プログラム理論

水谷哲也

授業スケジュール

詳しくはシラバス参照のこと

第1週(04/15)　　プログラムの意味，仕様表現と検証，Hoare論理概説

第2週(04/22)～第6週(5/27)　Hoare論理によるプログラムの検証

(04/29, 05/06は休み，05/09(Fri)は火曜日授業のためこの授業がある)

第7週(06/03)　　配列，手続き等の扱い

第8週(06/10)～第9週(06/17)　　Dijkstraの最弱前条件に基づくプログラム検証

第10週(06/24)　　時相論理等

期末試験　07/01(5限のみの予定)

参考書　林晋：プログラム検証論，共立出版，3200円

絶版のため講義では資料を配布またはダウンロード可能にする

http://www.cs.tsukuba.ac.jp/~mizutani/under_grad/programtheory/

成績　期末試験(50%), 演習およびレポート(50%) 出席を加味

授業に関する連絡事項は原則として授業中に行うが，webでも伝える予定のため，こまめに確認すること．

Web http://www.cs.tsukuba.ac.jp/~mizutani/under_grad/programtheory/

プログラム理論とは

計算についての我々の理解を数学的に厳格にすること,

なかんずくコンピュータ・プログラムについての検証のアート

(かの名高いデバッグの技術)を1つのサイエンスにすることを,

目指す理論

(John McCarthy: A Basis for Mathematical Theory of Computationより)

呼び方 (いずれも1960年代初頭より)

日本	Theory of Program (Program Theory)	Igarashi
アメリカ	Mathematical Theory of Computation	McCarthy
ソ連(ロシア)	Theoretical Programming	Ershov

(このprogrammingはsoftware engineering, software science, computer science といった意味.)

プログラムの検証とは

Theory of Program Verification　プログラム理論の中心課題

検証(verification):　プログラムが「正しく」動作することを確認する作業

数学や論理を用いてある程度厳格に行う.

例題

GCDプログラム　ver. 1

```
begin
  a:=x; b:=y;
  while b≠0 do
        begin
              a:=a mod b;
              c:=a; a:=b; b:=c
        end
end
```

正整数x, yの最大公約数を
求めるプログラム
↓
どのように正しさを
保証するのか?
(演習)

a mod bはaをbで割った余り

すなわち，r=a mod b such that a=b*q+r, 0≤r<|b|

プログラムの正しさを調べるには

充分な数のテストデータを準備し，テストを行う

　調べ尽くしたことになるのか？　有限時間内には調べ尽くせない

　考えられうる「危険」な場合だけテストするにしても

　　入力が2変数の場合100個づつの値の組み合わせだけで10000通りになる

　　3変数だと1000000通り

もとが有名な「ユークリッドのアルゴリズム」を用いているから大丈夫?

　ユークリッドのアルゴリズム自身の信頼性←これは一応大丈夫としても

　プログラムがそれを忠実にコーディングしているかどうか不明

プログラムの正しさを調べるには

充分な数のテストデータを準備し，テストを行う

調べ尽くしたことになるのか？　有限時間内には調べ尽くせない

考えられうる「危険」な場合だけテストするにしても

入力が2変数の場合100個づつの値の組み合わせだけで10000通りになる

3変数だと1000000通り

もとが有名な「ユークリッドのアルゴリズム」を用いているから大丈夫?

ユークリッドのアルゴリズム自身の信頼性←これは一応大丈夫としても

プログラムがそれを忠実にコーディングしているかどうか不明

↓

数理的な検証が必要

GCDプログラム　ver. 1

```
begin
a:=x; b:=y;
while b≠0 do
        begin
                a:=a mod b;
                c:=a; a:=b; b:=c
end
end
```

前提: x, yは非負整数,　どちらかは0でない

whileのi回目の実行後のa, bの値を

a_i, b_iとする.　最初は$a_0=x$, $b_0=y$

i+1回目のループを実行すると

$b_i>0$ (while文が実行されるので)

$a_{i+1}=b_i$,
$b_{i+1}=a_i \bmod b_i$
したがって

$\gcd(x, y)= \gcd(a_0, b_0)=...= \gcd(a_i, b_i)$
$= \gcd(a_{i+1}, b_{i+1})=...$ (Why? →演習)

$0 \le b_{i+1} < |b_i|$より b_iはいずれ0になる.

したがってこのループは必ず停止する.

$\gcd(a, 0)=a$

定理

$$\gcd(a_i, b_i) = \gcd(a_{i+1}, b_{i+1})$$

ただし $a_{i+1}=b_i$, $b_{i+1}=a_i \bmod b_i$

例題

GCDプログラム　ver. 1

```
begin
  a:=x; b:=y;
  while b≠0 do
        begin
              a:=a mod b;
              c:=a; a:=b; b:=c
  end
end
```

例題

GCDプログラム　ver. 1

```
begin
  a:=x; b:=y;
  while b≠0 do
        begin
              a:=a mod b;
              c:=a; a:=b; b:=c
  end
end
```

正整数x, yの最大公約数を
求めるプログラム
正しいことが一応証明された
↓
GCDは負の整数についても
定義される
↓
x, yが負の場合はどうなる?

