TaPL Seminar

3.1 - 3.4

Kosuke Kiuchi

March 20th, 2023

3 Untyped Arithmetic Expressions

3 Untyped Arithmetic Expressions

3.1 Introduction

Language

```
t ::= true
      false
      if t then t else t
      0
      succ t
      pred t
      iszero t
```

`t` : *metavariable*

Symbol `t` in the right-hand sides is called a *metavariable*.

It is a place-holder for some particular term.

"Meta" is because it is a variable of the *metalanguage*, not *object language*.

扱う言語の項は図のように表される

今の所、項と式は同義

Program

In the present language, program = term.

```
  if false then 0 else 1;
▶ 1
  iszero (pred (succ 0));
▶ true
```

Results are boolean constants or numbers, called *values*.

Notice: `succ true`, `if 0 then 0 else 0`, ... are allowed this time.

3 Untyped Arithmetic Expressions

3.2 Syntax

Ways to define terms

- Inductively
- Rule Inferred
- Concretely

Inductive Definition

Definition (3.2.1 Terms, Inductively)

The set of *terms* is the smallest set $\mathcal{T}$ such that

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$;
2. if $\texttt{t}_1 \in \mathcal{T}$, then $\{\texttt{succ t}_1, \texttt{pred t}_1, \texttt{iszero t}_1\} \subseteq \mathcal{T}$;
3. if $\texttt{t}_1 \in \mathcal{T}$, $\texttt{t}_2 \in \mathcal{T}$, and $\texttt{t}_3 \in \mathcal{T}$, then $\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in \mathcal{T}$.

1 は T の基本的な項

2,3 は複合的な式が T に属しているか判定する方法

"smallest" は T に余分な項が含まれていないことを示す

Definition by Inference Rules

Definition (3.2.2 Terms, by Inference Rules)

The set of terms is defined by the following rules:

$$\texttt{true} \in \mathcal{T} \qquad \texttt{false} \in \mathcal{T} \qquad 0 \in \mathcal{T}$$

$$\frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{succ t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{pred t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{iszero t}_1 \in \mathcal{T}}$$

$$\frac{\texttt{t}_1 \in \mathcal{T} \quad \texttt{t}_2 \in \mathcal{T} \quad \texttt{t}_3 \in \mathcal{T}}{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in \mathcal{T}}$$

Each rule is called *inference rule*.

Each rule is read, "If we have established the statements in the premise(s) listed above the line, then we may derive the conclusion below the line."

- Rules with no premises are often called *axioms*.
- The term *inference rule* includes both axioms and rules with one or more premises.
- Axioms are usually written with no bar.

Concrete Definition

Definition (3.2.3 Terms, Concretely)

For each natural number i, define a set $\mathcal{S}_i$ as follows:

$$\begin{aligned} \mathcal{S}_0 &= \emptyset \\ \mathcal{S}_{i+1} &= \quad \{\texttt{true},\ \texttt{false},\ \texttt{0}\} \\ &\quad \cup \{\texttt{succ t}_1,\ \texttt{pred t}_1,\ \texttt{iszero t}_1 | \texttt{t}_1 \in \mathcal{S}_i\} \\ &\quad \cup \{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 | \texttt{t}_1,\ \texttt{t}_2,\ \texttt{t}_3 \in \mathcal{S}_i\} \end{aligned}$$

Finally, let $\mathcal{S} = \bigcup_i \mathcal{S}_i$.

More "concrete" style that gives an explicit procedure for *generating* the elements of $\mathcal{T}$.

Ex. 3.2.4 [★★] How many elements does $\mathcal{S}_3$ have?

Ex. 3.2.4 [★★] How many elements does $\mathcal{S}_3$ have?

$$\begin{aligned}
\mathcal{S}_0 &= \emptyset \\
\therefore |\mathcal{S}_0| &= 0 \\
\mathcal{S}_{i+1} &= \{\texttt{true},\ \texttt{false},\ \texttt{0}\} \\
&\quad \cup \{\texttt{succ t}_1,\ \texttt{pred t}_1,\ \texttt{iszero t}_1 | \texttt{t}_1 \in \mathcal{S}_i\} \\
&\quad \cup \{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 | \texttt{t}_1,\ \texttt{t}_2,\ \texttt{t}_3 \in \mathcal{S}_i\} \\
\therefore |\mathcal{S}_{i+1}| &= 3 + 3 \times |\mathcal{S}_i| + |\mathcal{S}_i|^3
\end{aligned}$$

