1 計算理論 II について

- 評価は，発言表，レポートおよび期末テスト（小テストをする場合あり）の結果を合わせて行う．
- 授業は，横内寛文著「プログラム意味論」(情報数学講座 7，共立出版）に沿って行う．
- 細かい情報は Web（http://www.cs.is.noda.tus.ac.jp/~mune/sem/）で知らせる．

2 プログラム意味論

2.1 構文規則

プログラムの意味論を考える上で，プログラムを記述するための言語を定義する．まず，構文規則を定義する．例えば，BNF を用いて次のように定義できる．

$$
\begin{aligned}
&< \text{変数} > ::= A \mid B \mid C \mid \cdots \mid Z \\
&< \text{定数} > ::= 0 \mid 1 \mid 2 \mid \cdots \mid 9 \\
&\quad < \text{式} > ::= < \text{変数} > \mid < \text{定数} > \mid < \text{式} > + < \text{式} > \mid < \text{式} > - < \text{式} > \\
&\quad < \text{文} > ::= < \text{変数} > := < \text{式} > \\
&\qquad\qquad \mid \textbf{if } < \text{式} > = < \text{式} > \textbf{ then } < \text{文} > \textbf{ else } < \text{文} > \\
&\qquad\qquad \mid \textbf{for } < \text{式} > \textbf{ do } < \text{文} > \\
&\qquad\qquad \mid \textbf{noop}
\end{aligned}
$$

2.1.1 構文規則が表していること

1. A, B, C, ..., Z は変数
2. 0, 1, 2, ..., 9 は定数
3. (a) 変数と定数は式
   (b) E_1 と E_2 が式ならば，$E_1 + E_2$ と $E_1 - E_2$ も式
4. (a) x が変数で，E が式ならば，$x := E$ は文
   (b) E_1 と E_2 が式で，C_1 と C_2 が文ならば，**if** $E_1 = E_2$ **then** C_1 **else** C_2 は文
   (c) E が式で C が文ならば，**for** E **do** C は文
   (d) **noop** は文
5. 変数，定数，式，文は，1〜4 の規則を有限回適用して得られる記号列

2.1.2 曖昧な構文規則

上記の構文規則を用いて，$A + B - C$ を生成する場合，2 通りの方法がある．⇒ **曖昧**

方法 1 ：

1. A と B は式なので，$A + B$ は式，
2. $A + B$ と C は式なので，$A + B - C$ は式．

$$\Downarrow$$

$$(A + B) - C$$

方法 2 ：

1. B と C は式なので，$B - C$ は式，
2. A と $B - C$ は式なので，$A + B - C$ は式．

$$\Downarrow$$

$$A + (B - C)$$

2.1.3 曖昧性の除去

以降，構文規則によって生成される記号列に関しては，その適用順序に曖昧性はないものとする．

- 構文の変形,

$$< 式 > ::= < 変数 > \mid < 定数 > \mid (< 式 > \ + \ < 式 >) \mid (< 式 > \ - \ < 式 >)$$

- 決まった構文木を対象とするなど．

2.1.4 プログラムの意味

$$Z := 0;$$
$$\textbf{for } Y \textbf{ do } Z \ := \ Z \ + \ X$$

掛け算を行うプログラム ： 変数 X と Y に，負でない整数を代入して実行すると，変数 Z に $X * Y$ の値が代入されて停止する．

⇓

形式的に正確な定義を与えたい！

2.2 意味関数

構文規則によって生成された記号列の意味を考える．

2.2.1 式の値の定義

式の値の集合 ： 値の集合 Val は，自然数 $\mathbf{N}$={0,1,2,...}

2.2.2 式の意味関数

- 式の値は，その式に含まれる変数の値によって決まる．

⇓ 環境の導入

環境 ϕ, ψ ： 変数集合 Var から値集合 Val への関数

例：変数 x には，$\phi(x)$ の値が代入されている．

式の値 ： 式 E の環境 ϕ における値を $\mathcal{M}_{\exp}[\![\ E\ ]\!]\phi$ と表し，式の構文規則に対して，次のように定義する．

