x = \to$, then $n = 2$ and s_1 and s_2 are propositional by induction. Hence s is propositional. ■

Let E be a set of equations. We say that P is *complete for E* if $s \approx_2 t$ implies $s \sim_P t$ for every equation $s = t$ in E. Eventually, we will show that P is complete for all equations.

PROPOSITION 1.12. *P is complete for all closed equations $s = \top$.*

Proof. Let $s \approx_2 \top$. There exists a β-normal and closed term $t : B$ such that $s \sim_\lambda t$. By Soundness and Semantic Completeness we have $t \approx_2 \top$. By Proposition 1.11 we know that t is propositional. Hence $t \sim_P \top$ by Lemma 1.10 (2). Thus $s \sim_P \top$. ■

$$\text{MP}\quad \frac{A \vdash s \to t = \top \qquad A \vdash s = \top}{A \vdash t = \top} \quad P \subseteq A$$

$$\text{BE}\quad \frac{A \vdash s \equiv t = \top}{A \vdash s = t} \quad P \subseteq A \qquad \text{BE}^{-}\quad \frac{A \vdash s = t}{A \vdash s \equiv t = \top} \quad P \subseteq A$$

$$\text{Taut}\quad \frac{}{A \vdash e} \quad P \subseteq A \text{ and } e \text{ tautology}$$

$$\text{CA}\quad \frac{A \vdash e^x_\bot \qquad A \vdash e^x_\top}{A \vdash e} \quad P \subseteq A \qquad \text{Ded}\quad \frac{A, s = \top \vdash t = \top}{A \vdash s \to t = \top} \quad P \subseteq A$$

$$\text{DE}\quad \frac{A,\ s = \top \vdash t = \top \qquad A,\ t = \top \vdash s = \top}{A \vdash s = t} \quad P \subseteq A$$

Figure 1.3. Admissible Rules for $\vdash$

6 Propositional Sequent Rules

Figure 1.3 collects some sequent rules that are admissible for the relation $\vdash$ defined in § 3. Admissibility of a rule for $\vdash$ means that the conclusion of the rule is satisfied by $\vdash$ if the premises of the rule are satisfied by $\vdash$. MP stands for modus ponens, BE for Boolean equality, CA for case analysis, Ded for deductivity, and DE for deductivity and Boolean equality.

The rules formulate important properties of the conversion relations $\sim_A$ with $P \subseteq A$. By Proposition 1.5 a rule is admissible for $\vdash$ iff there is a conversion proof for the conclusion of the rule if there are conversion proofs for the premises of the rule. Hence the rules tell us how we can construct complex conversion proofs from simpler ones. The rules will be crucial for the completeness proof to come. Seen semantically, the rules express well-known properties of propositional logic.

PROPOSITION 1.13. MP *is admissible for* $\vdash$.

Proof. Let $P \subseteq A$. By Cut it suffices to give a conversion proof of $t = \top$ with $A \cup \{s{\to}t = \top,\ s = \top\}$. Here it is:

$t = \top \to t$	IT
$= s \to t$	$s = \top$
$= \top$	$s{\to}t = \top$

■

PROPOSITION 1.14. BE *is admissible for* $\vdash$.

Proof. Let $P \subseteq A$. Here is a conversion proof of $s = t$ with $A \cup \{s \equiv t = \top\}$:

$$\begin{array}{ll} s = \top \to s & \mathrm{I}\top \\ \quad = (s \equiv t) \to s & s \equiv t = \top \\ \quad = (s \equiv t) \to t & \mathrm{Rep} \\ \quad = \top \to t & s \equiv t = \top \\ \quad = t & \mathrm{I}\top \end{array}$$

■

Axiom Rep is used for the first time in the above proof. In fact, our axiomatization contains Rep so that we can show that BE is admissible.

LEMMA 1.15. *If* $s = \top$ *is a tautology, then* $P \vdash s = \top$.