Ex. 3.2.4 [★★] How many elements does $\mathcal{S}_3$ have?

$$
\begin{aligned}
|\mathcal{S}_0| =& 0 \\
|\mathcal{S}_{i+1}| =& 3 + 3 \times |\mathcal{S}_i| + |\mathcal{S}_i|^3
\end{aligned}
$$

$$
\begin{aligned}
|\mathcal{S}_1| =& 3 + 3 \times 0 + 0^3 &= 3 \\
|\mathcal{S}_2| =& 3 + 3 \times 3 + 3^3 &= 39 \\
|\mathcal{S}_3| =& 3 + 3 \times 39 + 39^3 &= 59439
\end{aligned}
$$

Ex. 3.2.5 [★★] Show that $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$

Ex. 3.2.5 [★★] Show that $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$

Prove inductively. Assume that $\mathtt{t} \in \mathcal{S}_i$.

- If `t` is either `true`, `false`, `0`, obvious.

Ex. 3.2.5 [★★] Show that $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$

Prove inductively. Assume that $\mathtt{t} \in \mathcal{S}_i$.

- If `t` is either `true`, `false`, `0`, obvious.
- If `t` has the form `succ t`$_1$, $\mathtt{t}_1 \in \mathcal{S}_{i-1}$ holds. From induction hypothesis, $\mathtt{t}_1 \in \mathcal{S}_i$.
  Therefore `succ t`$_1 \in \mathcal{S}_{i+1}$. The same holds for `pred` and `iszero`.

Ex. 3.2.5 [★★] Show that $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$

Prove inductively. Assume that $\mathtt{t} \in \mathcal{S}_i$.

- If $\mathtt{t}$ is either `true`, `false`, `0`, obvious.
- If $\mathtt{t}$ has the form $\mathtt{succ\ t_1}$, $\mathtt{t}_1 \in \mathcal{S}_{i-1}$ holds. From induction hypothesis, $\mathtt{t}_1 \in \mathcal{S}_i$.
  Therefore $\mathtt{succ\ t_1} \in \mathcal{S}_{i+1}$. The same holds for `pred` and `iszero`.
- If $\mathtt{t}$ has the form $\mathtt{if\ t_1\ then\ t_2\ else\ t_3}$, $\mathtt{t}_1, \mathtt{t}_2, \mathtt{t}_3 \in \mathcal{S}_{i-1}$ holds.
  From induction hypothesis, $\mathtt{t}_1, \mathtt{t}_2, \mathtt{t}_3 \in \mathcal{S}_i$.
  Therefore $\mathtt{if\ t_1\ then\ t_2\ else\ t_3} \in \mathcal{S}_{i+1}$

Two Views Define the Same Set

Proposition (3.2.6)

$\mathcal{T} = \mathcal{S}$

Proof.

Read p.28. $\square$

3 Untyped Arithmetic Expressions

3.3 Induction on Terms

Inductive

If $\mathtt{t} \in \mathcal{T}$, then one of three things must be true:

1. $\mathtt{t}$ is a constant.
2. $\mathtt{t}$ has the form $\mathtt{succ\ t_1}$, $\mathtt{pred\ t_1}$, or $\mathtt{iszero\ t_1}$ for some *smaller* term $\mathtt{t_1}$.
3. $\mathtt{t}$ has the form $\mathtt{if\ t_1\ then\ t_2\ else\ t_3}$ for some *smaller* terms $\mathtt{t_1}$, $\mathtt{t_2}$ $\mathtt{and\ t_3}$

We can

- give *inductive definitions* of functions.
- give *inductive proofs* of properties of terms.

