

プログラミング言語論



第14回

プログラムの意味論と検証(3)

不動点意味論

担当: 犬塚

今日の講義

- これまでに簡単なプログラミング言語について、表示的意味論を与えた。
- 本講義では、再帰呼び出しで定義される関数についての表示的意味を与える。
- この理論は**不動点意味論**と呼ばれる。
- この意味論のために導入する**領域理論**と合わせて、**D.Scott**が与えた。

再帰

- 再帰は、基本的な制御構造である。
- チューリング機械と同様によく用いられる計算の原理である、**帰納的関数論** (recursive function theory) では、再帰が基本的制御構造。
- 他のループも再帰に還元できる。
- 再帰は、内側の呼出しでサイズの小さなパラメータで呼び出すように書けば確実に停止するので、扱いの容易な制御構造である。

この講義で導入するアイデア

- 再帰定義を**汎関数**を用いて再定義する。
- 再帰的プログラムの意味を、**汎関数の不動点**として捉える。
- 再帰的関数の表示的意味としてふさわしい、関数の領域を用意する。
 - **連続関数**
- 領域の上で、汎関数の不動点が存在することを示す
= **不動点定理**

再帰

$$\square f(x, y) = \begin{cases} \text{if } x=0 & \text{then } y \\ & \text{else } f(x-1, 2*y) \end{cases}$$

$$f(3,3)=f(2,6)=f(1,12)=f(0,24)=24$$

$$\square g(x,y) = \begin{cases} \text{if } x=y & \text{then } y+1 \\ & \text{else } g(x, g(x-1, y+1)) \end{cases}$$

$$g(4,2)=g(4,g(3,3))=g(4,4)=5$$

$$g(5,1)=g(5,g(4,2))=g(5,g(4,g(3,3)))=g(5,g(4,4))=g(5,5)=6$$

$$g(2,3)=g(2,g(1,4))=g(2,g(1,g(0,5)))=g(2,g(1,g(0,g(-1,6))))=...$$

再帰と汎関数

再帰的関数定義

$$f(x, y) = (\text{if } x=0 \quad \text{then } y \\ \quad \quad \quad \text{else } f(x - 1, 2 * y))$$

この再帰的な関数定義は、汎関数

$$\Gamma = \lambda \varphi x y. (\text{if } x=0 \text{ then } y \text{ else } \varphi(x - 1, 2 * y))$$

を用いると、次の方程式として書くことができる。

$$f = \Gamma(f)$$

練習

次の再帰的関数定義を、汎関数を用いて書き直せ。

$$g(x,y) = (\text{if } x=y \text{ then } y+1 \text{ else } g(x-1, y+1))$$

Γ の不動点

$\Gamma_1 = \lambda \varphi xy. (\text{if } x=0 \text{ then } y \text{ else } \varphi(x-1, 2*y))$

は次の関数を不動点として持つ。

$$f = \lambda xy. 2^x y$$

$$\begin{aligned}\Gamma_1(f) &= \lambda xy. (\text{if } x=0 \text{ then } y \text{ else } f(x-1, 2*y)) \\ &= \lambda xy. (\text{if } x=0 \text{ then } y \text{ else } 2^{x-1} 2*y) \\ &= \lambda xy. (\text{if } x=0 \text{ then } 2^x y \text{ else } 2^x y) \\ &= \lambda xy. 2^x y = f\end{aligned}$$

2つの不動点

次の汎関数は少なくとも2つの不動点をもつ。

$$\Gamma_2 = \lambda \varphi xy. (\text{if } x=y \text{ then } y+1 \text{ else } \varphi(x, \varphi(x-1, y+1)))$$

$$\square f_1 = \lambda xy. x+1$$

$$\begin{aligned} \Gamma_2(f_1) &= \lambda xy. (\text{if } x=y \text{ then } y+1 \text{ else } f_1(x, f_1(x-1, y+1))) \\ &= \lambda xy. (\text{if } x=y \text{ then } x+1 \text{ else } x+1) = f_1 \end{aligned}$$