Proof. By induction on $|\mathcal{V}s|$. If s is closed, the claim follows by Lemma 1.10 (2). Otherwise, let $x \in \mathcal{V}s$. Then $s^x_\bot = \top$ and $s^x_\top = \top$ are tautologies. By induction $P \vdash s^x_\bot = \top$ and $P \vdash s^x_\top = \top$. Hence $P \vdash s = \top$ follows with a conversion proof:

$$\begin{array}{ll} s = \top \to s & \mathrm{I}\top \\ \quad = \top \to \top \to s & \mathrm{I}\top \\ \quad = s^x_\bot \to \top \to s & s^x_\bot = \top \\ \quad = s^x_\bot \to s^x_\top \to s & s^x_\top = \top \\ \quad = \top & \mathrm{BCA} \end{array}$$

■

PROPOSITION 1.16. BE^- *is admissible for* $\vdash$.

Proof. Let $P \subseteq A$. Since $x \equiv x = \top$ is a tautology, it suffices by Lemma 1.15 to give a conversion proof of $s \equiv t = \top$ with $A \cup \{s = t,\, x \equiv x = \top\}$. This is straightforward. ■

PROPOSITION 1.17. Taut *is admissible for* $\vdash$.

Proof. By BE and Lemma 1.15. ■

PROPOSITION 1.18. CA *is admissible for* $\vdash$.

Proof. By BE and BE^- it suffices to show the claim for $e = (s = \top)$. This can be done with BCA, Inst, and MP. ■

LEMMA 1.19. $P \vdash x{\to}fx = x{\to}f\top$

Proof. Follows by CA and conversion proofs with I⊥. ■

LEMMA 1.20. *Let $P \subseteq A$ and $t_1 = t_2$ be an $A \cup \{s = \top\}$-conversion step. Then $A \vdash s{\to}t_1 = s{\to}t_2$.*

Proof. If $t_1 = t_2$ is a A-conversion step, the claim follows by Inst. If $t_1 = t_2$ is a $\{s = \top\}$-conversion step, we have $t_1 = us$ and $t_2 = u\top$ for some u. Hence $P \vdash s{\to}t_1 = s{\to}t_2$ by Lemma 1.19. The claim follows by Weak. ■

PROPOSITION 1.21. Ded *is admissible for* $\vdash$.

Proof. Let $P \subseteq A$ and $A, s = \top \vdash t = \top$. Then there exists a basic conversion proof of $t = \top$ with $A \cup \{s = \top\}$. Hence $A \vdash s{\to}t = s{\to}\top$ by Lemma 1.20. Since $A \vdash x{\to}\top = \top$ by Taut, we have $A \vdash s{\to}t = \top$. ■

PROPOSITION 1.22. DE *is admissible for* $\vdash$.

Proof. By Ded and BE it suffices to give a conversion proof of $s{\equiv}t = \top$ with $s{\to}t = \top$, $t{\to}s = \top$, and tautologies. This is straightforward. ■

7 Denotational Completeness

We fix a propositional interpretation $\mathcal{B}$. Then we have $\mathcal{I}\sigma = \mathcal{B}\sigma$ for every propositional interpretation $\mathcal{I}$ and every type σ. Moreover, we have $\hat{\mathcal{I}}s = \hat{\mathcal{B}}s$ for every closed term s and every propositional interpretation $\mathcal{I}$. We will show that P is *denotationally complete*, that is, for every type σ and every value $a \in \mathcal{B}\sigma$ there is a closed term $s : \sigma$ such that $\hat{\mathcal{B}}s = a$.

We will define a family of *quote functions* $\downarrow^\sigma : \mathcal{B}\sigma \to \Lambda_0^\sigma$ by recursion on types. Λ_0^σ is the set of all closed terms of type σ. The quote functions will satisfy $\hat{\mathcal{B}}(\downarrow^\sigma a) = a$ for all $a \in \mathcal{B}\sigma$ and all types σ.

