ソフトウェア基礎論配布資料 (1)
はじめに

五十嵐 淳
京都大学 大学院情報学研究科知能情報学専攻
e-mail: igarashi@kuis.kyoto-u.ac.jp

平成 19 年 10 月 4 日

1 各種事務連絡

担当教員について

五十嵐 淳
居室　工学部 10 号館 1 階 142 号室
URL　http://www.sato.kuis.kyoto-u.ac.jp/~igarashi/class/sf/
e-mail　fsoft07@sato.kuis.kyoto-u.ac.jp

講義カレンダー

10 月	2, 9, 16, 23(休講予定), 30	12 月	4, 11, 18
11 月	6, 13, 20, 27	1 月	8(休講？), 15(休講予定), 22, 29

成績評価

レポート数回

2 講義内容

シラバスより：

> 数理論理学的手法を用いたソフトウェア科学の基礎理論について講述する．特に、プログラミング言語の形式化と意味論、形式化を用いてプログラムの性質 (型システムとプログラムの型安全性など) に関する議論をする．

ソフトウェア科学の基礎理論の目標 (のひとつ)

「思った通りに動くソフトウェアを作る」ための理論

or

ソフトウェアが思った通りに動くことを数学的に「証明」する技法の確立

- 「思った通り」… 意図を曖昧性なく表現するための記法: 記号論理・オートマトン・型
- 「ソフトウェアが動く」…プログラム意味論・プログラミング言語のモデル
  - **操作的意味論**(*operational semantics*)…プログラムの**動作**を正確に記述する
  - **表示的意味論**(*denotational semantics*)…プログラムを，よく知っている世界での何者かに写す (例：プログラムは初期メモリ状態から終了メモリ状態への (集合論的) 関数である)
  - **公理的意味論**(*axiomatic semantics*)…プログラム構成要素の性質を**公理化**する (c.f. ユークリッド幾何学)
- 意図とプログラムの照合 = プログラム検証技術
  - テスト
  - 実行時監視
  - 型システム
  - モデル検査
  - 公理的意味論・表示的意味論を使ったプログラムの性質に関する証明

メイン・トピック

1. 種々の計算体系 (プログラミング言語のモデル) とその操作的意味論 (60%)…λ 計算，オブジェクト指向計算，π 計算
2. プログラム検証技法のひとつとしての型システムの理論 (40%)

参考図書

Benjamin C. Pierce. *Types and Programming Languages.* The MIT Press. 2002. λ 計算に対する様々な型システムを解説する教科書．説明は丁寧だが，ぶ厚い．

Robin Milner. *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press, 1999. 並行計算のモデル π 計算に関する教科書．1999 年に出ているわりには内容はやや古い．型については軽く触れられているだけ．

3　意味論における問題を概観する

プログラムの動作・実行手順・プログラム同士の等しさを数学の (特に集合を使った) 言葉で正確に記述する.

Case: 四則演算式 as プログラム

- `(1 + 5) * (3 - 5)` の実行結果は `-12` である. なぜなら,
  - `1 + 5` の実行結果は `6` で,
  - `3 - 5` の実行結果は `-2` で,
  - `6 * (-2)` の実行結果は `-12` だからである.
- `(1 + 5) * (3 - 5)` というプログラムと `(1 + 5) * (-2)` というプログラムは等しく, かつ, 後者の方が,「より簡単な」プログラムである.
- `(1 + 5) * (3 - 5)` というプログラムと `(2 * 4) - (6 + 14)` というプログラムは等しい.

これを数学的に表現するには…

- 「`(1 + 5) * (3 - 5)`」「`6 * (3 - 5)`」などを要素とする四則演算式の集合: $\mathbf{Aexp}$.
- 式と実行結果の関係表す関係: $\Downarrow (\subseteq \mathbf{Aexp} \times \mathbf{Num})$.
- プログラムの等しさを表す四則演算式上の関係: $= (\subseteq \mathbf{Aexp} \times \mathbf{Aexp})$.

を定義する. (でも, どうやって? そもそも等しいって何?)