$$\square f_2 = \lambda xy. (\text{if } x \geq y \text{ then } x+1 \text{ else } y-1)$$

$$\begin{aligned} \Gamma_2(f_2) &= \lambda xy. (\text{if } x=y \text{ then } y+1 \text{ else } f_2(x, f_2(x-1, y+1))) \\ &= \lambda xy. (\text{if } x=y \text{ then } y+1 \text{ else } f_2(x, \{\text{if } x-1 \geq y+1 \text{ then } x \text{ else } y\})) \\ &= \lambda xy. (\text{if } x=y \text{ then } y+1 \text{ else } f_2(x, \{\text{if } x \geq y+2 \text{ then } x \text{ else } y\})) \\ &= \lambda xy. (\text{if } x=y \text{ then } y+1 \text{ else} \\ &\quad \{\text{if } x \geq y+2 \text{ then } x+1 \text{ else if } x \geq y \text{ then } x+1 \text{ else } y-1\})) \\ &= \lambda xy. (\text{if } x \geq y \text{ then } x+1 \text{ else } y-1) = f_2 \end{aligned}$$

汎関数によって定義される関数

- 関数 h について $x = h(x)$ の解 を、不動点(fixed point)という。
- x が関数 h の不動点であるとは、この x にこの関数を施しても値が変わらないということ。
- 再帰的に定義される関数は、汎関数 Γ の方程式 $f = \Gamma(f)$ の解と考えられる。つまり、 Γ の不動点である。

疑問点:

- どんな Γ は不動点をもつことが保証できるのか？
- 2つ以上の不動点を持つことはないのか？2つ以上不動点をもつ場合、どれが定義される関数なのか？

無限個の不動点をもつ汎関数の例

$\Gamma_3 = \lambda \varphi x. (\text{if } x=0 \text{ then } 1 \text{ else } \varphi(x+1))$

□ この汎関数は、以下のものすべてを不動点として持つ。

■ $f_0 = \lambda x. (\text{if } x=0 \text{ then } 1 \text{ else } 0)$

■ $f_1 = \lambda x. (\text{if } x=0 \text{ then } 1 \text{ else } 1)$

■ $f_2 = \lambda x. (\text{if } x=0 \text{ then } 1 \text{ else } 2)$

■ $f_3 = \lambda x. (\text{if } x=0 \text{ then } 1 \text{ else } 3)$

...

□ さらに、次の部分関数も不動点と見なせる。

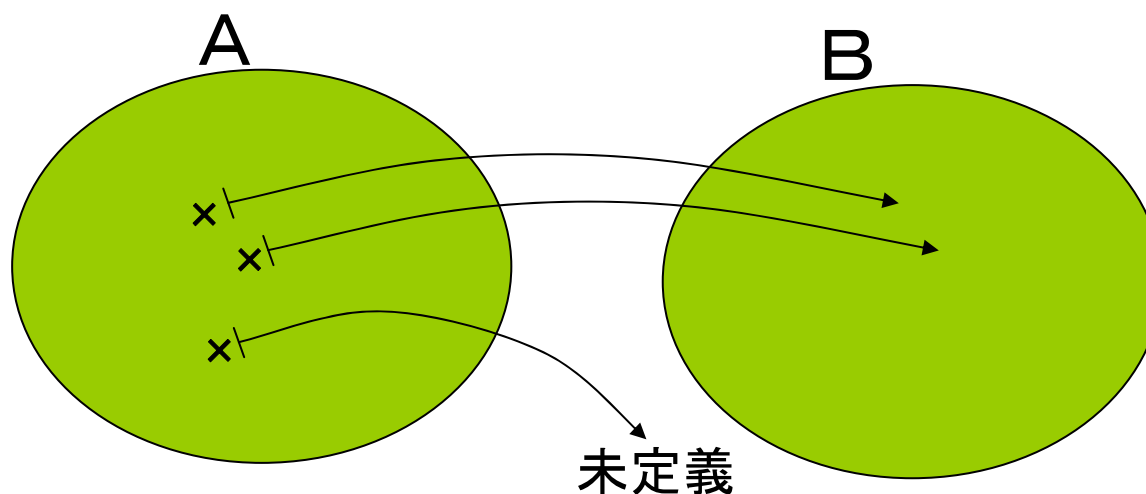
■ $f_u = \lambda x. (\text{if } x=0 \text{ then } 1 \text{ else } \text{未定義})$

