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*

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

Definition by Inference Rules

Definition (3.2.2 Terms, by Inference Rules)

The set of terms is defined by the following rules:

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

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

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

Each rule is called *inference rule*.

Concrete Definition

Definition (3.2.3 Terms, Concretely)

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

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

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

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

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

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

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 $\texttt{t} \in \mathcal{S}_i$.

- If `t` is either `true`, `false`, `0`, obvious.
- If `t` has the form `succ t`$_1$, $\texttt{t}_1 \in \mathcal{S}_{i-1}$ holds. From induction hypothesis, $\texttt{t}_1 \in \mathcal{S}_i$.
  Therefore $\texttt{succ t}_1 \in \mathcal{S}_{i+1}$. 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($\mathtt{t}$)

Definition (3.3.1)

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

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

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

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

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

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

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

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)

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

Proof.

By induction on the depth of $\mathtt{t}$.

Case: $\mathtt{t}$ is a constant

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

Case: $\mathtt{t} = \mathtt{succ\ t_1}$, $\mathtt{pred\ t_1}$ or $\mathtt{iszero\ t_1}$

By the IH, $|Consts(\mathtt{t_1})| \leq size(\mathtt{t_1})$.

$\therefore |Consts(\mathtt{t})| = |Consts(\mathtt{t_1})| \leq size(\mathtt{t_1}) < |Consts(\mathtt{t})|$

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 $|Consts(\mathtt{t_3})| \leq size(\mathtt{t_3})$.

$$\begin{aligned}
\therefore |Consts(\mathtt{t})| &= |Consts(\mathtt{t_1}) \cup Consts(\mathtt{t_2}) \cup Consts(\mathtt{t_3})| \\
&\leq |Consts(\mathtt{t_1})| + |Consts(\mathtt{t_2})| + |Consts(\mathtt{t_3})| \\
&\leq size(\mathtt{t_1}) + size(\mathtt{t_2}) + size(\mathtt{t_3}) \\
&< size(\mathtt{t}).
\end{aligned}$$

□

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):

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

Proof: Exercise (★★)

Proof: Exercise (★★)

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

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

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

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

Power of Structural Induction

Structural induction is often easier than other inductions.

Structural Induction.

By induction on $\mathtt{t}$.

Case: $\mathtt{t = true}$

...show $P(\mathtt{true})$ ...

Case: $\mathtt{t = succ\ t_1}$

...show $P(\mathtt{succ\ t_1})$ using $P(\mathtt{t_1})$ ...

Case: $\mathtt{t = if\ t_1\ then\ t_2\ else\ t_3}$

...show $P(\mathtt{if\ t_1\ then\ t_2\ else\ t_3})$ 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

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

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

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