Supplement to Lambda Logic

Michael Beeson *

San José State University, San Jose, CA, USA

Abstract

This document contains some explanations and proofs that had to be omitted from (4) to meet the length limitation. It is meant to be read only as a supplement to (4).

KEYWORDS: automated deduction, computer proofs, unification, second order, Otter

1. Predicate Abstraction

Should we allow the formation of new predicates by λ-abstraction, such as $\lambda y.\, P(x, y)$? This is used in Otter2 (5). It is, however, not necessary to build this into the syntax of lambda logic. Indeed, omitting it offers greater flexibility: we can then selectively allow predicate abstraction over certain predicates, while not allowing it over others. To allow predicate abstraction over P, include a function symbol χ_P and an axiom expressing that χ_P is the characteristic function of P.

5. Proof of Theorem 5.3

We repeat the statement of the theorem and then give a complete proof; (4) gives only a sketch.

Theorem 5.3 (Axiomatization of first-order theorems): *Let T be a first order theory, and let A be a first order sentence. Then T proves A in lambda logic if and only if for some positive integer n, T plus "there exist n distinct things" proves A in first order logic.*

Proof. First suppose A is provable from T plus "there exist n distinct things". We show A is provable in lambda logic, by induction on the length of the proof of A. Since lambda logic includes first order logic, the induction step is rivial.

*Research supported by NSF grant number CCR-0204362.

For the basis case we must show that lambda logic proves "there exist n distinct things" for each positive integer n. The classical constructions of numerals in lambda calculus produce infinitely many distinct things. However, it must be checked that their distinctness is provable in lambda logic. Defining numerals as on p. 130 of (1) we verify by induction on n that for all $m < n$, $\lceil m \rceil \neq \lceil n \rceil$ is provable in lambda logic. If $m < n+1$ then either $m = n$, in which case we are done by the induction hypothesis, or $m = n$. So what has to be proved is that for each n, lambda logic proves $\lceil n \rceil \neq \lceil n+1 \rceil$. This in turn is verifiable by induction on n.

Conversely, suppose that A is not provable in T plus "there exist n distinct things" for any n. Then by the completeness theorem for first order logic, there is an infinite model M of $\neg A$; indeed we may assume that M has infinitely many elements not denoted by closed terms of T. We will show that M can be expanded to a lambda model $\hat{M}$ satisfying the same first order formulas, by defining arbitrarily the required operation λ^* on M-terms, and then inductively defining relations $E(x,y)$ and Ap_M to serve as the interpretations of equality and Ap in $\hat{M}$.

To do this we define the relation Ap_M and a binary relation E on M by simultaneous induction. Ap_M will serve as the interpretation of Ap and E will serve as the interpretation of equality.† Since M is infinite we can define an element 0 and a pairing function $\langle a, b\rangle$ on M in such a way that the interpretations of the losed terms of T are never pairs, and 0 is not a pair. Define $< a, b, c >=< a, < b, c >>$, etc. Define the successor of x to be $s(x) =< 0, x >$, and define the "numeral" $\bar{m}$ inductively by $\bar{0} = 0$ and $m \bar{+} 1 = s(\bar{m})$. Henceforth we drop the bars, writing for example $\langle 1, k\rangle$ instead of $\langle \bar{1}, \bar{k}\rangle$. An element of the form $< 1, j, k >$ will be used as an "index" of the k-th function symbol of arity j in T, which we denote by f_k^j.

Tuples are defined from pairs by $\langle x_1, \ldots, x_{n+1}\rangle = \langle\langle x_1, \ldots, x_n\rangle, x_{n+1}\rangle$. The *members* of a tuple $\langle x_1, \ldots, x_n$ are the x_i for $i = 1, \ldots, n$. When y and w are two tuples of length at least m we write $E_m(y, w)$ for the conjunction of the formulas $E(y_i, w_i)$ for $i \leq m$.

To produce a λ-model we must define an operation λ^*, which takes a variable and an M-term. An M-term (as explained in (1), p. 86 *ff.*), is a term with "parameters from M"; more precisely, a closed term in a language containing a constant c_a for each element a of M. A convenient notation for M-terms is $t[y]$, where y is a tuple of elements of M whose length is the number of free variables of t. This means the following: if $x_1, \ldots, x_n$ are the free variables of t, in order of their occurrence, then $t[y]$ is $t[x_i := c_{y_i}]$. We also need the following notation: $t[y, x]$ where x is a variable, and y is a tuple of elements of M whose length is the number of free variables of t different from x, means the following: if $x_1, \ldots, x_n$

†If one insists on interpreting equality as identity instead of by an equivalence relation, one may use the equivalence classes of E as the elements of the model.

are the free variables of t different from x, in order of their occurrence, then $t[y]$ is $t[x_i := c_{y_i}]$. Note that x may or may not occur free in t.