The definition of the basic quote function $\downarrow^B$ is straightforward. To explain the definition of the other quote functions, we consider the special case $\downarrow^{\sigma B}$. We start with

$$\downarrow^{\sigma B}(a) \;=\; \lambda x.\, \bigvee_{\substack{b \in \mathcal{B}\sigma \\ ab = 1}} x \doteq_\sigma (\downarrow^\sigma b)$$

It remains to define a closed term $\doteq_\sigma$ that denotes the identity predicate for $\mathcal{B}\sigma$. If $\sigma = B$, $\lambda xy.\, x \equiv y$ does the job. If $\sigma = \sigma_1\sigma_2$, we rely on recursion and define

$$\doteq_\sigma \;=\; \lambda fg.\, \bigwedge_{a \in \mathcal{B}\sigma_1} f(\downarrow^{\sigma_1} a) \doteq_{\sigma_2} g(\downarrow^{\sigma_1} a)$$

$$\downarrow^{\sigma_1 \dots \sigma_n B} a \;:=\; \lambda x_1 \dots x_n.\; \bigvee_{\substack{\langle b_i \in \mathcal{B}\sigma_i \rangle \\ a b_1 \dots b_n = 1}} \bigwedge_{1 \le j \le n} x_j \doteq_{\sigma_j} (\downarrow^{\sigma_j} b_j) \qquad \mathrm{D}\downarrow$$

$$\forall_\sigma \;:=\; \lambda f.\; \bigwedge_{a \in \mathcal{B}\sigma} f(\downarrow^\sigma a) \qquad \mathrm{D}\forall$$

$$\doteq_B \;:=\; \lambda xy.\; x \equiv y \qquad \mathrm{D}\doteq$$

$$\doteq_{\sigma\tau} \;:=\; \lambda fg.\; \forall_\sigma(\lambda x.\; fx \doteq_\tau gx) \qquad \mathrm{D}\doteq$$

$\langle b_i \in \mathcal{B}\sigma_i \rangle$ abbreviates $(b_1, \dots, b_n) \in \mathcal{B}\sigma_1 \times \dots \times \mathcal{B}\sigma_n$

Figure 1.4. Quote Functions $\downarrow^\sigma$ and Terms $\forall_\sigma$ and $\doteq_\sigma$

Figure 1.4 shows the full definition of the quote functions. A disjunction with an empty index set denotes $\bot$, and a conjunction with an empty index set denotes $\top$. The notations $\doteq_\sigma$ and $\forall_\sigma$ will be used in the following. We will write $\forall_\sigma x.\, s$ for $\forall_\sigma(\lambda x.s)$. The notational operator $\doteq_\sigma$ will be used with a precedence higher than $\neg$ (i.e, $\neg\, s \doteq_\sigma t \rightsquigarrow \neg(s \doteq_\sigma t)$).

PROPOSITION 1.23. *The terms $\downarrow^\sigma a$, $\forall_\sigma$ and $\doteq_\sigma$ are closed.*

PROPOSITION 1.24. *Let σ be a type, $a, b \in \mathcal{B}\sigma$, and $\varphi \in \mathcal{B}(\sigma B)$. Then:*

1. $\hat{\mathcal{B}}(\downarrow^\sigma a) = a$
2. $\hat{\mathcal{B}}(\forall_\sigma)\varphi = 1 \iff \forall c \in \mathcal{B}\sigma\colon\ \varphi c = 1$
3. $\hat{\mathcal{B}}(\doteq_\sigma)ab = 1 \iff a = b$

THEOREM 1.25 (Denotational Completeness). *Let σ be a type and $a \in \mathcal{B}\sigma$. Then there is a closed term s such that $\hat{\mathcal{B}}s = a$.*

Proof. Follows from Proposition 1.24 (1). ■

8 Deductive Completeness

PROPOSITION 1.26. *P is complete for all equations if it is complete for all equations of the form $s = \top$.*

Proof. Assume P is complete for all equations of the form $s = \top$. Let $s \approx_2 t$. We show $s \sim_P t$. We choose distinct variables $\bar{x} = x_1 \dots x_n$ that do not occur in $s = t$ such that $s\bar{x} : B$. We have $s\bar{x} \approx_2 t\bar{x}$ and hence $s\bar{x} \equiv t\bar{x} \approx_2 \top$. By the assumption we have $s\bar{x} \equiv t\bar{x} \sim_P \top$. By BE we have $P \vdash s\bar{x} = t\bar{x}$. Thus $P \vdash s = t$ by Inst (or repeated use of ζ). ■