実際には、上の Γ_3 に対応する次のプログラムでは f_u がまさに、このプログラムの意味にふさわしい。

$f(x) = (\text{if } x=0 \text{ then } 1 \text{ else } f(x+1))$

部分関数

- プログラムとしての関数は、その計算が停止しない場合、**未定義 (undefined)** であると考えられる。
- 即ち、プログラムの意味は、未定義の部分を含んだ関数 = **部分関数 (partial function)** として与えなければならない。
- 未定義の箇所のない関数は **全域関数 (total function)**。

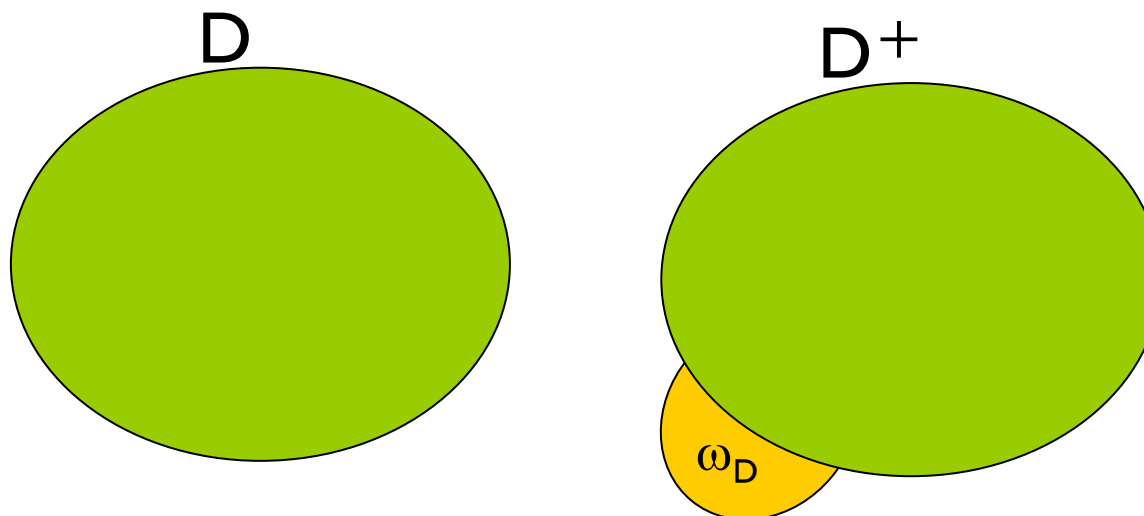


定義域、値域の拡張

- 関数の定義域、値域には未定義要素も含めて考えるのが便利である。
- そこで、集合Dに対して次の集合を定義することにする。

$$D^+ = D \cup \{\omega_D\}$$

- こうした集合を領域 (domain) という。



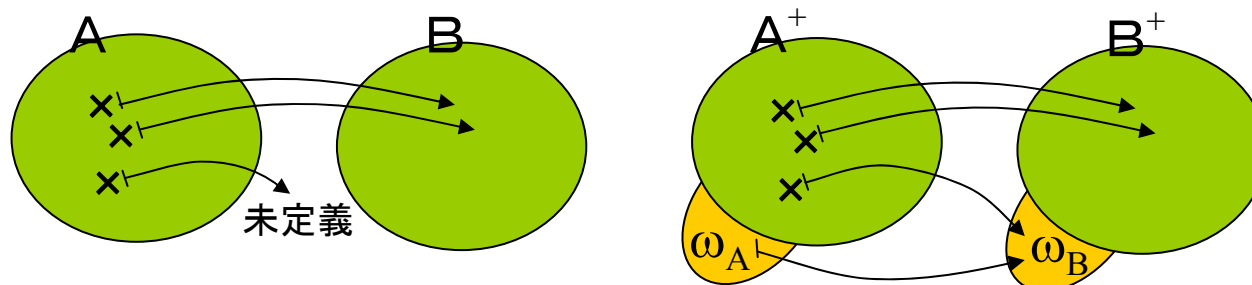
自然な拡張

未定義を明示的に扱うため、部分関数の扱いを再構成する。

□ 定義域A、値域Bの部分関数 $f: A \rightarrow B$

に対して次のとおり、全域関数 $f^+: A^+ \rightarrow B^+$ を与える。

$$f^+(x) = \begin{cases} f(x) & ; f(x) \text{ が定義されているとき} \\ \omega_B & ; x = \omega_A \text{ か、または } f(x) \text{ が未定義のとき} \end{cases}$$

