Lambda Calculus and Types

Mathematical Preliminaries and Untyped Arithmetic Expression

陳亮廷 Chen, Liang-Ting

2018 邏輯、語言與計算暑期研習營
Formosan Summer School on
Logic, Language, and Computation

Swansea University, UK

Introduction

Introduction

Why does theory of programming languages matter?

1. A rigorous approach yields unambiguous and yet neat definition of a language.

   The Definition of Standard ML: 116 pp.[MTHM97]

   vs.

   The Standard of C++ Programming Language: 1605 pp.

2. A formally verified compiler is possible:

   CakeML: A verified implementation of Standard ML.

   vs.

   CompCert: A formally verified compiler of ...C99

   (what is exactly C++, anyway?)

3. The use of static type prevents mistakes in the early stage of developments.

Let's start with Calculus ...

How to solve the following equation?

$$\int \sin(x)\,\mathrm{d}x = ?$$

$$\cdots = \int \sin(x)\,\mathrm{d}x$$

and it is mathematically correct! Aren't $x = x$ for any x? You are actually asked to evaluate an expression to its simplest form instead.

$$\int \sin(x)\,\mathrm{d}x \longrightarrow -\cos(x) + C$$

But, what should $\longrightarrow$ be?

Mathematical Preliminaries

Predicate, Relation

Definition 1

A *n-ary relation* R over sets $X_1, X_2, \ldots, X_n$ is a set

$$R \subseteq X_1 \times X_2 \cdots \times X_n.$$

Definition 2

1. A 1-ary relation $P \subseteq X$ is a *predicate* on X.
2. A 2-ary relation $S \subseteq X \times Y$ is a *binary relation* on X and Y.

For readability, $P(x)$ stands for $x \in P$. Also we use 'infix' or 'mixfix' notations for relation.

$$t \longrightarrow u \qquad \text{and} \qquad \Gamma \vdash x : \tau$$

stands for relations $(t, u) \in \longrightarrow$ and $(\Gamma, t, \tau) \in (\_ \vdash \_ : \_)$.

A bit more about relations

A binary relation $R \subseteq X \times X$ is ...

1. reflexive if $x \mathrel{R} x$ for any $x \in X$;
2. symmetric if $x_1 \mathrel{R} x_2$ implies $x_2 \mathrel{R} x_1$
3. transitive if $x_1 \mathrel{R} x_2$ and $x_2 \mathrel{R} x_3$ implies $x_1 \mathrel{R} x_3$;
4. an equivalence relation if it satisfies all of above conditions.

A binary relation $f \subseteq X \times Y$ is ...

1. functional or a partial function if

$$x \mathrel{f} y \quad \text{and} \quad x \mathrel{f} y' \quad \text{implies} \quad y = y'$$

2. a (total) function if it is functional and

$$\forall x \in X.\, \exists y \in Y.\, x \mathrel{f} y$$

Induction and Recursion

Definition 3 (Arithmetic expressions, by grammar)

The set of arithmetic expressions is defined by

$$t := \texttt{0} \mid \texttt{succ}\, t \mid \texttt{add}\, t\, t$$

`t` is a *metavariable* to be replaced by an expression. E.g., `add 0 (succ 0)` is an expression but `succ` alone is not.

Definition 4 (Arithetic expressions, inductively)

The set $\mathcal{T}$ for arithmetic expression is the least set satisfying

1. $\texttt{0} \in \mathcal{T}$,
2. $\texttt{succ}\, t \in \mathcal{T}$ if $t \in \mathcal{T}$,
3. $\texttt{add}\, t\, u \in \mathcal{T}$ if $t, u \in \mathcal{T}$.

Judgement and Inference Rules

A judgement is just a predicate and a rule of inference is an implication in a specific form, possibly with a name.

Judgement $0 \in \mathcal{T}$, t is of type τ, ...

Inference rules

$$\frac{}{\mathtt{0} \in \mathcal{T}} \qquad \frac{t \in \mathcal{T}}{\mathtt{succ}\, t \in \mathcal{T}} \qquad \frac{t \in \mathcal{T} \qquad u \in \mathcal{T}}{\mathtt{add}\, t\, u \in \mathcal{T}}$$

The set of arithmetic expressions is specified by the rules. This form of definition is widely used and we will use this form too.

Structural Induction

The Induction Principle holds not only for $\mathbb{N}$ but also for any structure defined inductively.

Theorem 5 (Induction Principle on arithmetic expressions)

Given a predicate P, if we can prove that

$$\frac{}{P(\mathtt{0})} \qquad \frac{P(t)}{P(\mathtt{succ}\, t)} \qquad \frac{P(t) \qquad P(u)}{P(\mathtt{add}\, t\, u)}$$

then $P(t)$ for any $t \in \mathcal{T}$. In other words, $\mathcal{T} \subseteq P$.

Recall that $\mathcal{T}$ is by definition the smallest set containing `0` and closed under `succ` and `add`.

Theorem 6 (Recursion on arithmetic expressions)

Given functions

1. $f_0 \colon \{*\} \to S$,
2. $f_{\mathtt{succ}} \colon S \to S$,
3. $f_{\mathtt{add}} \colon S \times S \to S$,

there exists a unique function $f \colon \mathcal{T} \to S$ *such that*

1. $f(\mathtt{0}) = f_0(*)$
2. $f(\mathtt{succ}\, n) = f_{\mathtt{succ}}(f(n))$
3. $f(\mathtt{add}\, n\, m) = f_{\mathtt{add}}(f(n), f(m))$.

