Lambda Definability

October 23, 2007
Goals

Most information taken from "Lambda Calculi with Types" by Henk Barendregt.

Two Goals:

1. All computable functions are $\lambda$-definable.
2. Undecidability of $\lambda$-Calculus.
Church numerals

Definition.

1. \( F^n(M) \) with \( n \in \mathbb{N} \) (the set of natural numbers) and \( F, M \in \Lambda \), is defined inductively as follows:

\[
F^0(M) \equiv M; \\
F^{n+1}M \equiv F(F^n(M)).
\]

2. The *Church numerals* \( c_0, c_1, c_2, \ldots \) are defined by

\[ c_n \equiv \lambda f. x. f^n(x). \]
Functions represented in Church’s numerals

Functions plus, times and exponentiation on $\mathbb{N}$ can be represented in the $\lambda$-calculus using Church’s numerals.

**Definition.**

\[
A_+ \equiv \lambda xypq. xp(ypq);
\]
\[
A_* \equiv \lambda xyz . x(yz);
\]
\[
A_{\text{exp}} \equiv \lambda xy . yx.
\]

We have for all $n, m \in \mathbb{N}$

1. $A_+ c_n c_m = c_{n+m}$.
2. $A_* c_n c_m = c_{n*m}$.
3. $A_{\text{exp}} c_n c_m = c_{n^m}$, $(m \geq 1)$. 
Functions represented in Church’s numerals

Proof.

\[ A_+ c_n c_m \equiv (\lambda ypq.xp(ypq))c_n c_m \]
\[ = \lambda pq.c_n p(c_mpq) \]
\[ = \lambda pq.c_n p(p^m q) \]
\[ = \lambda pq.p^n(p^m q) \]
\[ = \lambda pq.p^{n+m} q \]
\[ \alpha \equiv \lambda fx.f^{n+m} x \]
\[ \equiv c_{n+m} \]
Boolean functions represented in $\lambda$ – calculus

Boolean truth values and a conditional can be represented in the $\lambda$ – calculus.

**Definition (Booleans, conditional).**

1. $\text{true} \equiv \lambda xy. x$, $\text{false} \equiv \lambda xy. y$.
2. $\text{if } B \text{ then } P \text{ else } Q \equiv BPQ$

Indeed, $\text{true}PQ = P$ and $\text{false}PQ = Q$. 
Boolean functions represented in $\lambda$ – calculus

Boolean truth values and a conditional can be represented in the $\lambda$ – calculus.

**Definition (Booleans, conditional).**

1. **true** $\equiv \lambda xy. x$, **false** $\equiv \lambda xy. y$.
2. **if** $B$ **then** $P$ **else** $Q$ $\equiv BPQ$

Indeed, **true**$PQ$ $=$ $P$ and **false**$PQ$ $=$ $Q$.

**Definition (Pairing).** For $M, N \in \Lambda$ write

$$[M, N] \equiv \lambda z. zMN. \quad (\equiv \lambda z. \text{if } z \text{ then } M \text{ else } N)$$

Then

$$[M, N]\text{true} = M$$

$$[M, N]\text{false} = N$$
and hence \([M, N]\) can serve as an ordered pair. We use this pairing construction for defining primitive recursion later.

**Definition.**

1. A numeric function is a map \(f : \mathbb{N}^k \to \mathbb{N}\) for some \(k\).
2. A numeric function \(f\) with \(p\) arguments is called \(\lambda\)-definable if one has for some combinator \(F\)

\[
Fc_{n_1}...c_{n_k} = c_f(n_1,...,n_k) \quad (1)
\]

for all \((n_1, ..., n_k) \in \mathbb{N}\). If (1) holds, then \(f\) is said to be \(\lambda\) — *defined* by \(F\).
representable ⇔ computable

In the following we prove that every computable function can be represented by a $\lambda$-term.

**Definition.**
A function is computable iff it can be computed by a Turing machine.

**Remark.**
According to the Church-Turing thesis this notion of computability coincide with the informal notion of computability.

**Theorem.**
A function $f: \mathbb{N}^k \to \mathbb{N}$ is computable iff it is representable by a $\lambda$-term.
representable $\Rightarrow$ computable

Proof. "representable $\Rightarrow$ computable":
Lemma (consequence of the Church-Rosser Theorem).

\[ M = c_k, M = c_{k'} \Rightarrow k = k' \]

Assume $M$ represented by $f$:
We have inputs $n_1, \ldots, n_k$.
Do: check all terms $N$, $Mc_{n_1} \cdots c_{n_k} = N$.
Stop: if $N = c_m$.
Output: $m$

Proposition. given $n_1, \ldots, n_k$.