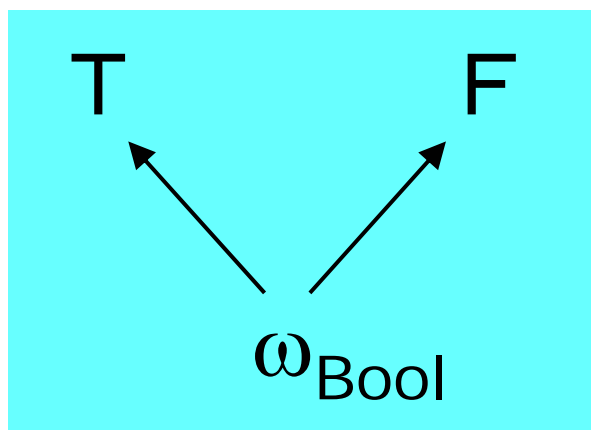


□ f^+ は f の自然な拡大 (natural extension) という。

領域の上の半順序関係

- 領域 D^+ 上で、未定義の値から定義の値への順序 \sqsubseteq を考える。
すなわち、 $x, y \in D^+$ について、
 $x = \omega_D$, または $x = y$ のときそのときに限り $x \sqsubseteq y$

例 Bool = {T, F}としたとき、 $\text{Bool}^+ = \{T, F, \omega_{\text{Bool}}\}$ で、次のハッセ図で表される順序をもつ。



2引数の関数

- $f(x, y) = x + y$ のような2引数関数の定義域に対する領域を与える。
- $D_1 \times D_2$ に対応する領域は次のとおり。

$$(D_1 \times D_2)^+ = D_1^+ \times D_2^+$$

- $(A \times B)^+$ 上の半順序を次のとおり定義する。

$(x, y), (x', y') \in (D_1 \times D_2)^+$ について、

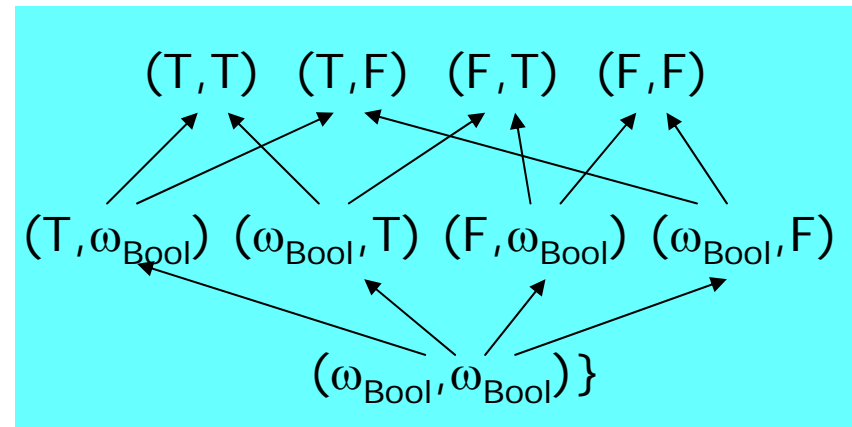
$x \sqsubseteq x'$ かつ $y \sqsubseteq y'$ のとき、そのときに限り $(x, y) \sqsubseteq (x', y')$