The operation λ^* is given by

$$\lambda^*(x, t[y, x]) :=< 2, j, \ulcorner t \urcorner, y >$$

where y is a tuple whose length is the number of free variables of $\lambda x.\, t$, and x is the j-th variable, and $\ulcorner t \urcorner$ is the (numeral for the) Gödel number of the closed term t. The definitions of Ap, E, and the interpretation $t[y]_M$ of each M-term $t[y]$ are given in one simultaneous inductive definition. The inductive conditions are as follows. In (iii) and (iv), m is the number of free variables of t.

(i) $E(x, y)$ if $x = y$.
(ii) $E(\lambda^*(x, t[y, x]), \lambda^*(z, t[x := z][y]))$.
(iii) $Ap_M(\lambda^*(x, t[y, x]), r_M) = t[x := r][y]_M$.
(iv) $E(\lambda^*(x, t[y, x]), \lambda^*(x, s[w, x])$ if $E(t[x := a][y]_M, s[x := a][w_M)$, and $E_m(y, w)$ where a is a the first constant not occurring in t or s.
(v) $E(Ap_M(a, b), Ap_M(c, d))$ if $E(a, c)$ and $E(b, d)$.
(vi) $Ap(a, b)_M = Ap_M(a_M, b_M)$.
(vii) $E(f(t[y])_M, f(s[w])_M)$ if $E(t[y]_M, s[w]_M)$, and similarly for several variables x.
(viii) $(\lambda x.\, t[y, x])_M = \lambda^*(x, t[y, x])$.
(ix) $f(x)_M = f_M(x_M)$ and similarly for several variables x.
(x) $(c_a)_M = a$ where c_a is the constant for a.
(xi) $E(a, c)$ if $E(a, b)$ and $E(b, c)$.

Since E and Ap occur only positively in these clauses, this is a legitimate inductive definition. We interpret equality in M as the relation E. For each predicate P of arity n in the language of T, we define a relation $\hat{P}$ on M by

$$\hat{P}(x_1, \ldots, x_n) := M \models P(y_1, \ldots, y_n) \wedge E_n(x, y)$$

and we define $\hat{M} \models P(x_1, \ldots, x_n)$ if and only if $\hat{P}(x_1, \ldots, x_n)$.

We start with the following lemma: if $E(r, q)$ then $E(t[x := r], t[x := q])$ for terms t, q, and r. This is proved by induction on the complexity of the M-term t. When t begins with λ or Ap, the corresponding induction step follows from (iv) and (v). When t begins with a function symbol f, we use (vii). When t is atomic, either it is a constant c_a for an element $a = y_1$ of M, in which case there is nothing to prove, or else it is a variable, in which case we have to prove that $E(r, q)$ from the assumption $E(r, q)$, which is immediate.

We next verify the substitutivity of equality, namely: if $E(r, s)$ and $\hat{M} \models A[x := r]$ then $\hat{M} \models A[x := s]$, where A is a formula of lambda logic with constants for elements of M. We prove this by induction on the complexity of A. Since substitution for free variables commutes with the logical connectives and quantifiers, only the case of atomic A needs a proof. If A is an equality $t = q$, then $A[x := r]$ is $t[x := r] = q[x := r]$ and $A[x := s]$ is $t[x := s] = q[x := s]$. By

the lemma we have $t[x := r] = t[x := s]$ in M, and $q[x := r] = q[x := s]$. Assume $\hat{M} \models A[x := r]$. Then by (xi) we have $\hat{M} \models t[x := s] = q[x := s]$. The remaining case is when A is an atomic formula $P(x)$ (x can be several variables.) This is taken care of by the definition of $\hat{P}$ above.

Because of (ii) and (iii), axioms (α) and (β) are satisfied. Now to verify axioms (ξ). Let $t[y]$ be an M-term, i.e. a term with constants for elements y of M substituted for its free variables. Suppose $\hat{M} \models \forall x(t[y]x = s[y]x)$. Then let c be the first constant not occurring in t or s; we have $\hat{M} \models t[x := c] = s[x := c]$. Then by (iv), we have $\hat{M} \models \lambda x.\, t[y] = \lambda x.\, s[y]$. Hence ($\xi$) holds in M.

We have now proved that $\hat{M}$ is a model of lambda logic. We still must prove it satisfies the theory T in first order logic. This follows from the following lemma: For each M-formula $A(x_1, \ldots, x_n)$ with n parameters from M, we have $\hat{M} \models A(x)$ if and only if there exists y with $E_n(x, y)$ and $M \models A(y)$. To prove the lemma: The case when A is an atomic formula $P(x)$ is true by definition of $\hat{P}$. The case of an atomic formula $t = q$ follows from what has been proved above. The proof is completed by induction on the complexity of A; the quantifiers do not offer any difficulty since the carrier sets of M and $\hat{M}$ are the same. That completes the proof of the theorem.

9. Lambda Completeness for LPT

Completeness of first-order LPT was proved in (3), by reduction of LPT to ordinary first order logic. Incidentally, completeness was also proved for intuitionistic LPT, which does not concern us here.

Here we provide the proof of theorem 9.1 of (4).

