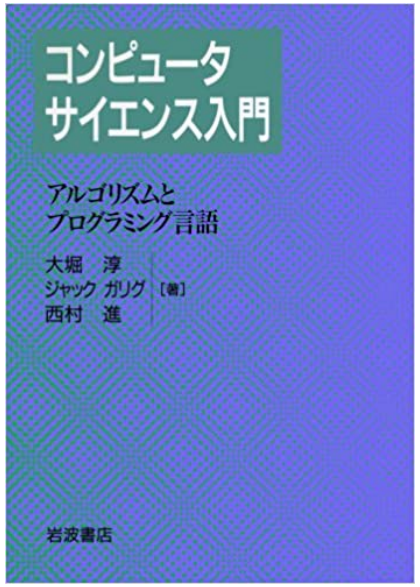
コンピュータサイエンス入門
論理とプログラム意味論

輪読するテキスト

- テキスト名：コンピュータサイエンス入門
  - 〈1〉アルゴリズムとプログラミング言語
  - 〈2〉論理とプログラム意味論
- 京都大学 1回生からの学部学生対象
  - 理科系向きの一般教養科目のテキスト
- 4分野のコンピュータサイエンスの初歩
  - アルゴリズム，プログラミング言語，論理学，プログラム意味論
- プログラミング言語Minimalの実装例
  - https://github.com/tetsuo-jp/Minimal



1. 序論 p.1-

- プログラミング言語の理論
  - コンピュータサイエンスの主要な分野の1つ
  - アプローチ：論理学的，意味論的

- 本書の構成

  第Ⅰ部　数理論理学入門
  第Ⅱ部　プログラムの不動点意味論
  第Ⅲ部　プログラミング論理入門
  第Ⅳ部　リアクティブシステムのための時相論理

§1.1 プログラミング言語とその理論

- 本書の主題：
  プログラミング言語と計算の分析

- 議論・推論するための考え方
  - 論理的な考え方
  - 意味論的な考え方

§1.2 数理論理学と形式化の技法

- 「正しさ」や「証明」とは何か？　→哲学的問題には答えない

- 数理論理学における「正しさ」と「証明」
- 「証明」
  - 形式的に，ある一定のルールにしたがって得られるもの
- 「正しい」
  - ある定められた数学的な構造に照らし合わせて，ある命題が正しいか否かを定める
- 「証明できる」≠「正しい」
- 形式化の効果は絶大

§1.3 言語の意味論

- 言語学では言語を構文論と意味論から扱う

- 構文論≒文法
  - アルファベットの要素をどのように並べればその言語の正しい文章ができるかを定めた規則(およびそれに関する理論)
  - 例．”John love Mary”は正しくない，”John loves Mary”は正しい
  - 例．交通信号の言語＝{赤，青，黄}
- 意味論
  - 正しい文章に何らかの人間の概念(人が考えうる対象)を対応させること
  - 例．”John loves Mary”≒「ジョンはメアリにほれている」
  - 例．赤は＜止まれ＞，青は＜進め＞，黄は＜早く行け＞

- プログラミング言語では厳密に構文論と意味論を展開できる

§1.4 プログラミング言語の意味論
(a)プログラミング言語petitの構文

構文領域

ξ : **Var**　プログラム(の中で用いられる)変数の記号全体

ε : **Exp**　式(expression)の全体

τ : **Pro**　プログラムの全体

§1.4 プログラミング言語の意味論
(a)プログラミング言語petitの構文

構文規則

(i) $\xi ::= \mathsf{A} \mid \mathsf{B} \mid \cdots \mid \mathsf{Z}$

(ii) $\varepsilon ::= 0 \mid \xi \mid \mathsf{suc}\ \varepsilon$

(iii) $\tau ::= \xi := \varepsilon \mid \tau;\tau \mid \mathsf{for}\ \varepsilon\ \mathsf{times\ do}\ \tau\ \mathsf{end}$

プログラムの例

X:=A; X:=B; for C times do X:=suc X; Y:=suc Y end

§1.4 プログラミング言語の意味論
(b) petitの意味論

表 1.1　petit の直感的な意味

構文要素	意味
0	整数値 0
suc 0	整数値 1
suc A	プログラム変数 A が現在もつ値に 1 を加えて得られる整数値
X:=ε	変数 X に ε の値を入れる
$\tau_0;\tau_1$	まず τ_0 を実行し，その後に τ_1 を実行する
for ε times do τ end	τ を ε 回実行する

§1.4 プログラミング言語の意味論 (b) petitの意味論

意味領域

$\mathbf{N}$	自然数の全体 (0 を含む)
$\mathbf{S} = (\mathbf{Var} \to \mathbf{N})$	コンピュータの状態全体
$\mathbf{C} = (\mathbf{S} \to \mathbf{S})$	コンピュータの状態の変換全体

