プログラミング言語の基礎概念

semantics

導出システム … プログラミング言語の意味論や型システムを
derivation system
記述するための枠組み

自然演繹(←数理論理学で形式的証明を記述するための)やシークエント計算といった体系を一般化した汎用的な枠組み

議論の対象(論理式、プログラム、型など)に対する様々な判断を推論規則に従って導くための記述体系

判断の形式とそれに対する推論規則群を与えることによって定められる

例) 自然数の大小比較を対象とした導出システム
判断の例: 3は5より小さい
推論規則の例: 任意の自然数 n_1, n_2, n_3 について、
n_1 が n_2 より小さく、かつ n_2 が n_3 より小さいならば、
n_1 は n_3 より小さい

1.1 自然数の加算・乗算: 導出システム Nat

<u>定義1.1</u> 導出システム Nat で我々が扱う判断の一般的な形式は
n_1, n_2, n_3 をペアノ自然数
- n_1 plus n_2 is n_3
- n_1 times n_2 is n_3

のいずれかとする。

※ 形式上は正しいが内容的には正しくない判断も考えてよい
例) S(S(Z)) plus S(S(S(Z))) is S(Z)

今後、単に「判断」といったら、形式的に正しい判断を指す

<u>定義1.2</u> 判断 "n_1 plus n_2 is n_3" のための推論規則は次のふたつである

規則 P-ZERO: 任意のペアノ自然数 n に対して、
n_1 plus n_2 is n_3 を導いてよい。

規則 P-SUCC: 任意のペアノ自然数 n_1, n_2, n_3 に対して、
n_1 plus n_2 is n_3 ならば
$S(n_1)$ plus n_2 is $S(n_3)$ を導いてよい。

規則 P-SUCC のような、「J_1 ならば J_2」という形の、前提 (premise) のある推論規則は、前提となる判断 J_1 が既に得られている場合に限り J_2 の判断を導いてよい

定義 1.3 判断 "n_1 times n_2 is n_3" のための推論規則は以下のふたつ

任意のペアノ自然数

規則 T-ZERO: Z times n is Z
規則 T-SUCC: n_1 times n_2 is n_3 かつ n_2 plus n_3 is n_4 ならば、S(n_1) times n_2 is n_4

- $n_1 \times n_2 = n_3$
- $n_2 + n_3 = n_4$

$\boxed{S(n_1) \times n_2 = n_4}$

$n_2 + (n_1 \times n_2) = n_4$

$n_2 \underbrace{(n_1 + 1)}_{= S(n_1)} = n_4$

S(S(Z)) times S(S(Z)) is S(S(S(S(Z))))

規則 T-ZERO より Z times S(S(Z)) is Z …①
〃 Z plus Z is Z
$\xrightarrow{\text{P-SUCC}}$ S(Z) plus Z is S(Z)
$\xrightarrow{\text{P-SUCC}}$ S(S(Z)) plus Z is S(S(Z)) …②
規則 T-SUCC、①、② より S(Z) times S(S(Z)) is S(S(Z)) …③
規則 P-ZERO より Z plus S(S(Z)) is S(S(Z))
$\xrightarrow{\text{P-SUCC}}$ S(Z) plus S(S(Z)) is S(S(S(Z)))
$\xrightarrow{\text{P-SUCC}}$ S(S(Z)) plus S(S(Z)) is S(S(S(S(Z)))) …④
規則 T-SUCC、③、④ より S(S(Z)) times S(S(Z)) is S(S(S(S(Z)))) 終

規則の適用によって判断を導く過程を導出 (derivation) と呼ぶ。

練習 1.2 (1) S(S(S(Z))) plus S(Z) is S(S(S(S(Z))))
n = S(Z) とすると P-ZERO より Z plus S(Z) is S(Z)
P-SUCC より S(Z) plus S(Z) is S(S(Z))
P-SUCC より S(S(Z)) plus S(Z) is S(S(S(Z)))
P-SUCC より S(S(S(Z))) plus S(Z) is S(S(S(S(Z))))

プログラミング言語の基礎概念

(2) $S(Z)$ plus $S(S(S(Z)))$ is $S(S(S(S(Z))))$

$n = S(S(S(Z)))$ と P-ZERO より Z plus $S(S(S(Z)))$ is $S(S(S(Z)))$

P-SUCC より $S(Z)$ plus $S(S(S(Z)))$ is $S(S(S(S(Z))))$

(3) $\underbrace{S(S(S(Z)))}_{n_1}$ times $\underbrace{Z}_{n_2}$ is $\underbrace{Z}_{n_4}$

メモ $\underbrace{S(S(Z))}_{n_1}$ times $\underbrace{Z}_{n_2}$ is $\underbrace{Z}_{n_3}$

$\underbrace{Z}_{n_2}$ plus $\underbrace{Z}_{n_3}$ is $\underbrace{Z}_{n_4}$

$n = Z$ とすると P-ZERO より Z plus Z is Z …①

$S(Z)$ times Z is Z

メモ Z times Z is Z $(n = Z)$

$n = Z$ とすると T-ZERO より Z times Z is Z …②

規則 T-SUCC, ①, ② より $S(Z)$ times Z is Z …③

規則 T-SUCC, ①, ③ より $S(S(Z))$ times Z is Z …④

規則 T-SUCC, ①, ④ より $S(S(S(Z)))$ times Z is Z $\boxed{終}$

練習 1.3 判断 "n_1 plus n_2 is n_3" が導出できる時、その導出には何回の規則の適用が必要か?

