Gödel's System T

Pramook Khungurn

December 11, 2019

Gödel's System T is the first logical system of functions such that, if you can determine a type of a function, the function terminates. The system is a restrictive subset of extended lambda calculus.

1 Definitions

1.1 Types

- We define **types** as follows:
  - $\mathbb{N}$ is a type.
  - If α and β are types, then $\alpha \to \beta$ is a type.
- $\mathbb{N}$ is called the **atomic type**,
  and types of the form $\alpha \to \beta$ are called **higher types**.
- Types are typically denoted by σ and τ.
- $\sigma_1 \to \sigma_2 \to \sigma_3 \to \cdots \to \sigma_n$ means $\sigma_1 \to (\sigma_2 \to (\sigma_3 \to \cdots \to \sigma_n)))$
- Stenlund called all the objects we deal with in System T as **computable functionals**.
- An object of type $\mathbb{N}$ is a natural number.
  Natural numbers are computable functionals.
- An object of type $\alpha \to \beta$ is a function
  which takes a computable functional of type α and
  spits out a computable functional of type β.

1.2 Constants

- 0 is a constant of type $\mathbb{N}$.
- s is a constant of type $\mathbb{N} \to \mathbb{N}$,
  representing the successor function.
- For each type τ, there's a constant R^τ of type $\tau \to (\mathbb{N} \to \tau \to \tau) \to \mathbb{N} \to \tau$.
  The R^τ represents the primitive recursive arithmetic combinator.
  (When R^τ is used, however, we will only write it as R.)
- The **numerals** in System T as defined as 0, $s(0)$, $s(s(0))$, $\ldots$.

1.3 Terms

- Each constant of type τ is a term of type τ.
- For each type τ, there exists a countably finite list of variables of type τ, which we will denote by x^τ, y^τ, z^τ, ...
- If a is a term of type τ and x is a variable of type σ, then $\lambda x.\ a$ is a term of type $\sigma \to \tau$ (Abtraction)
- If a is a term of type $\sigma \to \tau$ and b is term of type σ, then $a\ b$ is a term of type τ. (Application)

2 Equality

- There are two equality relations involved.
  - "$=$" is the equality between natural numbers.
  - "$=_i$" is the intensional equality between terms.
- Of course, "$=$" implies "$=_i$".
- Equality can be inferred from a number of axioms:
  - "$=_i$" is reflexive, symmetric, and transitive.
  - If $a =_i b$ and $c =_i d$, then $a\ b =_i b\ d$.
  - If $a =_i b$, then $\lambda x.\ a =_i \lambda x.\ b$.
  - $(\lambda x.\ (a\ x))\ b =_i a\ b$
  - $\lambda x.\ (a\ x) =_i a$
  - $R\ a\ b\ 0 =_i a$
  - $R\ a\ b\ s(t) =_i b\ t\ (R\ a\ b\ t)$
  - "$=$" is reflexive, symmetric, and transitive.
  - If $c = d$ and $a \in \mathbb{N} \to \mathbb{N}$, then $a\ c = a\ d$.
  - If $a = b$, then $(\lambda x.\ a)c = (\lambda x.\ b)c$ where $c \in \mathbb{N}$.
  - If $a, b \in \mathbb{N}$ and $a =_i b$, then $a = b$.
  - If $f\ 0 = a$ and $f\ s(t) = b\ t\ (f\ t)$, then $f\ t = R\ a\ b\ t$. Here, $a \in \mathbb{N}$, $b \in \mathbb{N} \to \mathbb{N} \to \mathbb{N}$, and $t \to \mathbb{N}$.

