言語の意味論1
―序と操作的意味論―

2007 秋学期

プログラミング言語の意味記述

- 言語の定義を、マニュアルのような曖昧な自然言語による記述から脱却させようとする運動が‘50年代後半から起こった。
- 大きくわけて以下のような3つの方法がある。
  それぞれに、目的、効用、展開が異なる。

3つの方法

- 操作的方法(operational method):
  - **プログラムを実行する抽象的な機械(abstract machine)やインタプリタを厳密に定義する。**
- 公理的方法(axiomatic method)
  - **プログラムの文面をその一部として含む論理体系を定義する。 C.A.R. HoareやR.Floydがはじめる。**
- 表示的方法(denotational method)
  - **一つのプログラムにたいして、それが意味する数学的実在物(e.g.,関数)を対応させる。プログラム全体とその数学的対象との関係が整合的に定義されなければならない。**

操作的意味論

- 対象となる言語で書かれたプログラムを実行させた時の挙動を明確(逐一）の与えるアプローチ
- 歴史的にはこの方法・考え方が一番古い。‘58ごろJ.McCarthyがLISPのinterpreter（解釈器）をLISPで記述している。（Scheme Int. in Scheme)
  - **このほか、‘60前半、ソースコードでなく構文解析木（abstract tree(抽象木)と当時呼ばれた）のようなものを解釈・変換する機械を定義する、Vienna Methodというものが提唱された。**
- 言語の実装に近い部分が詳細に記述できる。
  - **解釈器を与える方法は、記述がプログラムなので抽象性が低く展開しにくかったが、後述のようにG.Plotkinが論理体系のような形式に表現して、推論がしやすくなった。**
  - **実装を示唆する点で、言語の実装者にとっては便利。**

抽象構文によるアプローチ

- プログラムの文面を抽象構文で表現。
- 抽象構文で表現されたプログラムを解釈・実行する手続きを定義するとそれが操作的意味記述となる。
- その定義を記述するために、
  1. 機械を与える。(抽象構文によるプログラムを読みながらそれを実行する。)
  2. プログラムを与える (e.g., Scheme Int. in Scheme)
  3. ...

抽象構文

- 言語の各構文要素を抽象的なレコード

  [tag1: *r1*, tag2: *r2*, …, tagn: *rn*]

  で表現する。
- プログラムの構文解析が完了して抽象レコードの木構造でプログラムが表現されていることを仮定してする。
- 言語の表層的な表現と独立に意味記述の議論ができる。
- 抽象レコードに対して4つの操作が必要:

  0. 判定関数: レコードの名前を判定する　**xxx-?(R)**
  1. 構成関数: 新たに抽象レコードを作る

     make-xxx(*r1*,*r2*,…, *rn*)
  2. 選択関数: 抽象レコードの要素を取り出す　**get-*tagi*(…)**
  3. 更新関数: 抽象レコードの要素を更新する　**update-*tagi*(…)**

Interpreterによる方法（例） Abelson & Sussman's Book

```
(define (eval exp env)
  (cond ((self-evaluating? exp) exp)
        ((variable? exp) (lookup-variable-value exp env))
        ((quoted? exp) (text-of-quotation exp))
        ((assignment? exp) (eval-assignment exp env))
        ((definition? exp) (eval-definition exp env))
        ((if? exp) (eval-if exp env))
        ((lambda? exp)
         (make-procedure (lambda-parameters exp) (lambda-body exp) env))
        ((begin? exp) (eval-sequence (begin-actions exp) env))
        ((cond? exp) (eval (cond->if exp) env))
        ((application? exp)
         (apply (eval (operator exp) env) (list-of-values (operands exp) env)))
        (else (error "Unknown expression type -- EVAL" exp))))
```

expは、Schemeの式が何らかのデータ表現（in Scheme)に変換されていると仮定している。

Syntax-Directed アプローチ

- 前述の抽象構文に基づくアプローチと異なり、各構文単位に対して、プログラムの文面を取り込んだ推論規則または公理を定義し、それにより論理体系を与えて、インタープリタを定義する。プログラムの逐一の挙動を記述する方式がある。(G.Plotkin)
- この方式のoperational approachを学ぶために、以下、簡単な命令型(imperative)言語を定義し、そのsemanticsを与える。

IMP – a simple imperative language

Syntactic sets

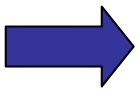
- numbers **N**, consisting of positive and negative integers with zero,
- truth values $\mathbf{T} = \{\mathbf{true}, \mathbf{false}\}$,
- locations **Loc**,
- arithmetic expressions **Aexp**,
- boolean expressions **Bexp**,
- commands **Com**.