Inductive Definitions of Consts($\texttt{t}$)

Definition (3.3.1)

$$
\begin{aligned}
Consts(\texttt{true}) &= \{\texttt{true}\} \\
Consts(\texttt{false}) &= \{\texttt{false}\} \\
Consts(\texttt{0}) &= \{\texttt{0}\} \\
Consts(\texttt{succ t}_1) &= Consts(\texttt{t}_1) \\
Consts(\texttt{pred t}_1) &= Consts(\texttt{t}_1) \\
Consts(\texttt{iszero t}_1) &= Consts(\texttt{t}_1) \\
Consts(\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3) &= Consts(\texttt{t}_1) \cup Consts(\texttt{t}_2) \cup Consts(\texttt{t}_3)
\end{aligned}
$$

Inductive Definition of size($\mathtt{t}$)

Definition (3.3.2 size($\mathtt{t}$))

$$
\begin{aligned}
size(\mathtt{true}) &= 1 \\
size(\mathtt{false}) &= 1 \\
size(\mathtt{0}) &= 1 \\
size(\mathtt{succ\ t_1}) &= size(\mathtt{t_1}) + 1 \\
size(\mathtt{pred\ t_1}) &= size(\mathtt{t_1}) + 1 \\
size(\mathtt{iszero\ t_1}) &= size(\mathtt{t_1}) + 1 \\
size(\mathtt{if\ t_1\ then\ t_2\ else\ t_3}) &= size(\mathtt{t_1}) + size(\mathtt{t_2}) + size(\mathtt{t_3}) + 1
\end{aligned}
$$

The size of $\mathtt{t}$ is the number of nodes in its abstract syntax tree.

Inductive Definition of depth($\mathtt{t}$)

Definition (3.3.2 depth($\mathtt{t}$))

$$
\begin{aligned}
depth(\mathtt{true}) &= 1\\
depth(\mathtt{false}) &= 1\\
depth(\mathtt{0}) &= 1\\
depth(\mathtt{succ\ t_1}) &= depth(\mathtt{t_1}) + 1\\
depth(\mathtt{pred\ t_1}) &= depth(\mathtt{t_1}) + 1\\
depth(\mathtt{iszero\ t_1}) &= depth(\mathtt{t_1}) + 1\\
depth(\mathtt{if\ t_1\ then\ t_2\ else\ t_3}) &= \max(depth(\mathtt{t_1}), depth(\mathtt{t_2}), depth(\mathtt{t_3})) + 1
\end{aligned}
$$

The depth of $\mathtt{t}$ is the smallest i such that $\mathtt{t} \in \mathcal{S}_i$.

Inductive Proof of a Simple Fact

Lemma (3.3.3)

$|Consts(\boldsymbol{t})| \leq size(\boldsymbol{t})$

Proof.

Inductive Proof of a Simple Fact

Lemma (3.3.3)

$|Consts(t)| \leq size(t)$

Proof.

By induction on the depth of `t`.

Case: `t` is a constant

Immediate: $|Consts(\texttt{t})| = |\{\texttt{t}\}| = 1 = size(\texttt{t})$.

Case: $\texttt{t} = \texttt{succ t}_1$, $\texttt{pred t}_1$ or $\texttt{iszero t}_1$

By the IH, $|Consts(\texttt{t}_1)| \leq size(\texttt{t}_1)$.

$\therefore |Consts(\texttt{t})| = |Consts(\texttt{t}_1)| \leq size(\texttt{t}_1) < |Consts(\texttt{t})|$

Inductive Proof of a Simple Fact

Lemma (3.3.3)

$|Consts(t)| \leq size(t)$

Proof.

Case: $\mathtt{t} = \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3$

By the IH, $|Consts(\mathtt{t}_1)| \leq size(\mathtt{t}_1)$, $|Consts(\mathtt{t}_2)| \leq size(\mathtt{t}_2)$ and $|Consts(\mathtt{t}_3)| \leq size(\mathtt{t}_3)$.

$$\begin{aligned} \therefore |Consts(\mathtt{t})| &= |Consts(\mathtt{t}_1) \cup Consts(\mathtt{t}_2) \cup Consts(\mathtt{t}_3)| \\ &\leq |Consts(\mathtt{t}_1)| + |Consts(\mathtt{t}_2)| + |Consts(\mathtt{t}_3)| \\ &\leq size(\mathtt{t}_1) + size(\mathtt{t}_2) + size(\mathtt{t}_3) \\ &< size(\mathtt{t}). \end{aligned}$$

$\square$

Thress Inductions on Terms

Theorem (3.3.4 Principles of Induction on Terms)

Induction on depth:

$$\forall \boldsymbol{s} \in \mathcal{T}, (\forall \boldsymbol{r} \in \mathcal{T} \text{ s.t. } depth(\boldsymbol{r}) < depth(\boldsymbol{s}), P(\boldsymbol{r}) \to P(\boldsymbol{s})) \\ \to \forall \boldsymbol{s} \in \mathcal{T}, P(\boldsymbol{s}).$$