3 Reduction

- As with other lambda calculus, System T has its rule for reduction of terms.
- Here are the small-step reduction rules for System T.
  - $R\ a\ b\ 0 \to_1 a$
  - $R\ a\ b\ s(t) \to_1 b\ t\ (R\ a\ b\ t)$
  - $(\lambda x.\ a)\ b \to_1 a[b/x]$

- $\lambda x.(a\ x) \to_1 a$

- A term is called **normal** if it does not contain a subterm which can be reduced by one of the above rules.
- We write $a \to_* b$ if there's a finite series of reduction $a \to_1 a_1 \to_1 a_2 \to_1 a_3 \to_1 \cdots \to_1 b$.
  Here, we say that a **reduces to** b.
- A term a is **normalizable** if it reduces to a normal term.
  The normal term is said to be the **normal form** of a.
- We say that two terms a and b are **definitionally equal** if reduces to the same normal term.
- **Theorem 3.1 (Church–Rosser).** *If $a \to_* b$ and $a \to_* c$, then there exists d such that $b \to_* d$ and $c \to_* d$.*
- **Corollary 3.2.** *Definitional equality is an equivalence relation.*
- **Theorem 3.3.** *Two terms a and b are definitionally equal if and only if $a =_i b$.*

  Proof. ($\to$) Observe that the two sides of the reduction rules are intensionally equal to one another. So, definitional equality implies intensional equality.

  ($\leftarrow$) This is done by the induction on the rules of intensional equality. □

4 Computability and Normal Form

- A term a is **strongly normalizable** (SN) if all reduction sequences

$$a \to_1 a_1 \to_1 a_2 \to_1 a_3 \to_1 \cdots$$

  starting from a are finite.
- In other words, a is SN if a is normalizable and each reduction sequence starting with a ends up in the normal form of a.
- There are terms which are normalizable but not SN. For example:

$$(\lambda x.y)\ ((\lambda x.\ x\ x)\ (\lambda x.\ x\ x))$$

- The main theorem is that all typed terms in System T are SN.
- The proof has three steps.
  - Define what it means for a term of the *computable*.
  - Show that computable terms are SN.
  - Show that all typed terms in System T are computable.
- We call a term a **computable** if it satisfies one of the following rules:
  - If $a \in \mathbb{N}$, then a is computable if it is SN.
  - If a has type $\tau \to \sigma$, then a is computable
    if $(a\ b)$ is computable for all comptuble terms b of type τ.
- We can state the second rule in another way:

A term a is computable if $a\ a_1\ a_2\ \cdots\ a_n$ is computable for all computable terms $a_1, a_2, \ldots,$ a_n such that $a\ a_1\ a_2\ \cdots\ a_n \in \mathbb{N}$.

- **Lemma 4.1.** *If $a \to_* b$ and a is computable, then b is computable.*

  Proof. The lemma follows immediately if $a \in \mathbb{N}$. If a is of higher type, then it follows from the other form of the second rule above. □

- **Lemma 4.2.** *For any type τ, a computable term of type τ is SN.*

  Proof. The proof is by structural induction on the type τ.

  The first case is when τ is $\mathbb{N}$. We have all computable terms are SN by definition.

  The second case is when τ is $\alpha \to \beta$. Let a be a computable term of type τ. Then, we have that for all computable term b of type α, we have that $a\ b$ is a computable term of type β. By induction hypothesis, we have that a and $a\ b$ are SN. Consider any reduction sequence

$$a \to_1 a_1 \to a_2 \to a_3 \to \cdots .$$

  We have that it generates the corresponding sequence

$$a\ b \to_1 a_1\ b \to a_2\ b \to a_3\ b \to \cdots .$$

  Since the sequence sequence terminates, the first one must terminate as well. Therefore, a is normalizable, and since all reduction sequence terminates a is SN. □

- **Lemma 4.3.** *The constants* 0 *and s are computable.*

  Proof. 0 is a normal term, so it is computable.

  Let a be a computable term of type $\mathbb{N}$. We have that a is SN. Then, $s(a)$ also SN because all the reduction done to $s(a)$ must be done to a. Therefore, $s(a)$ is computable. Hence, we have that s is also computable. □

- **Lemma 4.4.** *If $a[b/x]$ is computable, then $(\lambda x.\ a)\ b$ is also computable.*

  Proof. Let $a_1, a_2, \ldots, a_n$ be computable terms such that $a[b/x]\ a_1\ a_2\ \cdots\ a_n$ is a term of type $\mathbb{N}$. It follows that $a[b/x]\ a_1\ a_2\ \cdots\ a_n$ is computable and SN. We can prove the lemma by showing that $(\lambda x.\ a)\ b\ a_1\ a_2\ \cdots\ a_n$ is SN.