1回目の適用

P-ZERO Z plus $\overset{n_2}{n}$ is $\overset{n_2}{n}$ $\qquad \underline{(n_1 + 1)\text{回}}$ //

"n_1 times n_2 is n_3" についてはどうか.

$(n_1 + 1)$ 回、T-SUCC (最後の1回だけは T-ZERO) を適用する

T-SUCC を適用するごとに、$(n_2 + 1)$ 回の適用が必要になる

T-SUCC & T-ZERO

$(n_1 + 1) + (n_2 + 1) \times (n_1 + 1)$

1.2 推論規則と導出の記法

規則 X: J_1 かつ … かつ J_n ならば J_0 $\qquad \dfrac{J_1 \quad \cdots \quad J_n}{J_0}$

※ 前提のない規則の場合 $n = 0$

(P-ZERO) $\dfrac{}{Z \text{ plus } n \text{ is } n}$

(P-SUCC) $\dfrac{n_1 \text{ plus } n_2 \text{ is } n}{S(n_1) \text{ plus } n_2 \text{ is } S(n)}$

(T-ZERO) $\dfrac{}{Z \text{ times } n \text{ is } Z}$

(T-SUCC) $\dfrac{n_1 \text{ times } n_2 \text{ is } n_3 \qquad n_2 \text{ plus } n_3 \text{ is } n_4}{S(n_1) \text{ times } n_2 \text{ is } n_4}$

derivation tree
導出木 … 判断をノード、結論となる判断を根とする木構造

```
S(S(S(Z))) plus S(Z) is S(S(S(S(Z)))) by P-SUCC{
  S(S(Z)) plus S(Z) is S(S(S(Z))) by P-SUCC {
    S(Z) plus S(Z) is S(S(Z)) by P-SUCC{
      Z plus S(Z) is S(Z) by P-ZERO{}
    }
  }
}
```

$$\dfrac{\dfrac{\dfrac{\dfrac{}{Z \text{ plus } S(Z) \text{ is } S(Z)}\text{P-ZERO}}{S(Z) \text{ plus } S(Z) \text{ is } S(S(Z))}\text{P-SUCC}}{S(S(Z)) \text{ plus } S(Z) \text{ is } S(S(S(Z)))}\text{P-SUCC}}{S(S(S(Z))) \text{ plus } S(Z) \text{ is } S(S(S(S(Z))))}\text{P-SUCC}$$

(3)
```
S(S(S(Z))) times Z is Z by T-SUCC {
  S(S(Z)) times Z is Z by T-SUCC{
    S(Z) times Z is Z by T-SUCC{
      Z times Z is Z by T-ZERO{};
      Z plus Z is Z by P-ZERO{}
    };
    Z plus Z is Z by P-ZERO{}
  };
  Z plus Z is Z by P-ZERO{}
}
```

プログラミング言語の基礎概念

CompareNat 1

$$\frac{}{n \text{ is less than } S(n)} \quad (\text{L-SUCC})$$

$$\frac{n_1 \text{ is less than } n_2 \qquad n_2 \text{ is less than } n_3}{n_1 \text{ is less than } n_3} \quad (\text{L-TRANS})$$

練習問題1.5 (1) Z is less than S(S(Z))

$$\dfrac{\dfrac{}{Z \text{ is less than } S(Z)}\,\text{L-SUCC} \qquad \dfrac{}{S(Z) \text{ is less than } S(S(Z))}\,\text{L-SUCC}}{Z \text{ is less than } S(S(Z))}\,\text{L-TRANS}$$

CompareNat 2

$$\frac{}{Z \text{ is less than } S(n)} \quad (\text{L-ZERO})$$

$$\frac{n_1 \text{ is less than } n_2}{S(n_1) \text{ is less than } S(n_2)} \quad (\text{L-SUCCSUCC})$$

1.5 (1)

$$\frac{}{Z \text{ is less than } S(S(Z))}\,\text{L-ZERO}$$

CompareNat 3

$$\frac{}{n \text{ is less than } S(n)} \quad (\text{L-SUCC})$$

$$\frac{n_1 \text{ is less than } n_2}{n_1 \text{ is less than } S(n_2)} \quad (\text{L-SUCCR})$$

1.5 (1)

$$\dfrac{\dfrac{}{Z \text{ is less than } S(Z)}\,\text{L-SUCC}}{Z \text{ is less than } S(S(Z))}\,\text{L-SUCCR}$$