例 Bool = {T, F} のとき、

$(\text{Bool} \times \text{Bool})^+$

$= \{ (T, T), (T, F), (T, \omega_{\text{Bool}}), (F, T), (F, F), (F, \omega_{\text{Bool}}), (\omega_{\text{Bool}}, T), (\omega_{\text{Bool}}, F), (\omega_{\text{Bool}}, \omega_{\text{Bool}}) \}$

これは右のハッセ図で表される順序をもつ。



関数の単調性

- 通常通り定義された関数は領域上の順序に関して単調性をもつ。

単調性(monotonic): $x \sqsubseteq y$ のとき、 $f(x) \sqsubseteq f(y)$

例 通常、関数の引数が未定義なら関数の返り値は未定義。

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, \quad f(x, y) = x + y$$

$$f^+ : (\mathbb{N} \times \mathbb{N})^+ \rightarrow \mathbb{N}^+, \quad f^+(x, y) = x + y$$

f^+ は \sqsubseteq について単調である。

(\because) $(x, y) \sqsubseteq (x', y')$ とする。

x, y がどちらも $\omega_{\mathbb{N}}$ でなければ $(x, y) = (x', y')$ 。よって $f^+(x, y) \sqsubseteq f^+(x', y')$

x, y のどちらかが $\omega_{\mathbb{N}}$ なら $f^+(x, y) = \omega_{\mathbb{N}}$ なのでやはり $f^+(x, y) \sqsubseteq f^+(x', y')$

練習

$g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, $g(x, y) = \text{if } x=0 \text{ then } x \text{ else } x+y$

も単調性を持つことを確認せよ。

この場合は、もう少し場合わけが必要。

$(x, y) \sqsubseteq (x', y')$ とする。

1) $x = \omega_{\mathbb{N}}$ のとき

2) $x = 0$ のとき

3) x が $0, \omega_{\mathbb{N}}$ のどちらでもなく、 $y = \omega_{\mathbb{N}}$ のとき

4) x が $0, \omega_{\mathbb{N}}$ のどちらでもなく、 y も $\omega_{\mathbb{N}}$ 以外の値のとき

計算される関数の候補

- 我々の表示的意味論として考えるべき関数は、単調な関数だけでよい。
 - 単調でない関数は不自然である：ある引数が未定義であるかどうかで、答えが変わるのはおかしい。

そこで、

$$(D_1 \rightarrow D_2)^+ = \{ f^+ \mid f^+ : D_1^+ \rightarrow D_2^+ \text{ は単調} \}$$

関数の間の順序

- 汎関数も同様の議論で扱うには、汎関数の定義域＝関数の集合にも順序を導入する必要がある。

- $f_1, f_2 \in (D_1 \rightarrow D_2)^+$ について、
任意の $x \in D_1^+$ で $f_1(x) \sqsubseteq f_2(x)$ であるときそのときに限り、

$$f_1 \sqsubseteq f_2$$

- つまり、 $f_1 \sqsubseteq f_2$ であるとは、

f_1 が未定義でない限り $f_1(x) = f_2(x)$ であるということ。

($f_1(x)$ が未定義の場合は、 $f_2(x)$ はどんな値を持っていてもよい)

鎖、一意な極限

- D^+ の要素の(無限)列 x_0, x_1, x_2, \dots が $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \dots$ となるとき、この列を D^+ の鎖 (chain) という。
- D^+ の鎖 x_0, x_1, x_2, \dots に対し、つぎの x をこの鎖の一意な極限 (unique limit) という。
 - $x_i \sqsubseteq x, i=0, 1, \dots$
 - y が $x_i \sqsubseteq y, i = 0, 1, \dots$ をみたすなら $x \sqsubseteq y$
(x は鎖のどの要素より大きく、そういうものの中で一番小さい)
- 鎖 x_0, x_1, x_2, \dots の一意な極限を $\lim_{i \rightarrow \infty} x_i$ と書く。

領域の性質

- これまで検討してきた関数の領域は次の性質 D^+ を持つ。
 - D^+ は \sqsubseteq に関して半順序集合である。
 - D^+ は最小の要素を持つ。
 - D^+ のすべての鎖が、一意な極限をもつ。