1. Algorithm terminates with some output $m$.
2. $f(n_1, \ldots, n_k) = m$. 

Lambda Definability
representable ⇒ computable

Proof.

1. Since $M$ is represented by $f$, we have:

$$Mc_{n_1}...c_{n_k} = cf(n_1,...,n_k)$$

Hence, there is a number $m$ such that:

$$Mc_{n_1}...c_{n_k} = c_m$$

Therefore, the algorithm terminate.

2. 

$$
\begin{align*}
Mc_{n_1}...c_{n_k} &= c_m \\
Mc_{n_1}...c_{n_k} &= cf(n_1,...,n_k)
\end{align*}
\Rightarrow f(n_1, ..., n_k) = m$$
computable \Rightarrow \text{representable}

\textbf{Proof}. ”computable \Rightarrow \text{representable”}.

\textbf{Definition}. The $\mu$-recursive functions are total functions that take finite tuples of natural numbers and return a single natural number. They are the smallest class of total functions that includes the initial functions and is closed under composition, primitive recursion and minimalization.

\textbf{Proposition}. A function is computable iff it is $\mu$-recursive.

\textbf{Definition}. $\mu$-recursive functions are precisely the functions that can be computed by Turing machines.

Therefore, in order to show that all computable functions are $\lambda$-representable. It suffices to show that the initial function are $\lambda$-representable and the $\lambda$-representable functions are closed under composition, primitive recursion, and minimalization.
The initial functions are $\lambda$-definable

**Lemma.** The initial functions are $\lambda$-definable.

**Proof.** The initial functions are the numeric functions $U^i_r, S^+, Z$ defined by:

\[
U^i_r(x_1, ..., x_r) = x_i, \quad (1 \leq i \leq r)
\]
\[
S^+(n) = n + 1
\]
\[
Z(n) = 0
\]

(Be careful! the function names is different from the representation term name) Define the terms as following:

\[
U^i_r \equiv \lambda x_1 \cdots x_r.x_i
\]
\[
S^+ \equiv \lambda xyz.y(xyz)
\]
\[
Z \equiv \lambda x.c_0
\]
The initial functions are $\lambda$-definable

We easily see

$$U_r^i(c_{n_1} \ldots c_{n_r}) \equiv (\lambda x_1 \cdots x_r.x_i)(c_{n_1} \ldots c_{n_r});$$
$$= c_{n_i}$$

$$S^+ c_n \equiv (\lambda xyz.y(xyz))c_n$$
$$= \lambda yz.y(c_n yz)$$
$$= \lambda yz.y(y^n z)$$
$$= \lambda yz.y^{n+1} z$$
$$= c_{n+1}$$

$$Z c_n \equiv (\lambda x.c_0)c_n$$
$$= c_0$$
The $\lambda$-definable functions are closed under composition

**Definition.** Composition, also called substitution takes a function $g(n_1,\ldots,n_m)$ and functions $h_i(n_1,\ldots,n_k)$ for each $i$ with $1 \leq i \leq m$. and returns the function $f : \mathbb{N}^k \rightarrow \mathbb{N}$

$$f(n_1,\ldots,n_k) = g(h_1(n_1,\ldots,n_k),\ldots,h_m(n_1,\ldots,n_k))$$

**Lemma.** The $\lambda$-definable functions are closed under composition.

**Proof.** Let $g$, $h_1$, ..., $h_m$ be $\lambda$-defined by $G$, $H_1$, ..., $H_m$ respectively. This means:

\[
\begin{align*}
g : \mathbb{N}^m &\rightarrow \mathbb{N} \\
gc_{n_1}\ldots c_{n_m} &= c_g(n_1,\ldots,n_m) \\
h_1 : \mathbb{N}^k &\rightarrow \mathbb{N} \\
h_1c_{n_1}\ldots c_{n_k} &= c_{h_1}(n_1,\ldots,n_k) \\
\ldots \\
h_m : \mathbb{N}^k &\rightarrow \mathbb{N} \\
h_mc_{n_1}\ldots c_{n_k} &= c_{h_m}(n_1,\ldots,n_k)
\end{align*}
\]
The $\lambda$-definable functions are closed under composition

Then

$$f(n_1, \ldots, n_k) = g(h_1(n_1, \ldots, n_k), \ldots, h_m(n_1, \ldots, n_k))$$

is $\lambda$-defined by

$$F \equiv \lambda\bar{x}. G(H_1\bar{x})\ldots(H_m\bar{x})$$

We have to show

$$Fc_{n_1} \ldots c_{n_k} = cf(n_1, \ldots, n_k)$$

$$Fc_{n_1} \ldots c_{n_k} = G(H_1c_{n_1} \ldots c_{n_k})\ldots(H_mc_{n_1} \ldots c_{n_k})$$

$$= Gc_{h_1(n_1, \ldots, n_k)} \ldots c_{h_m(n_1, \ldots, n_k)}$$

$$= cg(h_1(n_1, \ldots, n_k), \ldots, h_m(n_1, \ldots, n_k))$$

$$= cf(n_1, \ldots, n_k)$$
The $\lambda$-definable functions are closed under primitive recursion

