Logical Grammar: Introduction to Typed Lambda Calculus

Carl Pollard

Department of Linguistics
Ohio State University

July 6, 2011

Typed Lambda Calculi: Background

- Developed starting with Church and Curry in 1930's
- Can be viewed model-theoretically (Henkin-Montague perspective) or proof-theoretically (Curry-Howard perspective). For now we focus on the former.
- Underlies higher-order logic (HOL) and functional programming. Widely used in semantics for formalizing theories of meaning (and in LG for formalizing phenogrammar).

What’s in a TLC

- A TLC is specified by giving its *types*, its *terms*, and an *equivalence relation* on the terms.
- There are different kinds of TLCs, depending on what kind of propositional logic its type system is based on.
- The TLC that underlies HOL, called *positive* TLC, is based on *positive intuitionistic propositional logic* (PIPL), which lacks false, negation and disjunction.

Types of Positive TLC

1. There are some *basic* types. For concreteness, here we assume two basic types motivated by NL semantics:

    p (for *propositions*, the kind of meanings expressed by utterances of declarative sentences)
    e (for *entities*, the kind of things that can be meanings of names (just for now assuming a direct reference theory of names))

2. T is a type.
3. If A and B are types, so is $A \wedge B$.
4. If A and B are types, so is $A \rightarrow B$.

Terms of Positive TLC (1/2)

Note: we write '$\vdash a : A$' to mean term a is of type A.

a. There are some *nonlogical constants*. In the typical appplication to NL semantics, these are interpreted as word meanings, e.g.:

$$\vdash \mathsf{fido} : \mathrm{e}$$
$$\vdash \mathsf{bark} : \mathrm{e} \rightarrow \mathrm{p}$$
$$\vdash \mathsf{bite} : \mathrm{e} \rightarrow \mathrm{e} \rightarrow \mathrm{p}$$
$$\vdash \mathsf{give} : \mathrm{e} \rightarrow \mathrm{e} \rightarrow \mathrm{e} \rightarrow \mathrm{p}$$
$$\vdash \mathsf{believe} : \mathrm{e} \rightarrow \mathrm{p} \rightarrow \mathrm{p}$$

b. There is a *logical constant* $\vdash * : \mathrm{T}$. In the application to NL semantics, this is interpreted as the vacuous meaning.

c. For each type A there are variables $\vdash x_i^A : A$ ($i \in \omega$).

Terms of Positive TLC (2/2)

d. If $\vdash a : A$ and $\vdash b : B$, then $\vdash (a, b) : A \wedge B$.
e. If $\vdash a : A \wedge B$, then $\vdash \pi(a) : A$.
f. If $\vdash a : A \wedge B$, then $\vdash \pi'(a) : B$.
g. If $\vdash f : A \rightarrow B$ and $\vdash a : A$, then $\vdash \mathsf{app}(f, a) : B$.
h. If $\vdash x : A$ is a variable and $\vdash b : B$, then $\vdash \lambda_x.b : A \rightarrow B$.

Note: $\mathsf{app}(f, a)$ is usually abbreviated to $(f\ a)$.

Positive TLC Term Equivalences (1/2)

Here t, a, b, p, and f are metavariables ranging over terms.

a. Equivalences for the term constructors:

1. $t \equiv *$ (for t a term of type T)
2. $\pi(a, b) \equiv a$
3. $\pi'(a, b) \equiv b$
4. $(\pi(p), \pi'(p)) \equiv p$

b. Equivalences for the variable binder ('lambda conversion')

(α) $\lambda_x.b \equiv \lambda_y.[y/x]b$
(β) $(\lambda_x.b)\ a \equiv [a/x]b$
(η) $\lambda_x.(f\ x) \equiv f$, provided x is not free in f

Note: The notation '$[a/x]b$' means the term resulting from substitution in b of all free occurrences of $x : A$ by $a : A$. This presupposes no free variable occurences in a become bound as a result of the substitution.

Positive TLC Term Equivalences (2/2)

c. Equivalences of Equational Reasoning

(ρ) $a \equiv a$
(σ) If $a \equiv a'$, then $a' \equiv a$.
(τ) If $a \equiv a'$ and $a' \equiv a''$, then $a \equiv a''$.
(ξ) If $b \equiv b'$, then $\lambda_x.b \equiv \lambda_x.b'$.
(μ) If $f \equiv f'$ and $a \equiv a'$, then $(f\ a) \equiv (f'\ a')$.

The Henkin-Montague Perspective

A **(set-theoretic) interpretation** I of a positive TLC assigns to each type A a set $I(A)$ and to each constant $\vdash a : A$ a member $I(a)$ of $I(A)$, subject to the following constraints:

1. $I(\mathrm{T})$ is a singleton
2. $I(A \wedge B) = I(A) \times I(B)$
3. $I(A \rightarrow B) \subseteq I(A) \rightarrow I(B)$

Note: As in Henkin 1950, the set inclusion in the last clause (3) can be proper, as long as there are enough functions to interpret all functional terms.

Assignments

An **assignment** relative to an interpretation is a function that maps each member of a set of variables to a member of the set that interprets the variable's type.