Proof of Uniqueness.

By structural induction on $\mathcal{T}$. $\square$

A glimpse of denotational semantics

Example 7

A recursion from our arithmetic expressions to natural numbers can be given by

$$\begin{aligned}
[\![-]\!] &: \mathcal{T} \to \mathbb{N} \\
[\![\mathtt{0}]\!] &= 0 \\
[\![\mathtt{succ}\, t]\!] &= 1 + [\![t]\!] \\
[\![\mathtt{add}\, t\, u]\!] &= [\![t]\!] + [\![u]\!]
\end{aligned}$$

which stipulates the *denotational semantics* of our arithmetic expressions.

The subject of denotational semantics is left for self-study. For interested readers, see [Sco76, Str06]

Operational Semantics

Reduction Relation

Instead of giving the *meaning* of expressions in other languages, arithmetic expressions can be computed to other expressions.

Define a binary relation $\longrightarrow$ between arithmetic expressions by

$$\frac{t_1 \longrightarrow t_2}{\mathtt{succ}\, t_1 \longrightarrow \mathtt{succ}\, t_2}\ (\rightarrow\text{-}\mathtt{succ})$$

$$\frac{t_1 \longrightarrow t_2 \qquad t_1 \neq \mathtt{succ}\, t}{\mathtt{add}\, t_1\, u \longrightarrow \mathtt{add}\, t_2\, u}\ (\rightarrow\text{-}\mathtt{add})$$

$$\frac{}{\mathtt{add}\, 0\, u \longrightarrow u}\ (\rightarrow\text{-}\mathtt{add0})$$

$$\frac{}{\mathtt{add}\, (\mathtt{succ}\, t)\, u \longrightarrow \mathtt{succ}\, (\mathtt{add}\, t\, u)}\ (\rightarrow\text{-}\mathtt{add}\mathrm{succ})$$

Some Reductions

`add (succ 0) (succ (succ 0)`

`succ (add (succ 0) (succ 0))`

Values, Normal forms

Definition 8

The set of values for arithmetic expression is defined by

$$\frac{}{\mathtt{0} \in \mathsf{Val}} \qquad\qquad \frac{t \in \mathsf{Val}}{\mathtt{succ}\, t \in \mathsf{Val}}$$

In this case, a value is simply a numeral.

Definition 9

An expression is in normal form if it cannot be reduced further, i.e.

$$\neg(\exists t' \in \mathcal{T} .\, t \longrightarrow t')$$

Theorem 10

Every value is in normal form.

Determinacy

The reduction relation is deterministic:

Theorem 11

Suppose that $t \in \mathcal{T}$ is an expression. If $t \longrightarrow u$ and $t \longrightarrow u'$, then $u = u'$. That is, $\longrightarrow$ is functional.

Proof.

By structural induction on the reduction relation $\longrightarrow$ (not $\mathcal{T}$).

$\square$

Multi-Step Reduction, Transitive Closure

$t \longrightarrow u$ prescribes how t reduces to u in one step, so it is a one-step reduction.

Definition 12

The *transitive and reflexive closure* R^* of a binary relation R is

$$\frac{}{t \; R^* \; t} \qquad\qquad \frac{t \; R \; u \qquad u \; R^* \; v}{t \; R^* \; v}$$

In particular, $\longrightarrow^*$ is the multi-step reduction.

Theorem 13 (Uniqueness of normal forms)

If $t \longrightarrow^* u$ and $t \longrightarrow^* u'$ where u and u' are in normal form, then $u = u'$.

Proof.

By induction. □

Homework

1. Finish the uniqueness proof of Theorem 6.
2. Finish the proof of Theorem 11.
3. Show that the reflexive and transitive closure of any relation R is reflexive and transitive.
4. Define Boolean expressions as follows:

$$\frac{}{\texttt{true} \in \mathcal{T}_{\mathbb{B}}} \qquad \frac{t_1 \in \mathcal{T}_{\mathbb{B}} \qquad t_2 \in \mathcal{T}_{\mathbb{B}} \qquad t_3 \in \mathcal{T}_{\mathbb{B}}}{\texttt{if}\; t_1 \;\texttt{then}\; t_2 \;\texttt{else}\; t_3 \in \mathcal{T}_{\mathbb{B}}}$$

$$\frac{}{\texttt{false} \in \mathcal{T}_{\mathbb{B}}}$$

where values are `true` and `false`. Please give a reduction relation satisfying determinacy and that expressions are in normal forms if and only if they are values.

Acknowledgement

I am grateful to 游書泓 (Shu-Hung You), Chia-An Yu, and Yu-Hsi Chiang for their suggestions and corrections.

References i

- Robert Milner, Mads Tofte, Robert Harper, and David MacQueen, *The definition of standard ml (revised)*, 1997.
- Dana Scott, *Data types as lattices*, SIAM J. Comput. **5** (1976), no. 3, 522–587.
- Thomas Streicher, *Domain-theoretic foundations of functional programming*, World Scientific, December 2006.