Loc might consist of non-empty strings of letters or such strings followed digits. Locations are often called program variables

Meta-variables associated with Syntactic sets

- n, m range over numbers **N**,
- X, Y range over locations **Loc**,
- a ranges over arithmetic expressions **Aexp**,
- b ranges over boolean expressions **Bexp**,
- c ranges over commands **Com**.

Inductive definition of syntactic set IMP, the least set closed under the rules.



Aexp:

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1.$$

Bexp:

$$b ::= \mathbf{true} \mid \mathbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$$

Com:

$$c ::= \mathbf{skip} \mid X := a \mid c_0; c_1 \mid \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1 \mid \mathbf{while}\ b\ \mathbf{do}\ c$$

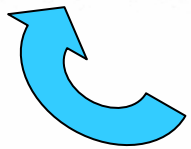
状態σと算術式(Aexp)の評価

- 「状態」とは、各変数・メモリー語の現在持つ・格納している値のこと。すなわち、状態σは関数で表現され、σ：　Loc —＞　N

The set of *states* Σ consists of functions $\sigma : \mathbf{Loc} \to \mathbf{N}$ from locations to numbers.
$\sigma(X)$ is the value, or contents, of location X in state σ.

- 評価関係(evaluation relation)の定義（例）

$$\langle a, \sigma \rangle \to n$$



算術式 a が状態σで評価せれると結果 n が得られる。

算術式の評価規則群

Evaluation of numbers:

$$\langle n, \sigma \rangle \to n$$

Thus any number is already evaluated with itself as value.

Evaluation of locations:

$$\langle X, \sigma \rangle \to \sigma(X)$$

Thus a location evaluates to its contents in a state.

Evaluation of sums:

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 + a_1, \sigma \rangle \to n}$$

where n is the sum of n_0 and n_1.

Evaluation of subtractions:

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 - a_1, \sigma \rangle \to n}$$

where n is the result of subtracting n_1 from n_0.

Evaluation of products:

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \times a_1, \sigma \rangle \to n}$$

where n is the product of n_0 and n_1.

推論規則の適用による評価

$$\frac{\langle 2, \sigma_0 \rangle \rightarrow 2 \quad \langle 3, \sigma_0 \rangle \rightarrow 3}{\langle 2 \times 3, \sigma_0 \rangle \rightarrow 6}$$

$$\frac{\langle 2, \sigma_0 \rangle \rightarrow 3 \quad \langle 3, \sigma_0 \rangle \rightarrow 4}{\langle 2 \times 3, \sigma_0 \rangle \rightarrow 12},$$

どれも導出されない

consider the evaluation of $a \equiv (\mathrm{Init} + 5) + (7 + 9)$ in state σ_0, where Init is a location with $\sigma_0(\mathrm{Init}) = 0$.

$$\frac{\dfrac{\overline{\langle \mathrm{Init}, \sigma_0 \rangle \rightarrow 0} \quad \overline{\langle 5, \sigma_0 \rangle \rightarrow 5}}{\langle (\mathrm{Init} + 5), \sigma_0 \rangle \rightarrow 5} \quad \dfrac{\overline{\langle 7, \sigma_0 \rangle \rightarrow 7} \quad \overline{\langle 9, \sigma_0 \rangle \rightarrow 9}}{\langle 7 + 9, \sigma_0 \rangle \rightarrow 16}}{\langle (\mathrm{Init} + 5) + (7 + 9), \sigma_0 \rangle \rightarrow 21}$$