Theorem 9.1 (Lambda Completeness Theorem for LPT): *Let T be a consistent theory in partial lambda logic. Then T has a partial lambda model.*

Proof. Again we imitate the Henkin proof of completeness for first order logic. If T does not contain infinitely many constant symbols, we begin by adding them; this does not destroy the consistency of T since in any proof of contradiction, we could replace the new constants by variables not occurring elswhere in the proof. We construct the "canonical structure" M for a theory T. The elements of M are equivalence classes of closed terms of T under the equivalence relation of provable equality: $t \sim r$ iff $T \vdash t = r$. But now, we take only those closed terms t for which T proves $t\downarrow$; that ensures the validity of the axiom $x = x$, and of the axioms $x\downarrow$ for variables x and of the axioms $c\downarrow$ for constants c. Let $[t]$ be the equivalence class of t. We define the interpretations of constants, function symbols, and predicate symbols in M as follows:

$$\begin{aligned} c^M &= [c] \\ f^M([t_1], \ldots, [t_n]) &= [f(t_1, \ldots, t_n)] \\ P^M([t_1], \ldots, [t_n]) &= [P(t_1, \ldots, t_n)] \end{aligned}$$

In this definition, the right sides depend only on the equivalence classes $[t_i]$ (as shown in (6), p. 44). The atomic formula $t\downarrow$ is satisfied in M if and only if T proves $t\downarrow$. It follows by induction on the complexity of the closed first-order term t that if T proves $t\downarrow$ then the interpretation t^M of t in M is the equivalence class $[t]$ of t. The strictness axiom for first order function symbols is used in the induction step, so this does not apply to terms containing Ap unless we have strictness for Ap.

Exactly as in (6) one then verifies that M is a first order model of T in the sense of LPT. Thus the completeness theorem for first order LPT (with strictness) is proved (again).

To turn M into a λ-model, we must define $\lambda^*(x,t)$, where t is an M-term, i.e. a closed term with parameters from M. The "parameters from M" are constants $c_{\lceil}q]$, where $[q]$ is the equivalence class of a closed term q of T. If $[t]$ is the equivalence class of an M-term t, let $[t]^\circ$ be a closed term of T obtained from t by replacing each constant $c_{\lceil}q]$ by some closed term q in the equivalence class $[q]$. Which closed term we select does not affect the equivalence class $[t]^\circ$. In other words, $[t]^\circ$ is a well-defined operation on equivalence classes.

We define $\lambda^*(x,t) = [\lambda x.\,[t]^\circ]$. Because of the axiom $\lambda x.\,t\downarrow$, $\lambda^*(x,t)$ is indeed defined for M-terms t. We must show that the value depends only on the equivalence class $[t]$. Suppose T proves $t = s$. Then by axiom (xi), T proves $\lambda x.\,[t] = \lambda x.\,[s]$.

We verify that the axioms of partial lambda logic hold in M. First, the beta axiom: $Ap(\lambda x.\,t, r) \cong t[x := r]$. It suffices to consider the case when t has only x free. Suppose that the left hand side is defined in M. Its interpretation is the equivalence class of $Ap(\lambda x.\,t, r)$. By axiom (β), the right side is provably equal to the left side, so it is defined in M, and its interpretation is the class of $t[x := r]$. On the other hand if the right hand side is defined in M, then it is provably equal to the left side and so both are defined and equal in M. This verifies axiom (β).

Now for axiom (ξ). Suppose t and s are closed terms defined in M, and $Ap(t,x) \cong Ap(s,x)$ is valid in M. Then for each closed term r such that T proves $r\downarrow$, we have tr provably equal to sr. Since T contains infinitely many constant symbols, we can select a constant c that does not occur in t or s, so $tc = sc$ is provable. Replacing the constant c by a variable in the proof, $tx = sx$ is provable. Hence by axiom (ξ), $\lambda x.\,t = \lambda x.\,s$ is probable, and hence that equation holds in M. In verifying axiom (ξ), it suffices to consider the case when s and t are closed terms. The axiom (α) holds in M since it simply asserts the equality of pairs of provably equal terms. The axiom $\mathbf{T} \neq F$ holds since T does not prove $\mathbf{T} = \mathbf{F}$, because T is consistent. That completes the proof.

References

1. Barendregt, H., *The Lambda Calculus: Its Syntax and Semantics*, Studies in Logic and the Foundations of Mathematics **103**, Elsevier Science Ltd. Revised

edition (October 1984).

2. Beeson, M., *Foundations of Constructive Mathematics*, Springer-Verlag, Berlin Heidelberg New York (1985).

3. Beeson, M., Proving programs and programming proofs, in: Barcan, Marcus, Dorn, and Weingartner (eds.), *Logic, Methodology, and Philosophy of Science VII, proceedings of the International Congress, Salzburg, 1983*, pp. 51-81, North-Holland, Amsterdam (1986).

4. Beeson, M., Lambda Logic, submitted to IJCAR2004

5. Beeson, M., Otter Two System Description, submitted to IJCAR 2004.

6. Shoenfield, J. R., *Mathematical Logic*, Addison-Wesley, Reading, Mass. (1967).