$$\begin{aligned}
\mathcal{M}_{\exp}[\![\ x\ ]\!]\phi &= \phi(x), \\
\mathcal{M}_{\exp}[\![\ 0\ ]\!]\phi &= 0, \\
&\cdots \\
\mathcal{M}_{\exp}[\![\ 9\ ]\!]\phi &= 9, \\
\mathcal{M}_{\exp}[\![\ E_1 + E_2\ ]\!]\phi &= \mathcal{M}_{\exp}[\![\ E_1\ ]\!]\phi \ + \ \mathcal{M}_{\exp}[\![\ E_2\ ]\!]\phi, \\
\mathcal{M}_{\exp}[\![\ E_1 - E_2\ ]\!]\phi &= \mathcal{M}_{\exp}[\![\ E_1\ ]\!]\phi \ - \ \mathcal{M}_{\exp}[\![\ E_2\ ]\!]\phi.
\end{aligned}$$

- $\mathcal{M}_{\exp}[\![\ 0\ ]\!]\phi \ = \ 0$： 左辺 $[\![\ ]\!]$ 内の 0 は記号であり，右辺の 0 は自然数,
- $\mathcal{M}_{\exp}[\![\ E_1 + E_2\ ]\!]\phi \ = \ \mathcal{M}_{\exp}[\![\ E_1\ ]\!]\phi \ + \ \mathcal{M}_{\exp}[\![\ E_2\ ]\!]\phi$ ： 左辺の + は記号であり，右辺の + は，自然数上の加算,
- $\mathcal{M}_{\exp}[\![\ E_1 - E_2\ ]\!]\phi \ = \ \mathcal{M}_{\exp}[\![\ E_1\ ]\!]\phi \ - \ \mathcal{M}_{\exp}[\![\ E_2\ ]\!]\phi$ ： $\mathcal{M}_{\exp}[\![\ E_1\ ]\!]\phi < \mathcal{M}_{\exp}[\![\ E_2\ ]\!]\phi$ の場合は，0 と定義.

2.2.3 文の意味関数

- 代入文： 変数の値を置き換える働き．⇒ 環境を別の環境に移す関数,
- その他の文：代入文同様，環境を別の環境に移す関数.

環境から環境への関数 ： 環境の集合を Env として，Env から Env の関数を，$\mathcal{M}_{\mathrm{cmd}}[\![\ C\ ]\!]$ と表す.

例： $\mathcal{M}_{\mathrm{cmd}}[\![\ C\ ]\!]\phi$： 環境 ϕ の下で文 C を実行したのちの環境.

各文の構文規則に対して，次のように定義する.

$$\mathcal{M}_{\mathrm{cmd}}[\![\ x := E\ ]\!]\phi = \phi(x := \mathcal{M}_{\mathrm{exp}}[\![\ E\ ]\!]\phi)$$

$$\mathcal{M}_{\mathrm{cmd}}[\![\ \textbf{if } E_1 = E_2 \textbf{ then } C_1 \textbf{ else } C_2\ ]\!]\phi = \begin{cases} \mathcal{M}_{\mathrm{cmd}}[\![\ C_1\ ]\!]\phi & (\mathcal{M}_{\mathrm{exp}}[\![\ E_1\ ]\!]\phi = \mathcal{M}_{\mathrm{exp}}[\![\ E_2\ ]\!]\phi) \\ \mathcal{M}_{\mathrm{cmd}}[\![\ C_2\ ]\!]\phi & (\mathcal{M}_{\mathrm{exp}}[\![\ E_1\ ]\!]\phi \neq \mathcal{M}_{\mathrm{exp}}[\![\ E_2\ ]\!]\phi) \end{cases}$$