LEMMA 1.27. *P is complete for all equations if for all types σ there are variables f, x such that $P \vdash \forall_\sigma f \to fx = \top$.*

Proof. Assume that for all types σ there are variables f, x such that $P \vdash \forall_\sigma f \to fx = \top$. Let $s \approx_2 \top$. By Proposition 1.26 it suffices to show that $s \sim_P \top$. There exist variables $\bar{x} = x_1 \dots x_n$ such that $\forall \bar{x}.s$ is closed. By the second assumption we have $\forall \bar{x}.s \approx_2 \top$. By Proposition 1.12 we have $P \vdash \forall \bar{x}.s = \top$. Hence $P \vdash s = \top$ by the first assumption, Sub and MP. ■

LEMMA 1.28. $P \vdash x \doteq_\sigma x = \top$

Proof. By induction on σ, exploiting Taut. ■

LEMMA 1.29. *If $a, b \in \mathcal{B}\sigma$ are distinct, then $P \vdash (\downarrow^\sigma a) \doteq_\sigma (\downarrow^\sigma b) = \bot$.*

Proof. Let $a, b \in \mathcal{B}\sigma$ be distinct. Then:

$\neg(\downarrow^\sigma a) \doteq_\sigma (\downarrow^\sigma b) \approx_2 \top$	Proposition 1.24
$P \vdash \neg(\downarrow^\sigma a) \doteq_\sigma (\downarrow^\sigma b) = \top$	Proposition 1.12
$P \vdash \neg\neg(\downarrow^\sigma a) \doteq_\sigma (\downarrow^\sigma b) = \neg\top$	Inst
$P \vdash (\downarrow^\sigma a) \doteq_\sigma (\downarrow^\sigma b) = \bot$	Taut

■

LEMMA 1.30. *If $\sigma = \rho\tau$, then for all $a \in \mathcal{B}\sigma$ and $b \in \mathcal{B}\rho$:*
$P \vdash (\downarrow^\sigma a)(\downarrow^\rho b) = \downarrow^\tau (ab)$.

Proof. Let $\sigma = \rho\tau$ and $\tau = \tau_1 \dots \tau_n B$. Let $a \in \mathcal{B}\sigma$ and $b \in \mathcal{B}\rho$. By D$\downarrow$ and β, we have

$$
\begin{aligned}
& (\downarrow^{\rho\tau_1\dots\tau_n B} a)(\downarrow^\rho b) \\
= & \left(\lambda x y_1 \dots y_n . \bigvee_{\substack{c \in \mathcal{B}\rho \\ \langle d_i \in \mathcal{B}\tau_i \rangle \\ acd_1\dots d_n = 1}} x \doteq_\rho (\downarrow^\rho c) \ \wedge \bigwedge_{1 \le j \le n} y_j \doteq_{\tau_j} (\downarrow^{\tau_j} d_j) \right) (\downarrow^\rho b) \\
= & \ \lambda y_1 \dots y_n . \bigvee_{\substack{c \in \mathcal{B}\rho \\ \langle d_i \in \mathcal{B}\tau_i \rangle \\ acd_1\dots d_n = 1}} (\downarrow^\rho b) \doteq_\rho (\downarrow^\rho c) \ \wedge \bigwedge_{1 \le j \le n} y_j \doteq_{\tau_j} (\downarrow^{\tau_j} d_j)
\end{aligned}
$$

By Lemma 1.28, Lemma 1.29, and Taut we obtain

$$
\begin{aligned}
= & \ \lambda y_1 \dots y_n . \bigvee_{\substack{\langle d_i \in \mathcal{B}\tau_i \rangle \\ abd_1\dots d_n = 1}} \bigwedge_{1 \le j \le n} y_j \doteq_{\tau_j} (\downarrow^{\tau_j} d_j) \\
= & \ \downarrow^{\tau_1\dots\tau_n}(ab)
\end{aligned}
$$

■

LEMMA 1.31. *Let I and J be finite sets and $x_{i,j} : B$ be a variable for all $i \in I$, $j \in J$. Moreover, let $[I \to J]$ be the set of all total functions $I \to J$. Then the equation*

$$\bigwedge_{i\in I} \bigvee_{j\in J} x_{i,j} \;=\; \bigvee_{\varphi\in[I\to J]} \bigwedge_{i\in I} x_{i,\varphi i}$$

is a tautology and hence is provable with P.