プログラム意味論のやっかいなところ

言語を対象とした議論 (対象言語とメタ言語)

- 見方の変化: ただの記号列としてのプログラム vs 意味をもったプログラム
  - `+` に「かけ算」という意味を与えて, `9+9` $\longrightarrow$ `81` と定義してもよい.
  - `1+2` $= 3$?
  - $\displaystyle\int_0^1 x^2 dx = \int_0^1 y^2 dy$?
- メタ変数: 対象言語の要素を示すメタ言語の変数. 例: 英文法の講義での S, V, O など.
- セオリー (体系の中で示せることがら) とメタセオリー (体系を外から眺めたときにわかる性質)

  `+` の交換則: $\forall x, y \in \mathbf{Aexp}.\forall z \in \mathbf{Num}.(x\texttt{+}y \longrightarrow \cdots \longrightarrow z \iff y\texttt{+}x \longrightarrow \cdots \longrightarrow z)$

A　数学的準備: 集合に関する基礎知識

A.1　論理の記法

以下で A, B を，**命題**(*proposition*)，つまり何かを述べている文 (statement, 疑問文・命令文ではない) とする．また，変数を含む statement を**述語**(*predicate*) という．述語の真偽は，変数に適当に数を割り当てることで決まる．例: 2 変数述語 $P(x, y) \stackrel{\text{def}}{=} (x \leq y)$．命題や述語に,「かつ」「ならば」といった**論理結合子**(*logical connectives*) や,「任意の〜に対して」「ある〜に対して」といった**限量子**(*quantifier*) を適用することで複雑な命題，述語を構成することができる．この講義では，命題・述語を簡潔に記述するために,「かつ」などの日本語の代わりに,「&」などの記号を用いることが多い．これらの記号を表 1 にまとめておく．

記法	説明
$\neg A$	A ではない．(not)
$A \,\&\, B$	A かつ B である．(and, conjunction)
A or B	A または B である．(or, disjunction)
$A \Longrightarrow B$	もし A ならば B である．(if-then, implication)
$A \iff B$	A ならば B であり，A であるのは B のときのみである．(if-and-only-if, logical equivalence)
$\exists x_1, \ldots, x_n.P(x_1, \ldots, x_n)$	ある $x_1, \ldots, x_n$ に対して $P(x_1, \ldots, x_n)$ である．(there exists)
$\forall x_1, \ldots, x_n.P(x_1, \ldots, x_n)$	任意の $x_1, \ldots, x_n$ について $P(x_1, \ldots, x_n)$ である．(for all)
$\forall x \in X.P(x)$	$\forall x.x \in X \Longrightarrow P(x)$ の略記．x の属する集合を指定する．
$\exists x \in X.P(x)$	$\exists x.x \in X \,\&\, P(x)$ の略記．x の属する集合を指定する．
$\exists! x.P(x)$	$(\exists x.P(x)) \,\&\, \forall y, z.(P(y) \,\&\, P(z) \Longrightarrow y = z)$ の略記．「$P(x)$ を満す x がただひとつ存在する」

表 1: 論理の記法

A.2　集合の記法

この講義で主に用いる集合に関する記法を表 2 にまとめる．

A.3　関係と関数

n 項関係　集合 $X_1, \ldots, X_n$ が与えられたとき，$\mathcal{P}(X_1 \times \cdots \times X_n)$ の要素を $X_1, \ldots, X_n$ 間の n **項関係**(n-*ary relation*) であるという．R が二項関係 (binary relation) であるとき．$(x, y) \in R$ を xRy と表記する．

概念	記法	定義・補足
外延的 (要素の列挙による) 定義	$\{a, b, c, \ldots\}$	
要素である	$a \in X$	
部分集合	$X \subseteq Y$	$\forall z \in X.z \in Y$
集合が等しい	$X = Y$	$X \subseteq Y \,\&\, Y \subseteq X$
空集合	$\emptyset$	$(\forall x.x \notin \emptyset)$
内包的定義	$\{x \in X \mid P(x)\}$	X の要素で，P を満すもの全ての集合
巾 (べき) 集合	$\mathcal{P}(X)$	$\{Y \mid Y \subseteq X\}$
		X の全ての部分集合の集合
和集合	$X \cup Y$	$\{a \mid a \in X \text{ or } a \in Y\}$
集合族の和	$\bigcup X$	$\{a \mid \exists x \in X.a \in x\}$ 但し X は集合の集合
	$\bigcup_{i \in I} x_i$	I は 添字の集合
共通部分	$X \cap Y$	$\{a \mid a \in X \,\&\, a \in Y\}$
集合族の共通部分	$\bigcap X$	$\{a \mid \forall x \in X.a \in x\}$ 但し X は集合の集合
	$\bigcap_{i \in I} x_i$	I は 添字の集合
デカルト積	$X \times Y$	$\{(a, b) \mid a \in X \,\&\, b \in Y\}$
直和	$X \uplus Y$	$\{(0, a) \mid a \in X\} \cup \{(1, b) \mid b \in Y\}$
差	$X \setminus Y$	$\{x \mid x \in X \,\&\, x \notin Y\}$

