プログラム言語論
意味論

亀山幸義

筑波大学 情報科学類

No. 2

前回

- プログラム言語「論」
- コンパイラとインタープリタ
- 構文と意味

演習

```
#include <stdio.h>
void foo (int y, int z) {
  printf("%d %d\n", y, z);
}
main () {
  int x = 0;
  foo(++x, ++x);
}
```

答え: "1 2"と"2 1" と"2 2"のどれにでもなり得る (処理系依存). ※ C 言語の仕様書では,「左から」とも「右から」とも決めていない (unspecified) どころか、2 つの++を計算してから 2 つの引数を積む、ということも許している。

“TYPES” メイリングリスト

(Robbert Krebbers 氏の記事より引用, 2013/4/24)
Let me then notice that in the case of C, it is worse than just non-determinism. There are also so called sequence point violations, which happen if you modify an object more than once (or read after you've modified it) in between two sequence points. For example

```
int x = 0;
int main() {
  printf("%d ", (x = 3) + (x = 4));
  printf("%d\n", x);
  return 0;
}
```

not just randomly prints "7 3" or "7 4", but instead gives rise to undefined behavior, and could print arbitrary nonsense. When compiled with gcc at my machine, it for example prints "8 4".

意味を厳密に考える必要性

「プログラムの意味を決める」とは、プログラムを実行するとどのような結果になるかを (厳密に) 決めること。

言葉による説明しかないプログラム言語では‥。

- コンパイラが正しいかどうか確かめられない。
- プログラムの性質を解析・検証できない。
- プログラムの保守・再利用もできない。

プログラム言語の意味論

- 厳密な意味論がある言語: Scheme, Standard ML
- 言葉による意味論がある言語: C, Java, etc.
- 言葉による意味論もない言語: Ruby, etc.

さまざまな意味論の与え方

- 操作的意味論 (operational semantics)
  - small-step semantics (structural operational semantics)
  - bigl-step semantics (natural semantics)
  - abstract machine semantics
- 公理的意味論 (axiomatic semantics)
- 表示的意味論 (外延的意味論; denotational semantics)

ここでは、小さなプログラム言語に対する意味論を 3 通りの方法で与える．(表示的意味論，操作的意味論 1，操作的意味論 2)

小さな小さなプログラム言語

自然数に対する加算と乗算のみを許すプログラム言語の (具体) 構文：

$$n ::= 0 \mid 1 \mid 2 \mid \cdots \quad \text{(自然数定数)}$$
$$e ::= n \mid (e) \mid e + e \mid e * e \quad \text{(式)}$$

式の例: ((1+(2*3)))

上記のプログラム言語の抽象構文：

$$e ::= \mathtt{Int}(n) \mid \mathtt{Plus}(e, e) \mid \mathtt{Times}(e, e)$$

抽象構文: 構文木に相当するもの，((1)) と (1) と 1 は，異なる (具体) 構文を持つが，構文木としては同じと考えたい．曖昧さはない．

表示的意味論

プログラムの各要素を、何らかの数学的要素に対応付ける。

$$[\![\mathtt{Int}(n)]\!] = n$$
$$[\![\mathtt{Plus}(e_1, e_2)]\!] = [\![e_1]\!] \dot{+} [\![e_2]\!]$$
$$[\![\mathtt{Times}(e_1, e_2)]\!] = [\![e_1]\!] \dot{*} [\![e_2]\!]$$

ただし、$\dot{+}$ は、式の構文にでてくる $+$ の記号ではなく、2 つの自然数の足し算をあらわす。$\dot{*}$ も同様
例. $[\![\mathtt{Plus}(\mathtt{Int}(1), \mathtt{Times}(\mathtt{Int}(2), \mathtt{Int}(3)))]\!] =$
$[\![\mathtt{Int}(1)]\!] \dot{+} [\![\mathtt{Times}(\mathtt{Int}(2), \mathtt{Int}(3))]\!] = 1 \dot{+} ([\![\mathtt{Int}(2)]\!] \dot{*} [\![\mathtt{Int}(3)]\!]) = 7$