**Definition.** A function $f$ is defined by primitive recursive. From function $g$ and $h$ if

\[
\begin{align*}
    f(0, \vec{n}) &= g(\vec{n}) \\
    f(k + 1, \vec{n}) &= h(f(k, \vec{n}), k, n)
\end{align*}
\]

**Lemma.** The $\lambda$-definable functions are closed under primitive recursion.
The $\lambda$-definable functions are closed under primitive recursion

**Proof.** For simplicity, we assume there is no parameters $\vec{n}$ i.e

$$f(0) = g$$

$$f(k + 1) = h(f(k), k)$$

Let $f$ be defined as above. Assume $g \in \mathbb{N}$ and $h : \mathbb{N}^2 \to \mathbb{N}$ are $\lambda$-defined by $G$ and $H$.

$$G = c_g$$

$$Hc_mc_n = c_h(m, k)$$

Hence, we have $G = c_{f(0)}$
The \( \lambda \)-definable functions are closed under primitive recursion

Consider, \( T \equiv \lambda p.[S^+(p \text{true}), H(p \text{false})(p \text{true})] \)

Then, for all \( k \) one has:

\[
T([c_k, c_{f(k)}]) = [S^+ c_k, Hc_{f(k)} c_k] \\
= [c_{k+1}, c_{h(f(k))} c_k] \\
= [c_{k+1}, c_{f(k+1)}]
\]

By induction on \( k \) it follows that.

\[
[c_k, c_{f(k)}] = T^k[c_0, c_{f(0)}] \\
c_{f(k)} = T^k[c_0, c_{f(0)}]\text{false} \\
c_{f(k)} = c_k T[c_0, c_{f(0)}]\text{false}
\]
The $\lambda$-definable functions are closed under primitive recursion

Let $f$ be $\lambda$-defined by.

\[
F \equiv \lambda x. x T[c_0, G]\text{false}
\]

\[
Fc_k = c_k T[c_0, G]\text{false}
\]

\[
= c_f(k)
\]
The $\lambda$-definable functions are closed under minimalization

**Lemma.** The $\lambda$-definable functions are closed under minimalization.

**Proof.** Let $g$ be such that $\forall \vec{n} \exists m g(\vec{n}, m) = 0$, $f$ be defined by

$$f(\vec{n}) = \mu m[g(\vec{n}, m) = 0]$$

where $\vec{n} = n_1...n_k$ and $g$ is $\lambda$-defined by $G$.

**Definition.** $\text{zero} \equiv \lambda n. n(\text{true false})\text{true}$

Then

$$\text{zero } c_0 = c_0(\text{true false})\text{true}$$
$$= \lambda f x. x(\text{true false})\text{true}$$
$$= \text{true}$$
The \( \lambda \)-definable functions are closed under minimalization

\[
\text{zero } c_{n+1} = c_{n+1} \langle \text{true false} \rangle \text{true} \\
= \lambda fx. f^{n+1}x \langle \text{true false} \rangle \text{true} \\
= \langle \text{true false} \rangle ^{n+1} \text{true} \\
= \text{true false}(\langle \text{true false} \rangle ^{n} \text{true}) \\
= \text{false}
\]

Recall the Corollary, there is a term \( H \) such that

\[
H\vec{n}y = \text{if } (\text{zero}(G\vec{n}y)) \text{ then } y \text{ else } H\vec{n}(S^{+}y)
\]

Set \( F = \lambda \vec{x}. H\vec{x}c_{0}, \ c_{\vec{n}} = c_{n_{1}} \ldots c_{n_{k}}. \)
The $\lambda$-definable functions are closed under minimalization

Then $F$ $\lambda$-defines $f$:

$$Fc_{\vec{n}} = Hc_{\vec{n}}c_0$$

$$= c_0, \quad \text{if } Gc_{\vec{n}}c_0 = c_0,$$

$$= Hc_{\vec{n}}c_1 \quad \text{else;}$$

$$= c_1, \quad \text{if } Gc_{\vec{n}}c_1 = c_0,$$

$$= Hc_{\vec{n}}c_2 \quad \text{else;}$$

$$= c_2, \quad \text{if ...}$$

$$= ...$$
Undecidability of $\lambda$-Calculus

Definition.