Extending an Interpretation Relative to an Assignment

Given an assignment α relative to an interpretation I, there is a unique extension of I, denoted by I_α, that assigns interpretations to all terms, such that:

1. for each variable x, $I_\alpha(x) = \alpha(x)$
2. for each constant a, $I_\alpha(a) = I(a)$
3. if $\vdash a : A$ and $\vdash b : B$, then $I_\alpha((a, b))$ is $\langle I_\alpha(a), I_\alpha(b) \rangle$
4. if $\vdash p : A \wedge B$, then $I_\alpha(\pi(p))$ is the first component (= projection onto $I(A)$) of $I_\alpha(p)$; and $I_\alpha(\pi'(p))$ is the second component (= projection onto $I(B)$) of $I_\alpha(p)$
5. if $\vdash f : A \to B$ and $\vdash a : A$, then $I_\alpha((f\ a)) = (I_\alpha(f))(I_\alpha(a))$
6. if $\vdash b : B$, then $I_\alpha(\lambda_{x \in A}.b)$ is the function from $I(A)$ to $I(B)$ that maps each $s \in I(A)$ to $I_\beta(b)$, where β is the assignment that coincides with α except that $\beta(x) = s$.

Observations about Interpretations

- Two terms $\vdash a : A$ and $\vdash b : B$ of positive TLC are term-equivalent iff $A = B$ and, for any intepretation I and any assignment α relative to I, $I_\alpha(a) = I_\alpha(b)$.
- Another way of stating the preceding is to say that term equivalence (viewed as an equational proof system) is sound and complete for the class of set-theoretic interpretations described earlier.
- For any term a, $I_\alpha(a)$ depends only on the restriction of α to the free variables of a.
- In particular, if a is a closed (i.e. has no free variables), then $I_\alpha(a)$ is independent of α so we can simply write $I(a)$.
- Thus, an interpretation for the basic types and constants extends uniquely to all types and all closed terms.

Sequent-Style ND with Proof Terms

- We review a style of ND equipped to analyze not just provability, but also proofs.
- We illustrate how this works for PIPL, starting from the (term-free) sequent-style ND for PIPL already introduced.
- We'll see that in addition to (or at the same time as) being thought of as denoting elements of models, TLC terms can also be thought of as notations for proofs.
- This idea was first articulated by Curry (1934, 1958), then elaborated by Howard (1969 [1980]), Tait (1967), etc..
- Soon, we'll use this kind of ND for phenos and meanings in linguistic derivations.

Preliminary Definitions

1. A (TLC) term is called **closed** if it has no free variables.
2. A closed term is called a **combinator** if it contains no nonlogical constants.
3. A type is said to be **inhabited** if there is a closed term of that type.

Curry-Howard Correspondence (1/2)

- Types are (the same thing as) formulas.
- Type constructors are logical connectives.
- (Equivalence classes of) terms are proofs.
- The free variables of a term are the undischarged hypotheses on which the proof depends.
- The nonlogical constants of a term are the nonlogical axioms used in the proof.
- A type is a theorem iff it is inhabited.
- A type is a pure theorem (requires no nonlogical axioms to prove it) iff it is inhabited by a combinator.

Curry-Howard Correspondence (2/2)

- Application corresponds to Modus Ponens.
- Abstraction corresponds to Hypothetical Proof (discharge of hypothesis).
- Pairing corresponds to Conjunction Introduction.
- Projections correspond to Conjunction Eliminations.
- Identification of free variables corresponds to collapsing of duplicate hypotheses (Contraction).
- Vacuous abstraction corresponds to discharge of a nonexistent hypothesis (Weakening).

Notation for Sequent-Style ND with Proof Terms

Judgments are of the form $\Gamma \vdash a : A$, read 'a is a proof of A with hypotheses Γ', where

1. A is a formula (= type)
2. a is a term (= proof)
3. Γ, the **context** of the judgment, is a set of variable/formula pairs of the form $x : A$, with a distinct variable in each pair.

Axiom Schemas

Hypotheses:

$$x : A \vdash x : A$$

(x a variable of type A)

Nonlogical Axioms:

$$\vdash a : A$$

(a a nonlogical constant of type A)

Logical Axiom:

$$\vdash * : \mathrm{T}$$

Rule Schemas for Implication

$\to$-Elimination or Modus Ponens:

$$\frac{\Gamma \vdash f : A \to B \qquad \Delta \vdash a : A}{\Gamma, \Delta \vdash (f\ a) : B} \to\text{E}$$

This presupposes no variable occurs in both Γ and Δ.

$\to$-Introduction or Hypothetical Proof:

$$\frac{x : A, \Gamma \vdash b : B}{\Gamma \vdash \lambda_x.b : A \to B} \to\text{I}$$

Other Rule Schemas

There are also rules schemas (which we will not need) for:

- pairing/conjunction introduction
- projections/conjunction elimination
- identifying variables/contraction
- useless hypotheses/weakening

For details see e.g. John C. Mitchell and Philip J. Scott (1989) "Typed lambda models and cartesian closed categories", **Contemporary Mathematics** 92:301-316.