操作的意味論1： SOS

Strucutal Operational Semantics (構造的操作的意味論):
式 e を計算した結果が n であることを $e \downarrow n$ と書く。

$$\frac{}{\mathtt{Int}(n) \downarrow n}$$

$$\frac{e_1 \downarrow n_1 \quad e_2 \downarrow n_2 \quad (n_1 \dot{+} n_2 = n)}{\mathtt{Plus}(e_1, e_2) \downarrow n} \qquad \frac{e_1 \downarrow n_1 \quad e_2 \downarrow n_2 \quad (n_1 \dot{*} n_2 = n)}{\mathtt{Times}(e_1, e_2) \downarrow n}$$

例題。$\mathtt{Plus}(\mathtt{Int}(1), \mathtt{Times}(\mathtt{Int}(2), \mathtt{Int}(3))) \downarrow 7$ の導出は以下の通り。

$$\dfrac{\dfrac{}{\mathtt{Int}(1) \downarrow 1} \quad \dfrac{\dfrac{}{\mathtt{Int}(2) \downarrow 2} \quad \dfrac{}{\mathtt{Int}(3) \downarrow 3} \quad (2 \dot{*} 3 = 6)}{\mathtt{Times}(\mathtt{Int}(2), \mathtt{Int}(3)) \downarrow 6} \quad (1 \dot{+} 6 = 7)}{\mathtt{Plus}(\mathtt{Int}(1), \mathtt{Times}(\mathtt{Int}(2), \mathtt{Int}(3))) \downarrow 7}$$

操作的意味論2： 抽象機械による意味論

CK 機械 (C=constrol string or code, K=continuation or stack)

$$\begin{aligned}
\langle \mathit{eval}, \mathtt{Int}(n), K \rangle &\to \langle \mathit{apply}, K, n \rangle \\
\langle \mathit{eval}, \mathtt{Plus}(e_1, e_2), K \rangle &\to \langle \mathit{eval}, e_1, \ \mathtt{push}((\mathit{plus}1, e_2), K) \rangle \\
\langle \mathit{eval}, \mathtt{Times}(e_1, e_2), K \rangle &\to \langle \mathit{eval}, e_1, \ \mathtt{push}((\mathit{times}1, e_2), K) \rangle \\
\langle \mathit{apply}, \mathtt{push}((\mathit{plus}1, e), K), n \rangle &\to \langle \mathit{eval}, e, \mathtt{push}((\mathit{plus}2, n), K) \rangle \\
\langle \mathit{apply}, \mathtt{push}((\mathit{plus}2, n_2), K), n_1 \rangle &\to \langle \mathit{apply}, K, n \rangle \quad (n_2 \dot{+} n_1 = n) \\
\langle \mathit{apply}, \mathtt{push}((\mathit{times}1, e), K), n \rangle &\to \langle \mathit{eval}, e, \mathtt{push}((\mathit{times}2, n), K) \rangle \\
\langle \mathit{apply}, \mathtt{push}((\mathit{times}2, n_2), K), n_1 \rangle &\to \langle \mathit{apply}, K, n \rangle \quad (n_2 \dot{*} n_1 = n) \\
\langle \mathit{apply}, \mathtt{init}, n \rangle &\to n \ \ \text{(最終結果)}
\end{aligned}$$

計算は $\langle \mathit{eval}, e, \mathtt{init} \rangle$ という状態からはじめる。
initスタックが空であることを意味する.