1. **Notation.** $\nu^0 = \nu$; $\nu^{(n+1)} = \nu^{(n)'}$.

2. Let $<,>: \mathbb{N}^2 \to \mathbb{N}$ be a computable pairing function. If $(n_1, m_1) \neq (n_2, m_2) \Rightarrow < n_1, m_1 > \neq < n_2, m_2 >$

   \[ \forall k \exists (n, m) (< n, m > = k) \quad \text{i.e.} \quad \{ < n, m > | n, m \in \mathbb{N} \} = \mathbb{N} \]

   \[ < n, m > = \frac{(n + m)(n + m + 1)}{2} + n \]

3. **Notation**

   $\lceil M \rceil := C_\#(M)$
Undecidability of $\lambda$-Calculus

**Definition.** Let $\mathcal{A} \subseteq \Lambda$.

1. $\mathcal{A}$ is closed under $=$ if

$$M \in \mathcal{A}, \lambda \vdash M = N \Rightarrow N \in \mathcal{A}$$

2. $\mathcal{A}$ is non-trivial if $\mathcal{A} \neq \emptyset$ and $\mathcal{A} \neq \Lambda$

3. $\mathcal{A}$ is decidable if $\#\mathcal{A} := \{\#M | M \in \mathcal{A}\}$ is decidable.

**Theorem.** Let $\mathcal{A} \subseteq \Lambda$ be non-trivial and closed under $=$. Then $\mathcal{A}$ is not decidable.
Undecidability of $\lambda$-Calculus

Proof.

Definition.

$$B = \{ M | \neg \neg M \in A \} \quad (b)$$

Suppose $A$ is decidable, then we have $B$ is decidable. $B$ decidable $\Rightarrow \#B \subseteq \mathbb{N}$ is decidable.

$$f(n) = \begin{cases} 
0 & \text{if } n \in \#B \\
1 & \text{if } n \notin \#B 
\end{cases}$$

is computable. By the **Representability Theorem**, there is $\lambda$ term $F$ s.t. $\forall n : Fc_n = c_{f(n)}$.
Undecidability of $\lambda$-Calculus

\[ M \in B \Rightarrow \#M \in \#B \]
\[ \Rightarrow f(\#M) = 0 \]
\[ \Rightarrow Fc_{\#M} = c_0 \]
\[ \Rightarrow F\downarrow M\downarrow = c_0 \]

\[ M \notin B \Rightarrow \#M \notin B \]
\[ \Rightarrow f(\#M) = 1 \]
\[ \Rightarrow Fc_{\#M} = c_1 \]
\[ \Rightarrow F\downarrow M\downarrow = c_1 \]
Undecidability of $\lambda$-Calculus

Let $M_0 \in \mathcal{A}$, $M_1 \notin \mathcal{A}$. We easily can find a $G \in \Lambda$ such that,

\[ M \in \mathcal{B} \Rightarrow G \downarrow M \downarrow = M_1 \notin \mathcal{A} \quad (1) \]

\[ M \notin \mathcal{B} \Rightarrow G \downarrow M \downarrow = M_0 \in \mathcal{A} \quad (2) \]

$G$ can be defined as follows:

\[ G = \lambda x. \text{if zero}(Fx) \text{ then } M_1 \text{ else } M_0 \]

Where $\text{zero} \equiv \lambda n. n(\text{true} \ \text{false}) \text{true}$. 

Now it follows:

\[ G \in \mathcal{B} \Rightarrow (1) \ G \downarrow G \downarrow = M_1 \notin \mathcal{A} \Rightarrow (b) \ G \notin \mathcal{B}, \]

\[ G \notin \mathcal{B} \Rightarrow (2) \ G \downarrow G \downarrow = M_0 \in \mathcal{A} \Rightarrow (b) \ G \in \mathcal{B}, \]
Undecidability of $\lambda$-Calculus

Let us have a look an application of the above Theorem. The Set

$$\mathcal{A} := \{ M | M = \text{true} \}$$

is not decidable.

**Proof.**
(1) The set is non-trivial.

$$\mathcal{A} \neq \emptyset \quad \text{because,} \quad \text{true} \in \mathcal{A}$$

$$\mathcal{A} \neq \Lambda \quad \text{because,} \quad \text{false} \notin \mathcal{A}$$

$$\text{true} \neq \text{false}$$

$$\lambda xy. x \neq \lambda xy. y \quad (\text{by Church – Rosser theorem})$$
Undecidability of $\lambda$-Calculus

(2) The set is closed under $=$, Let $M \in A$ and $M = N$

\[
M \in A \Rightarrow M = \text{true} \\
M = N \Rightarrow N = \text{true}
\]

By the above Theorem we know the set $A := \{M | M = \text{true}\}$ is not decidable.