$$\mathcal{M}_{\mathrm{cmd}}[\![\ \textbf{for } E \textbf{ do } C\ ]\!]\phi = f(...(f(\phi))...) \quad (n = \mathcal{M}_{\mathrm{exp}}[\![\ E\ ]\!]\phi,\ f = \mathcal{M}_{\mathrm{cmd}}[\![\ C\ ]\!])$$

$$\mathcal{M}_{\mathrm{cmd}}[\![\ \textbf{noop}\ ]\!]\phi = \phi$$

- $\mathcal{M}_{\mathrm{cmd}}[\![\ x := E\ ]\!]\phi\ = \psi$ とおくと,

$$\psi(x) = \mathcal{M}_{\mathrm{exp}}[\![\ E\ ]\!]\phi,$$
$$\psi(y) = \phi(y)\ (y \text{ は } x \text{ 以外の変数})$$

$\phi(x := \mathcal{M}_{\mathrm{exp}}[\![\ E\ ]\!]\phi)$ は この ψ を表す.

2.3 表示的意味論

構文規則で定義された式や文を表す記号列に対して，$\mathcal{M}_{\mathrm{exp}}$ や $\mathcal{M}_{\mathrm{cmd}}$ のような**意味関数**（meaning function）を定義して，プログラムの意味を定義する手法を**表示的意味論**（denotational semantics）と呼ぶ.

- $\mathcal{M}_{\mathrm{exp}}$ ： Exp → (Env → Val)
- $\mathcal{M}_{\mathrm{cmd}}$ ： Cmd → (Env → Env)

2.3.1 表示的意味論の方向性

上記の構文規則に次の **while** 文を追加する.

$$\textbf{while } <\text{式}> = <\text{式}> \textbf{ do } <\text{文}>$$

- 何回ループを繰り返すか分らない．⇒ 無限ループに陥る可能性.

$$\Downarrow$$

$\mathcal{M}_{\mathrm{cmd}}[\![C]\!]\phi$ が未定義（$\mathcal{M}_{\mathrm{cmd}}$ は**部分関数**）

- **while** $E_1 = E_2$ **do** C は，次のように展開できる.

$$\textbf{if } E_1 = E_2 \textbf{ then } C;\ \textbf{while } E_1 = E_2 \textbf{ do } C \quad \textbf{else noop}$$

$$\Downarrow$$

$$\mathcal{M}_{\mathrm{cmd}}[\![\textbf{while } E_1 = E_2 \textbf{ do } C]\!]\phi = \begin{cases} \mathcal{M}_{\mathrm{cmd}}[\![\textbf{while } E_1 = E_2 \textbf{ do } C]\!]\phi' & (\mathcal{M}_{\mathrm{exp}}[\![E_1]\!]\phi = \mathcal{M}_{\mathrm{exp}}[\![E_2]\!]\phi \text{のとき}) \\ \phi & (\mathcal{M}_{\mathrm{exp}}[\![E_1]\!]\phi \neq \mathcal{M}_{\mathrm{exp}}[\![E_2]\!]\phi \text{のとき}) \end{cases}$$

ただし, $\phi' = \mathcal{M}_{\mathrm{cmd}}[\![C]\!]\phi$

これは定義になっていない ⇒ 方程式を解く必要

他の例： $\mathrm{fact}(n) = \textbf{if } n = 0 \textbf{ then } 1 \textbf{ else } n * \mathrm{fact}(n-1)$

2.4 操作的意味論

プログラムをコンピュータに対する指令ととらえ，コンピュータのメモリやレジスタなどの内部状態がどのように変かするかを定義する意味の定義法を，**操作的意味論**（operational semantics）と呼ぶ.

操作的意味の定義法 ：

- コンピュータの内部状態の列を定義,
- インタプリタや仮想的なコンピュータに対するコンパイラを定義.

- 表示的意味論が，与えられたプログラムが何をするのかを静的に定義するのに対して，操作的意味論は，**時間的な経過**に着目して，意味を定義する.

2.5 再帰的定義と構造帰納法

節 2.1 のような再帰的に定義したものの性質は，構造帰納法によって証明することができる.