抽象機械の実行例

$e_0 = \mathtt{Times}(\mathtt{Int}(3), \mathtt{Plus}(\mathtt{Int}(1), \mathtt{Int}(2)))$ の計算.

$$\begin{aligned}
&\langle \mathit{eval}, e_0, \mathtt{init} \rangle \\
&\to \langle \mathit{eval}, \mathtt{Int}(3), \mathtt{push}((\mathit{times}1, \mathtt{Plus}(\mathtt{Int}(1), \mathtt{Int}(2))), \mathtt{init}) \rangle \\
&\to \langle \mathit{apply}, \mathtt{push}((\mathit{times}1, \mathtt{Plus}(\mathtt{Int}(1), \mathtt{Int}(2))), \mathtt{init}), 3 \rangle \\
&\to \langle \mathit{eval}, \mathtt{Plus}(\mathtt{Int}(1), \mathtt{Int}(2)), \mathtt{push}((\mathit{times}2, 3), \mathtt{init}) \rangle \\
&\to \langle \mathit{eval}, \mathtt{Int}(1), \mathtt{push}((\mathit{plus}1, \mathtt{Int}(2)), \mathtt{push}((\mathit{times}2, 3), \mathtt{init})) \rangle \\
&\to \langle \mathit{apply}, \mathtt{push}((\mathit{plus}1, \mathtt{Int}(2)), \mathtt{push}((\mathit{times}2, 3), \mathtt{init})), 1 \rangle \\
&\to \langle \mathit{eval}, \mathtt{Int}(2), \mathtt{push}((\mathit{plus}2, 1), \mathtt{push}((\mathit{times}2, 3), \mathtt{init})) \rangle \\
&\to \langle \mathit{apply}, \mathtt{push}((\mathit{plus}2, 1), \mathtt{push}((\mathit{times}2, 3), \mathtt{init})), 2 \rangle \\
&\to \langle \mathit{apply}, \mathtt{push}((\mathit{times}2, 3), \mathtt{init}), 3 \rangle \\
&\to \langle \mathit{apply}, \mathtt{init}, 9 \rangle \\
&\to 9
\end{aligned}$$

抽象機械 (Abstract Machine)

- (本当の) 機械: machine (computer)
  - ハードウェアで実現
- 仮想機械: virtual machine
  - インタープリタで実現される。
  - 通常、instruction set を持ち、本当の機械に近い。
- 抽象機械: abstract machine
  - インタープリタで実現される。
  - 抽象度が高い (高級言語に、より近い)。
  - 通常、instruction set を持たない。

代表的な抽象機械

- SECD machine [Landin 1964]
- CEK machine [Felleisen 1989]
- WAM, CHAM, Krivine machine, ZINC, ...

(形式) 意味論

ともかく、プログラムの意味を厳密に決めるもの。

- 操作的意味論 (operational semantics)
  - プログラムの動作を記述。
- 公理的意味論 (axiomatic semantics)
  - プログラムが満たす性質を記述。
- 表示的意味論 (外延的意味論; denotational semantics)
  - プログラムが表している (数学的な) ものを記述。

宿題

CK 機械について，以下の問題に答えたレポートを，moodle から投入せよ．(締め切りは，次回授業の前日 24:00)

- $e =$ $\mathtt{Plus}(\mathtt{Int}(1), \mathtt{Times}(\mathtt{Int}(2), \mathtt{Times}(\mathtt{Int}(3), \mathtt{Plus}(\mathtt{Int}(4), \mathtt{Int}(5)))))$ に対して，初期状態 $\langle \mathit{eval}, e, \mathtt{init} \rangle$ からの状態遷移を記述せよ．(図で書く必要はなく，$\langle \mathit{eval}, e, K \rangle$ や $\langle \mathit{apply}, K, n \rangle$ の形の状態の遷移のみ書けばよい．)
- (余力がある人のみ) この機械で，足し算の計算順序を逆にするにはどうしたらよいか？

宿題 1 の解答例 (1/2)