表 2: 集合の記法

部分関数　以下の条件

$$\forall x, y, y'.((x, y) \in f \,\&\, (x, y') \in f) \Longrightarrow y = y'$$

を満す (二項) 関係 $f \in \mathcal{P}(X \times Y)$ を X から Y への**部分関数**(*partial function*) という．$(x, y) \in f$ であるとき，$f(x) = y$ と表記する．またこのとき f は x において **定義される**(*defined*) といい，x に対して $\forall y.(x, y) \notin f$ のとき x において **未定義**(*undefined*) であるという．

X から Y への部分関数全体の集合を $X \rightharpoonup Y$ と表記する．

全値関数　X から Y への部分関数 y で，$\forall x \in X.\exists y \in Y.f(x) = y$ をみたすものを**全値関数**(*total function*) という．

X から Y への全値関数全体の集合を $X \to Y$ と表記する．

集合 X に対して $Id_X \in X \to X \stackrel{\text{def}}{=} \{(x, x) \mid x \in X\}$ を X 上の**恒等関数**(*identity function*) という．

関数 $f \in X \to Y$ に対して，$\exists g \in Y \to X.(\forall x \in X.g(f(x)) = x \,\&\, \forall y \in Y.f(g(y)) = y)$ が成立するとき，X と Y は **1 対 1 対応**(*1-1 correspondence*) しているという．また，この関数 g を f の **逆関数**(*inverse*) という．また，**Nat** の部分集合と 1 対 1 対応している集合を**可算**(*countable*) であるという．

関数・関係の合成

R を X と Y の間の関係，S を Y と Z の間の関係とする．R と S の**合成**(*composition*) を

$$S \circ R \stackrel{\text{def}}{=} \{(x, z) \mid \exists y \in Y.(x, y) \in R \,\&\, (y, z) \in S\}$$

と定義する．$S \circ R$ は X と Z の間の関係である．関数の合成は関係の合成の定義から自然に導かれる．

同値関係

X 上の関係 $R \in \mathcal{P}(X \times X)$ が以下の条件を満すとき**同値関係**(*equivalence relation*) であるという．

- **反射律**(*reflexivity*): $\forall x \in X.xRx$
- **対称律**(*symmetricity*): $\forall x, y \in X.xRy \Longrightarrow yRx$
- **推移律**(*transitivity*): $\forall x, y, z \in X.(xRy \,\&\, yRz) \Longrightarrow xRz$

～閉包

関係 R が与えられた時に，R を含み「～律」を満たすような最小の関係を作ることができる．そのような関係を,「～閉包」という．

例： $R = \{(n, n+1) \mid n \in \mathbf{Nat}\}$ とする

- R の対称閉包は,「差が 1 である」という関係 $\{(n, n+1) \mid n \in \mathbf{Nat}\} \cup \{(n+1, n) \mid n \in \mathbf{Nat}\}$ である．
- R の推移閉包は関係 $<$ である．
- R の反射推移閉包は関係 $\leq$ である．
- $\{(n, n+5) \mid n \in \mathbf{Nat}\}$ の反射対称推移閉包 (同値閉包) は「5 で割った余りが等しい」という関係である．

A.4 数学的帰納法

「任意の自然数 x に対し，$P(x)$」を示すには

1. $P(0)$ であること
2. 任意の自然数 n に関して，$P(n)$ ならば $P(n+1)$ であること

を示せばよい．これを記号ばかりで書くと

$$(P(0) \,\&\, \forall n \in \mathbf{Nat}.(P(n) \Longrightarrow P(n+1))) \Longrightarrow \forall x \in \mathbf{Nat}.P(x)$$

となる．$P(n+1)$ を示すための仮定 $P(n)$ を帰納法の仮定という．

A.4.1 例: 任意の自然数 n について

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

であることを示せ．