  Suppose by way of proof by contradiction that there is a infinite sequence of reduction in

$$(\lambda x.\ a)\ b\ a_1\ a_2\ \cdots\ a_n.$$

  We know that thisreudction must be an infinite reduction of $(\lambda x.\ a)$. Suppose such a sequence is

$$\lambda x.\ a \to_1 \lambda x.\ a' \to_1 \lambda x.\ a'' \to_1 \cdots .$$

  This sequence would yield

$$a[b/x] \to_1 a'[b/x] \to_1 a''[b/x] \to_1 \cdots$$

  which is an infinite reduction of $a[b/x]$, which is impossible. We have a contradiction. □

- **Lemma 4.5.** *The constant R^τ for all τ is computable.*

Proof. It is sufficient to prove that $R\ a\ b\ c$ is computable for all computable a, b, and c of the appropriate types. We do so by the number of s in c.

Let $a_1,\ a_2,\ \ldots,\ a_n$ be computable terms such that $R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n$ is of type $\mathbb{N}$.

If c does not reduce to 0 or $s(t)$ for some $t \in \mathbb{N}$, then we cannot use the axiom for R to reduce $R\ a\ b\ c$. Thus, all the reduction done to $R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n$ must be done in $a,\ b,\ c,\ a_1,\ a_2,\ \ldots,\ a_n$. Since all these terms are computable and thus SN, then $R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n$ is SN and thus computable.

If c reduces to 0, then we have that

$$R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n \to_* R\ a'\ b'\ 0\ a_1'\ a_2'\ \cdots\ a_n' \to_1 a'\ a_1'\ a_2'\ \cdots\ a_n'.$$

For some $a', b', a_1', a_2', \ldots a_n'$ that $a, b, a_1, a_2, \ldots, a_n$ reduce to, respectively. Since $a\ a_1\ a_2\ \cdots\ a_n \to a'\ a_1'\ a_2'\ \cdots\ a_n'$, it must be the case that $a'\ a_1'\ a_2'\ \cdots\ a_n'$ is SN. Therefore, $R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n$ is SN as well.

If c reduces to $s(t)$, we have that $s(t)$ is computable, thus SN. This means that t is SN, thus computable. It follows by induction hypothesis that $(R\ a'\ b'\ t)$ is computable for any computable a' and b'. Consider the reduction of $R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n$, we have that it must be of the form

$$R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n \to_* R\ a'\ b'\ s(t)\ a_1'\ a_2'\ \cdots\ a_n' \to_1 b'\ t\ (R\ a'\ b'\ t)\ a_1'\ a_2'\ \cdots\ a_n'.$$

Since all the terms in the last expression is computable, we have that the term is SN. Therefore, $R\ a\ b\ c\ a_1\ a_2\ \cdots\ a_n$ is SN, that thus computable too. □

- **Lemma 4.6.** *If a is a term with free variable $x_1,\ x_2,\ \cdots,\ x_n$ and $b_1, b_2, \ldots, b_n$ are computable terms with the same types as $x_1,\ x_2,\ \ldots,\ x_n$, respectively, then $a[b_1/x_1][b_2/x_2]\cdots[b_n/x_n]$ is computable.*

  Proof. We prove this by induction on the structure of a. For convenience let

  $$a' := a[b_1/x_1][b_2/x_2]\cdots[b_n/x_n].$$

  If a is one of the free variables, say x_i, then $a' = b_i$, which is computable.

  If a is $a_1\ a_2$, then a' is $a_1'\ a_2'$. a_1' and a_2' are computable by the induction hypothesis. Therefore, $a_1'\ a_2'$ is computable too.

  If a is $\lambda x.\ a_1$, then by induction hypothesis $a_1'[b/x]$ is computable for all computable b. This implies that $(\lambda x.a_1)\ b$ is computable for all computable b, which means that $\lambda x.\ a_1$ is computable. □

- **Theorem 4.7.** *All terms are computable, and thus SN.*

  Proof. Use the last four lemmas with structural induction on the terms. □