こうしたことは、 $(D_1 \times D_2)^+$ 、 $(D_1 \rightarrow D_2)^+$ にも言える。

- ここで改めて、上の性質を持つとき **領域** ということにする。

連続

- D_1^+ 、 D_2^+ を領域、 $f : D_1^+ \rightarrow D_2^+$ を単調な関数とする。
- このとき、 D_1^+ の任意の鎖、 x_0, x_1, x_2, \dots を考えると、 f は単調であるので、 $f(x_0), f(x_1), f(x_2), \dots$ も鎖である。
- D_2^+ は領域なので、任意の鎖は一意的な極限をもつ。

このとき次が成立つ場合、 f は連続(continuous)であるという。

$$f\left(\lim_{i \rightarrow \infty} x_i\right) = \lim_{i \rightarrow \infty} f(x_i)$$

解析での連続と、ここでの連続

- ここで用いた連続の概念は、特別のものでない。
- 普通に用いる連続の概念は、実はここで述べた連続の特別なケース。

通常連続は次のように定義される。

- \mathbb{R} (実数)の点列 a_0, a_1, a_2, \dots は次のとき**極限** a を持つ。
任意の $\varepsilon > 0$ に対して、ある N があり、すべての $i > N$ で
 $|a - a_i| < \varepsilon$ 。
- 関数 $f: \mathbb{R} \rightarrow \mathbb{R}$ が**連続**であるとは、極限をもつ任意の \mathbb{R} の点列 a_0, a_1, a_2, \dots に対し、 $f(\lim_{i \rightarrow \infty} a_i) = \lim_{i \rightarrow \infty} f(a_i)$ 。

不動点定理

- D^+ を領域、 $f: D^+ \rightarrow D^+$ を連続関数とする。
- このとき f は、

$$x = \lim_{n \rightarrow \infty} f^n(\omega_D)$$

によって与えられる不動点 x を持つ。

- ただし、 ω_D は D^+ の最小要素。
- さらに、この x は最小の不動点である。

不動点定理の証明(1)

- まず $f^i(\omega_D) \sqsubseteq f^{i+1}(\omega_D)$ を i に関する数学的帰納法で示す。
 $i=0$ の場合、 ω_D は最小要素なので $\omega_D \sqsubseteq f(\omega_D)$ 。
 $f^i(\omega_D) \sqsubseteq f^{i+1}(\omega_D)$ ならば f の単調性から $f^{i+1}(\omega_D) \sqsubseteq f^{i+2}(\omega_D)$
したがって、 $\omega_D \sqsubseteq f(\omega_D) \sqsubseteq f^2(\omega_D) \sqsubseteq \dots$ 示された。
- したがってこれらは鎖となり、一意な極限 $x = \lim_{i \rightarrow \infty} f^i(\omega_D)$ がある。
すると、 f の連続性から、
$$f(\lim_{i \rightarrow \infty} f^i(\omega_D)) = \lim_{i \rightarrow \infty} f^{i+1}(\omega_D) = \lim_{i \rightarrow \infty} f^i(\omega_D)$$
- つまり、 x は不動点である。

不動点定理の証明(2)

つぎに x が最小の不動点であることを示す。

- y を f のある不動点とする。
- すると ω_D は最小要素なので $\omega_D \sqsubseteq y$ 。
- f の単調性から $f(\omega_D) \sqsubseteq f(y) = y$
- 同様に任意の i について $f^i(\omega_D) \sqsubseteq y$ である。
- したがって、極限の性質より

$$x = \lim_{i \rightarrow \infty} f^i(\omega_D) \sqsubseteq y$$

(以上)

まとめ

- 再帰的プログラムの意味を与える不動点意味論を紹介した。
- 再帰的プログラムは汎関数の不動点として定義できる。
- 関数が不動点を持つための定理＝不動点定理を利用して、再帰定義が定義するものを明らかにすることができる。
- 不動点定理が述べる最小不動点は確かに、プログラムの動きと一致する。