Proof. Let s and t be the left and the right term of the equation, and let $\mathcal{I}$ be a propositional interpretation. We have to show that $\hat{\mathcal{I}}s = 1$ iff $\hat{\mathcal{I}}t = 1$. Let $\hat{\mathcal{I}}s = 1$. Then for every $i \in I$ there exists a $j \in J$ such that $\mathcal{I}(x_{i,j}) = 1$. Hence there exists a function $\varphi \in [I \to J]$ such that $\mathcal{I}(x_{i,\varphi i}) = 1$ for every $i \in I$. Hence $\hat{\mathcal{I}}t = 1$. The other direction follows analogously. ■

LEMMA 1.32. $P \vdash \bigvee_{a\in\mathcal{B}\sigma} x \doteq_\sigma (\downarrow^\sigma a) = \top$

Proof. By induction on σ. If $\sigma = B$, the claim follows with Taut. Otherwise, let $\sigma = \sigma_1\sigma_2$. We show the claim with a conversion proof.

$$\begin{aligned}
&\bigvee_{a\in\mathcal{B}\sigma} x \doteq_\sigma (\downarrow^\sigma a) \\
=\ &\bigvee_{a\in\mathcal{B}\sigma} \forall_{\sigma_1} y.\ xy \doteq_{\sigma_2} (\downarrow^\sigma a)y && \text{D}\doteq \\
=\ &\bigvee_{a\in\mathcal{B}\sigma} \bigwedge_{b\in\mathcal{B}\sigma_1} x(\downarrow^{\sigma_1} b) \doteq_{\sigma_2} (\downarrow^\sigma a)(\downarrow^{\sigma_1} b) && \text{D}\forall \\
=\ &\bigvee_{a\in\mathcal{B}\sigma} \bigwedge_{b\in\mathcal{B}\sigma_1} x(\downarrow^{\sigma_1} b) \doteq_{\sigma_2} (\downarrow^{\sigma_2}(ab)) && \text{Lemma 1.30} \\
=\ &\bigwedge_{b\in\mathcal{B}\sigma_1} \bigvee_{c\in\mathcal{B}\sigma_2} x(\downarrow^{\sigma_1} b) \doteq_{\sigma_2} (\downarrow^{\sigma_2} c) && \text{Lemma 1.31, } \mathcal{B}\sigma = [\mathcal{B}\sigma_1 \to \mathcal{B}\sigma_2] \\
=\ &\bigwedge_{b\in\mathcal{B}\sigma_1} \top && \text{induction for } \sigma_2 \\
=\ &\top && \text{Taut}
\end{aligned}$$

■

LEMMA 1.33. *If $P, x \doteq_\sigma y = \top \vdash x = y$, then $P \vdash \forall_\sigma f \to fx = \top$.*

Proof. Let $P, x \doteq_\sigma y = \top \vdash x = y$. Then we obtain with DE and Ded

$$P \vdash x \doteq_\sigma y \to fx = x \doteq_\sigma y \to fy \tag{1.1}$$

Now we show the claim with a conversion proof.