Boolean式(Bexp)の評価規則

$\langle \mathbf{true}, \sigma \rangle \to \mathbf{true}$

$\langle \mathbf{false}, \sigma \rangle \to \mathbf{false}$

$$\frac{\langle a_0, \sigma \rangle \to n \quad \langle a_1, \sigma \rangle \to m}{\langle a_0 = a_1, \sigma \rangle \to \mathbf{true}}$$ if n and m are equal

$$\frac{\langle a_0, \sigma \rangle \to n \quad \langle a_1, \sigma \rangle \to m}{\langle a_0 = a_1, \sigma \rangle \to \mathbf{false}}$$ if n and m are unequal

$$\frac{\langle a_0, \sigma \rangle \to n \quad \langle a_1, \sigma \rangle \to m}{\langle a_0 \leq a_1, \sigma \rangle \to \mathbf{true}}$$ if n is less than or equal to m

$$\frac{\langle a_0, \sigma \rangle \to n \quad \langle a_1, \sigma \rangle \to m}{\langle a_0 \leq a_1, \sigma \rangle \to \mathbf{false}}$$ if n is not less than or equal to m

$$\frac{\langle b, \sigma \rangle \to \mathbf{true}}{\langle \neg b, \sigma \rangle \to \mathbf{false}} \qquad \frac{\langle b, \sigma \rangle \to \mathbf{false}}{\langle \neg b, \sigma \rangle \to \mathbf{true}}$$

$$\frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \wedge b_1, \sigma \rangle \to t}$$

where t is **true** if $t_0 \equiv \mathbf{true}$ and $t_1 \equiv \mathbf{true}$, and is **false** otherwise.

$$\frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \vee b_1, \sigma \rangle \to t}$$

where t is **true** if $t_0 \equiv \mathbf{true}$ or $t_1 \equiv \mathbf{true}$, and is **false** otherwise.

命令の実行

- 式の役目はある状態で値を得る・評価すること。
  一方、命令は状態を変化させることである。
- 命令 c を状態 σ で実行し、実行が終了したとき状態が σ' となるとすると:

$$\langle c, \sigma \rangle \rightarrow \sigma'$$

- 例えば:

$$\langle X := 5, \sigma \rangle \rightarrow \sigma'$$

where σ' is the state σ updated to have 5 in location X.

更新を表す記法

Notation: Let σ be a state. Let $m \in \mathbf{N}$. Let $X \in \mathbf{Loc}$. We write $\sigma[m/X]$ for the state obtained from σ by replacing its contents in X by m, i.e. define

$$\sigma[m/X](Y) = \begin{cases} m & \text{if } Y = X, \\ \sigma(Y) & \text{if } Y \neq X. \end{cases}$$

Now we can instead write

$$\langle X := 5, \sigma \rangle \to \sigma[5/X].$$

命令に対する推論規則

Rules for commands

Atomic commands:

$$\langle \mathbf{skip}, \sigma \rangle \to \sigma$$

$$\frac{\langle a, \sigma \rangle \to m}{\langle X := a, \sigma \rangle \to \sigma[m/X]}$$

Sequencing:

$$\frac{\langle c_0, \sigma \rangle \to \sigma'' \quad \langle c_1, \sigma'' \rangle \to \sigma'}{\langle c_0; c_1, \sigma \rangle \to \sigma'}$$

Conditionals:

$$\frac{\langle b, \sigma \rangle \to \mathbf{true} \quad \langle c_0, \sigma \rangle \to \sigma'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma \rangle \to \sigma'}$$

$$\frac{\langle b, \sigma \rangle \to \mathbf{false} \quad \langle c_1, \sigma \rangle \to \sigma'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma \rangle \to \sigma'}$$

While-loops:

$$\frac{\langle b, \sigma \rangle \to \mathbf{false}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \to \sigma}$$

$$\frac{\langle b, \sigma \rangle \to \mathbf{true} \quad \langle c, \sigma \rangle \to \sigma'' \quad \langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma'' \rangle \to \sigma'}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \to \sigma'}$$

同値関係と課題

- 2つの命令が同値とは:

$$c_0 \sim c_1 \text{ iff } \forall \sigma, \sigma' \in \Sigma.\ \langle c_0, \sigma \rangle \to \sigma' \iff \langle c_1, \sigma \rangle \to \sigma'.$$

- 課題:

Let $w \equiv$ **while true do skip**. By considering the form of derivations, explain why, for any state σ, there is no state σ' such that $\langle w, \sigma \rangle \to \sigma'$. □

- 課題:

Let $w \equiv$ **while** b **do** c *with* $b \in$ **Bexp**, $c \in$ **Com**. *Then*

$$w \sim \textbf{if } b \textbf{ then } c; w \textbf{ else skip}.$$

これを証明せよ！

課題

- IMPで階乗を計算するプログラムを書き、それが正しいプログラムであることを、証明せよ。
- IMPでユークリッドの互除法のアルゴリズムをかき、それが最大公約数を計算することを証明せよ。