§1.4 プログラミング言語の意味論
(b) petitの意味論

<u>意味関数</u>：構文領域から意味領域への関数

$$\mathcal{E} : \mathbf{Exp} \to (\mathbf{S} \to \mathbf{N})$$

$$\mathcal{C} : \mathbf{Pro} \to \mathbf{C}$$

意味領域上を動くメタ変数

$$\nu : \mathbf{N}, \quad \sigma : \mathbf{S}, \quad \theta : \mathbf{C}$$

§1.4 プログラミング言語の意味論
(b) petitの意味論

意味関数 ε

$$(\mathrm{i}) \quad \mathcal{E}[\![0]\!](\sigma) = 0$$
$$(\mathrm{ii}) \quad \mathcal{E}[\![\xi]\!](\sigma) = \sigma(\xi)$$
$$(\mathrm{iii}) \quad \mathcal{E}[\![\mathrm{suc}\ \varepsilon]\!](\sigma) = \mathcal{E}[\![\varepsilon]\!](\sigma) + 1$$

- 構文領域の引数は二重括弧[[ ]]で明示する.
- E[[ε]]やC[[τ]]をそれぞれ ε と τ の表示という.
- 表示 E[[ε]] : S → N は関数

§1.4 プログラミング言語の意味論
(b) petitの意味論

(i) $\mathcal{E}[\![0]\!](\sigma) = 0$

(ii) $\mathcal{E}[\![\xi]\!](\sigma) = \sigma(\xi)$

(iii) $\mathcal{E}[\![\mathrm{suc}\ \varepsilon]\!](\sigma) = \mathcal{E}[\![\varepsilon]\!](\sigma) + 1$

例 1.1 式 suc suc 0 の表示を計算してみよう.

$$\begin{aligned}
\mathcal{E}[\![\mathrm{suc\ suc\ 0}]\!](\sigma) &\overset{\text{(iii)}}{=} \mathcal{E}[\![\mathrm{suc\ 0}]\!](\sigma) + 1 \\
&\overset{\text{(iii)}}{=} (\mathcal{E}[\![0]\!](\sigma) + 1) + 1 \\
&\overset{\text{(i)}}{=} (0 + 1) + 1 \\
&= 2
\end{aligned}$$

すなわち，式 suc suc 0 の表示は，コンピュータの状態 σ に依存せず 2 になることがわかる. □

§1.4 プログラミング言語の意味論 (b) petitの意味論

意味関数C

(iv) $C[\![\xi := \varepsilon]\!](\sigma) = update(\xi, \mathcal{E}[\![\varepsilon]\!](\sigma))(\sigma)$

(v) $C[\![\tau_0; \tau_1]\!](\sigma) = C[\![\tau_1]\!](C[\![\tau_0]\!](\sigma))$

(vi) $C[\![\text{for } \varepsilon \text{ times do } \tau \text{ end}]\!](\sigma) = iterate(C[\![\tau]\!], \mathcal{E}[\![\varepsilon]\!](\sigma))(\sigma)$

ここで，関数 $update : \mathbf{Var} \times \mathbf{N} \to \mathbf{C}$ と $iterate : \mathbf{C} \times \mathbf{N} \to \mathbf{C}$ は次のように定義する．

$$update(\xi, \nu)(\sigma) = \sigma\{\xi : \nu\}$$

$$iterate(\theta, \nu) = \underbrace{\theta \circ \theta \circ \ldots \circ \theta}_{\nu\text{回}}$$

ただし，$\sigma \in \mathbf{S}$ に対し，$\sigma\{\xi : \nu\} \in \mathbf{S}$ は

$$\sigma\{\xi : \nu\}(\eta) = \begin{cases} \nu & if \ \eta = \xi \\ \sigma(\eta) & if \ \eta \neq \xi \end{cases}$$

また，$\circ$ は関数の合成を表す $((g \circ f)(x) = g(f(x)))$．

§1.4 プログラミング言語の意味論 (c)表示的意味論

以上の意味の与え方は
プログラミング言語の表示的意味論という.

その原則は以下の通り：

> プログラミング言語の意味は，$\mathbf{Syn}$ をその言語の構文領域とすると，意味領域 $\mathbf{D}$ と意味関数 $\mathcal{V} : \mathbf{Syn} \to \mathbf{D}$ の対 $\langle \mathbf{D}, \mathcal{V} \rangle$ によって規定される．構文要素 $s \in \mathbf{Syn}$ に対し，$\mathcal{V}(s)$ (いままで使用した記法によれば $\mathcal{V}[\![s]\!]$) を s の表示とよぶ．