- Natを使って定義された加算が交換法則を満たすことを示す
- EvalNatExpを使って定義された評価は、与えられた算術式に対して、その値を一意的に定める

といった、具体的な導出システムにおける、判断についての一般的な性質を考える

→このような、判断についての一般的な性質(のうち数学的に証明されたもの)をメタ定理と呼ぶ

metatheorem

帰納法 induction

ここではメタ定理の証明に用いる

操作的意味論 … 評価・簡約を使った意味論
operational semantics (ここでは算術式に意味を与えている)

ある判断が導出できているかどうかは、単なる記号操作の問題

$\dfrac{}{Z \text{ plus } n \text{ is } S(n)}$ (P-WRONG) といった規則を考えれば

$Z \text{ plus } S(Z) \text{ is } S(S(Z))$ という内容的には誤った判断を導出できるが、

P-ZEROしかないような導出システムでは
内容的には正しいはずの $S(Z) \text{ plus } S(Z) \text{ is } S(S(Z))$ を導出することができない

導出システムを「何らかの概念を表現した理論」と考えたとき、こういった導出システムの「出来」を考察した理論は**メタ理論**(理論に関する理論)である。

健全性 … 導出された判断が全て内容的に正しいという性質

完全性 … 内容的に正しい判断は全て導出できるという性質

構造帰納法　ペアノ自然数は自然数と1対1対応

(1) 任意のペアノ自然数 n に対し、$Z \text{ plus } n \text{ is } n$
(2) 任意のペアノ自然数 n に対し、$n \text{ plus } Z \text{ is } n$

(1) 規則P-ZEROにより、n が実際にどんなペアノ自然数であかによらず $\dfrac{}{Z \text{ plus } n \text{ is } n}$ P-ZERO という導出木が作れる

(2) ペアノ自然数に関する帰納法より、以下の2つを証明すればよい
(a) $Z \text{ plus } Z \text{ is } Z$ が導出できる
(b) 任意のペアノ自然数 k について $k \text{ plus } Z \text{ is } k$ が導出できるなら、$S(k) \text{ plus } Z \text{ is } S(k)$ が導出できる

※ここでは $P(n)$ は「判断 $n \text{ plus } Z \text{ is } n$ が導出できる」

(証明) (a) は P-ZERO を使えば導出できる
(b) $k \text{ plus } Z \text{ is } k$ の導出を

$$D \overset{def}{=} \dfrac{\vdots}{k \text{ plus } Z \text{ is } k}$$ とすると、

規則 P-SUCC を用いて

$$\frac{D}{S(k)\ \text{plus}\ Z\ \text{is}\ S(k)}\ \text{P-SUCC}$$

が導出できるので (b) がいえる $\boxed{\text{終}}$

(1) Z plus n is n は P-ZERO より 1ステップで導出できる。

(2) n plus Z is n は P-SUCC の n 回適用により導出できる。
↳ この直観を、数学的帰納法を使った証明という形で記述している

定理 2.2 任意のペアノ自然数 n_1, n_2, n_3, n_4 に対して、
n_1 plus n_2 is n_3 かつ n_1 plus n_2 is n_4 ならば、
$n_3 \equiv n_4$ である。

方針: n_1 についての(ペアノ自然数に関する)帰納法で証明する。

(証明) (1) $n_1 \equiv Z$ の場合 P(Z)を示す

判断 Z plus n_2 is n_3, Z plus n_2 is n_4 の導出をそれぞれ D_1, D_2 とすると、これらは以下のような形をしている。

$$D_1 \equiv \frac{}{Z\ \text{plus}\ n_2\ \text{is}\ n_3}\ \text{P-ZERO}$$

$$D_2 \equiv \frac{}{Z\ \text{plus}\ n_2\ \text{is}\ n_4}\ \text{P-ZERO}$$

さらに、P-ZERO の形より、$n_3 \equiv n_4 \equiv n_2$

P(n): n plus n_2 is n_3, n plus n_2 is n_4 が導出できるなら $n_3 \equiv n_4$

(2) ある k に対して $n_1 \equiv S(k)$ の場合

まず、判断 $S(k)$ plus n_2 is n_3, $S(k)$ plus n_2 is n_4 が導出できていると仮定する。これらの導出を D_1, D_2 とすると、$n_3 \equiv S(n_3')$, $n_4 \equiv S(n_4')$ が成り立つようなペアノ自然数 n_3', n_4' が存在して、D_1, D_2 は以下のような形をしている

$$D_1 \equiv \frac{\dfrac{\vdots}{k\ \text{plus}\ n_2\ \text{is}\ n_3'}}{S(k)\ \text{plus}\ n_2\ \text{is}\ n_3}\ \text{P-SUCC} \qquad D_2 \equiv \frac{\dfrac{\vdots}{k\ \text{plus}\ n_2\ \text{is}\ n_4'}}{S(k)\ \text{plus}\ n_2\ \text{is}\ n_4}\ \text{P-SUCC}$$

部分に注目

帰納法の仮定 P(k) より、$n_3' \equiv n_4'$

よって $n_3 \equiv S(n_3') \equiv S(n_4') \equiv n_4$ $\boxed{\text{終}}$