$e = \mathtt{Plus}(\mathtt{Int}(1), \mathtt{Times}(\mathtt{Int}(2), \mathtt{Times}(\mathtt{Int}(3), \mathtt{Plus}(\mathtt{Int}(4), \mathtt{Int}(5)))))$ の計算．(長いので Plus を P, Times を T 等と省略する)

$$
\begin{aligned}
&\langle \mathit{ev}, e, \mathtt{init} \rangle \\
&\to \langle \mathit{ev}, I(1), p((p1, T(I(2), T(I(3), P(I(4), I(5))))), \mathtt{init}) \rangle \\
&\to \langle \mathit{ap}, p((p1, T(I(2), T(I(3), P(I(4), I(5))))), \mathtt{init}), 1 \rangle \\
&\to \langle \mathit{ev}, T(I(2), T(I(3), P(I(4), I(5)))), p((p2, 1), \mathtt{init}) \rangle \\
&\to \langle \mathit{ev}, I(2), p((t1, T(I(3), P(I(4), I(5)))), p((p2, 1), \mathtt{init})) \rangle \\
&\to \langle \mathit{ap}, p((t1, T(I(3), P(I(4), I(5)))), p((p2, 1), \mathtt{init})), 2 \rangle \\
&\to \langle \mathit{ev}, T(I(3), P(I(4), I(5))), p((t2, 2), p((p2, 1), \mathtt{init})) \rangle \\
&\to \langle \mathit{ev}, I(3), p((t1, P(I(4), I(5))), p((t2, 2), p((p2, 1), \mathtt{init}))) \rangle \\
&\to \langle \mathit{ap}, p((t1, P(I(4), I(5))), p((t2, 2), p((p2, 1), \mathtt{init}))), 3 \rangle \\
&\to \langle \mathit{ev}, P(I(4), I(5)), p((t2, 3), p((t2, 2), p((p2, 1), \mathtt{init}))) \rangle
\end{aligned}
$$

宿題 1 の解答例 (2/2)

$$
\begin{aligned}
&\to \langle \mathit{ev}, P(I(4), I(5)), p((t2, 3), p((t2, 2), p((p2, 1), \mathtt{init}))) \rangle \\
&\to \langle \mathit{ev}, I(4), p((p1, I(5)), p((t2, 3), p((t2, 2), p((p2, 1), \mathtt{init})))) \rangle \\
&\to \langle \mathit{ap}, p((p1, I(5)), p((t2, 3), p((t2, 2), p((p2, 1), \mathtt{init})))), 4 \rangle \\
&\to \langle \mathit{ev}, I(5), p((p2, 4), p((t2, 3), p((t2, 2), p((p2, 1), \mathtt{init})))) \rangle \\
&\to \langle \mathit{ap}, p((p2, 4), p((t2, 3), p((t2, 2), p((p2, 1), \mathtt{init})))), 5 \rangle \\
&\to \langle \mathit{ap}, p((t2, 3), p((t2, 2), p((p2, 1), \mathtt{init}))), 9 \rangle \\
&\to \langle \mathit{ap}, p((t2, 2), p((p2, 1), \mathtt{init})), 27 \rangle \\
&\to \langle \mathit{ap}, p((p2, 1), \mathtt{init}), 54 \rangle \\
&\to \langle \mathit{ap}, \mathtt{init}, 55 \rangle \\
&\to 55
\end{aligned}
$$

宿題 2(optional) の解答例

問 (余力がある人のみ) この機械で，足し算の計算順序を逆にするにはどうしたらよいか？
解答例．この CK 機械では，$\texttt{Plus}(e_1, e_2)$ の計算の際に (定義の 2 行目)，e_2 をスタックに積み，e_1 の計算を始めるので,「+ の左にある式を先に計算する」方式である．
これを「+ の右にある式を先に計算する」方式に変更するためには，定義の 2 行目を以下のように変更するとよい．

$$\langle \mathit{eval}, \texttt{Plus}(e_1, e_2), K\rangle \to \langle \mathit{eval}, e_2, \ \texttt{push}((\mathit{plus}1, e_1), K)\rangle$$

なお，この変更方法では，$e_1 \dot{+} e_2 = e_2 \dot{+} e_1$ であることを仮定している．これが不成立の場合 (たとえば，引き算の演算子のとき) は，5 行目の，$n_2 \dot{+} n_1 = n$ という計算も，$n_1 \dot{+} n_2 = n$ に変更する必要がある．