Induction on size:

$$\forall \boldsymbol{s} \in \mathcal{T}, (\forall \boldsymbol{r} \in \mathcal{T} \text{ s.t. } size(\boldsymbol{r}) < size(\boldsymbol{s}), P(\boldsymbol{r}) \to P(\boldsymbol{s})) \\ \to \forall \boldsymbol{s} \in \mathcal{T}, P(\boldsymbol{s}).$$

Structural induction (構造的帰納法):

$$\forall \boldsymbol{s} \in \mathcal{T}, (\forall \boldsymbol{r} \in \mathcal{T} \text{ s.t. } \boldsymbol{r} \textit{ is a immediate subterm of } s, P(\boldsymbol{r}) \to P(\boldsymbol{s})) \\ \to \forall \boldsymbol{s} \in \mathcal{T}, P(\boldsymbol{s}).$$

Proof: Exercise (★★)

Proof: Exercise (★★)

Maybe use the concrete definition and induction on natural numbers......

$$
\begin{aligned}
\mathcal{S}_0 &= \emptyset \\
\mathcal{S}_{i+1} &= \quad \{\texttt{true, false, 0}\} \\
&\quad \cup \{\texttt{succ t}_1\texttt{, pred t}_1\texttt{, iszero t}_1 | \texttt{t}_1 \in \mathcal{S}_i\} \\
&\quad \cup \{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 | \texttt{t}_1\texttt{, t}_2\texttt{, t}_3 \in \mathcal{S}_i\}
\end{aligned}
$$

let $\mathcal{P}_i = (\forall \mathtt{s} \in \mathcal{S}_i, (\forall \mathtt{r} \in \mathcal{S}_i \ s.t.\ depth(\mathtt{r}) < depth(\mathtt{s}), P(\mathtt{r}) \rightarrow P(\mathtt{s}))$
$\rightarrow \forall \mathtt{s} \in \mathcal{S}_i, P(\mathtt{s}))$.

$\mathcal{P}_0 \wedge (\forall i \in \mathbb{N}, \mathcal{P}_i \rightarrow \mathcal{P}_{i+1}) \rightarrow \forall i \in \mathbb{N}, \mathcal{P}_i$. (induction on natural numbers)

Power of Structural Induction

Structural induction is often easier than other inductions.

Structural Induction.

By induction on `t`.

Case: `t = true`

...show P(`true`) ...

Case: `t = succ` $\mathtt{t}_1$

...show P(`succ` $\mathtt{t}_1$) using $P(\mathtt{t}_1)$ ...

Case: `t = if` $\mathtt{t}_1$ `then` $\mathtt{t}_2$ `else` $\mathtt{t}_3$

...show P(`if` $\mathtt{t}_1$ `then` $\mathtt{t}_2$ `else` $\mathtt{t}_3$) using $P(t_1), P(t_2), P(t_3)$ ...

$\square$

3 Untyped Arithmetic Expressions

3.4 Semantic Styles

In the First Place...

Two elements that characterize programming languages:

- Syntax (構文)
- Semantics (意味論)

Three Semantic Styles

- Operational semantics (操作的意味論)
- Denotational semantics (表示的意味論)
- Axiomatic semantics (公理的意味論)

Operational Semantics

Define an *abstract machine* and specify the behavior.

Example

$$\texttt{if true then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \Rightarrow \texttt{t}_2 \qquad \texttt{if false then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \Rightarrow \texttt{t}_3$$

$$\frac{\texttt{t}_1 \rightarrow \texttt{t}_1'}{\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \Rightarrow \texttt{if } \texttt{t}_1' \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3}$$

We usually use this!

Denotational Semantics

The meaning of a term is taken to be some mathematical object.

Example

$$[\![0]\!] = 0 \qquad [\![\texttt{succ t}]\!] = [\![t]\!] + 1 \qquad [\![\texttt{pred t}]\!] = [\![t]\!] - 1$$

Axiomatic Semantics

One concrete example is Hoare logic.

Hoare Triple

$$\{P\}\ C\ \{Q\}$$

where P is a precondition, C is a program, Q is a postcondition.
If P holds and C executes, then Q holds.
Ex.

$$\{x = 1, y = 1\}\ z := x + y\ \{x = 1, y = 1, z = 2\}$$