$$\begin{aligned}
& \forall_\sigma f \to fx \\
= \ & \top \to \forall_\sigma f \to fx && \text{Taut} \\
= \ & \left(\bigvee_{a \in \mathcal{B}\sigma} x \doteq_\sigma (\downarrow^\sigma a) \right) \to \forall_\sigma f \to fx && \text{Lemma 1.32} \\
= \ & \bigwedge_{a \in \mathcal{B}\sigma} x \doteq_\sigma (\downarrow^\sigma a) \to \forall_\sigma f \to fx && \text{Taut} \\
= \ & \bigwedge_{a \in \mathcal{B}\sigma} x \doteq_\sigma (\downarrow^\sigma a) \to \forall_\sigma f \to f(\downarrow^\sigma a) && (1.1) \\
= \ & \bigwedge_{a \in \mathcal{B}\sigma} x \doteq_\sigma (\downarrow^\sigma a) \to \left(\bigwedge_{b \in \mathcal{B}\sigma} f(\downarrow^\sigma b) \right) \to f(\downarrow^\sigma a) && \text{D}\forall \\
= \ & \top && \text{Taut}
\end{aligned}$$

■

LEMMA 1.34. $P,\ x \doteq_\sigma y = \top \ \vdash \ x = y$

Proof. By induction on σ. If $\sigma = B$, the claim follows with BE. Otherwise, let $\sigma = \sigma_1 \sigma_2$. By D$\doteq$ we have

$$P,\ x \doteq_\sigma y = \top \ \vdash \ \forall_{\sigma_1} z.\ xz \doteq_{\sigma_2} yz = \top$$

for some variable z. By induction for σ_1 and Lemma 1.33 we have

$$P \ \vdash \ (\forall_{\sigma_1} z.\ xz \doteq_{\sigma_2} yz) \to xz \doteq_{\sigma_2} yz = \top$$

By MP we have

$$P,\ x \doteq_\sigma y = \top \ \vdash \ xz \doteq_{\sigma_2} yz = \top$$

Now induction for σ_2 and Cut yield

$$P,\ x \doteq_\sigma y = \top \ \vdash \ xz = yz$$

which yields the claim with Inst. ■

THEOREM 1.35 (Deductive Completeness).
P is complete for all equations.

Proof. Follows by Lemma 1.34, Lemma 1.33 and Lemma 1.27. ■

9 Final Remarks

We have shown that the pure simply typed lambda calculus furnished with three axioms is a complete deduction system for propositional type theory with falsity and implication. This yields the most minimal set-up of propositional type theory known so far. Our results carry over to propositional type theories with quantifiers and identity predicates [8]. The motivation of our research is to better understand the deductive power of the lambda-calculus when applied to higher-order logic. The idea to capture higher-order logic as a lambda theory appears in Mitchell [9] (Example 4.4.7).

Acknowledgments. We benefited from discussions with Chad E. Brown and Jan Schwinghammer. Jan pointed us to [1].

BIBLIOGRAPHY

[1] Thorsten Altenkirch and Tarmo Uustalu. Normalization by evaluation for $\lambda^{\to 2}$. In Yukiyoshi Kameyama and Peter J. Stuckey, editors, *Symposium on Functional and Logic Programming*, volume 2998 of *LNCS*, pages 260–275. Springer, 2004.

[2] Peter B. Andrews. A Reduction of the Axioms for the Theory of Propositional Types. *Fundamenta Mathematicae*, 52:345–350, 1963.

[3] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*, volume 27 of *Applied Logic Series*. Kluwer Academic Publishers, 2nd edition, 2002.

[4] Ulrich Berger and Helmut Schwichtenberg. An Inverse of the Evaluation Functional for Typed Lambda-Calculus. In *Proc. 6th Annual IEEE Symposium on Logic in Computer Science (LICS'91)*, pages 203–211. IEEE Computer Society Press, 1991.

[5] Chad E. Brown. Personal communication, January 2007.

[6] Leon Henkin. A Theory of Propositional Types. *Fundamenta Mathematicae*, 52:323–344, 1963.

[7] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-calculus*. Cambridge University Press, 1986.

[8] Mark Kaminski. *Completeness Results for Higher-Order Equational Logic*. Master's Thesis, Saarland University, 2006.

[9] John C. Mitchell. *Foundations for Programming Languages*. Foundations of Computing. The MIT Press, 1996.

[10] S. Vorobyov. The most nonelementary theory. *Information and Computation*, 190(2):196–219, 2004.