例題

GCDプログラム　ver. 1

```
begin
  a:=x; b:=y;
  while b≠0 do
        begin
              a:=a mod b;
              c:=a; a:=b; b:=c
  end
end
```

正整数x, yの最大公約数を
求めるプログラム
正しいことが一応証明された
↓
GCDは負の整数についても
定義される
↓
x, yが負の場合はどうなる?

プログラムが正しいとは?
↓
プログラムが仕様(Specification)に
合致していること

例題

GCDプログラム　ver. 2

```
begin
  a:=x; b:=y;
  while b≠0 do
        begin
              a:=a mod b;
              c:=a; a:=b; b:=c
  end ;
  if a<0 then a:=-a fi
end
```

仕様

整数x, yの最大公約数を

求めるプログラム

但しx, yのどちらかは0でないとする

演習

このプログラムが正しいことを示せ.

但し，今まで示した

「ver.1が正整数x, yのGCDを計算する

正しいプログラムである」

という事実は

定理として使っても良い

Hoare論理概説

1960年代に考案された手続き型プログラムのための形式的証明体系

現在でもプログラム検証の基礎として有効である

ほとんどの検証体系はこの論理に影響を受けている

形式的検証体系(formal system)なので,

プログラムを(前に述べたようなプログラムの意味を考えるのではなく)

形式的に検証することができる

とりあえず，Hoare論理の概説を行い，厳密な定義は授業の中盤以降で行う.

表明付きプログラム

Asserted Program

プログラムに，表明(assertion)を対応させたもの

{A}P{B}または<A>P<B>の形

A, B: 表明(assertion)　プログラム変数を含む論理式

P: プログラム

<A>P<B>の意味

プログラムPを，条件Aが成立つ状態で実行させると，必ず正常終了し，

停止後に条件Bが成立つ.

A: 前条件(precondition)，B: 後条件(postcondition)

例:　GCDプログラムver. 2をPと表すことにすると

<(x≠0∨y≠0)∧Z(x)∧Z(y)> P <a=GCD(x, y)>

表明付きプログラム

Asserted Program

{A}P{B}の意味

プログラムPを，条件Aが成立つ状態で実行させると,

もし正常終了するならば,

停止後に条件Bが成立つ.

部分的正当性(partial correctness)

(完全)正当性(total correctness)：部分的正当性＋停止性(termination)

例

{(x≠0∨y≠0)∧Z(x)∧Z(y)}　while 0=0 do a:=0　{a=GCD(x, y)}

表明付きプログラムの正しさの証明

プログラムを分割して正しさを示す

例　GCDプログラム　ver. 2

1:　a:=x;
2:　b:=y;
3:　**while** b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
4:　**if** a<0 **then** a:=-a **fi**

3はさらに分割する

3.1: a:=a **mod** b;
3.2: c:=a;
3.3: a:=b;
3.4: b:=c

各々を小さなプログラムだと考え，それぞれに表明をつける

表明付きプログラムの正しさの証明

各々を小さなプログラムだと考え，それぞれに表明をつける

例　GCDプログラム　ver. 2

1:　{x≠0∨y≠0} a:=x{(x≠0∨y≠0)∧a=x}

2:　{(x≠0∨y≠0)∧a=x} b:=y{(x≠0∨y≠0)∧a=x∧b=y}

3:　{(x≠0∨y≠0)∧a=x∧b=y}

　　while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end**

　　{|a|=gcd(x, y)}

4:　{|a|=gcd(x, y)} **if** a<0 **then** a:=-a **fi** {a=gcd(x, y)}

形式的体系 (formal system)

公理

妥当(valid)な命題

推論規則

一つ以上の妥当な命題から

他の妥当な命題への写像

命題が**証明可能(provable)**，**演繹可能(deducible)**，**定理(theorem)**

公理，推論規則の適用の繰り返し(**証明(proof)**)から

得られる命題

Hoare論理の体系

複合文の規則 (composition rule)

$$\frac{\{A\}P_1\{S_1\} \quad \{S_1\}P_2\{S_2\} \quad \ldots \quad \{S_{n-1}\}P_n\{B\}}{\{A\}\ \text{begin}\ P_1; \ldots ; P_n\ \text{end}\ \{B\}}$$

前提

結論

例　P : a:=x; b:=y;
while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
if a<0 **then** a:=-a **fi**

$$\{x \neq 0 \vee y \neq 0\}\ P\ \{a = \gcd(x, y)\}$$

例　P : a:=x; b:=y;
while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
if a<0 **then** a:=-a **fi**

$\{x \neq 0 \vee y \neq 0\}\ P\ \{a = \gcd(x, y)\}$

例　P : a:=x; b:=y;
while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
if a<0 **then** a:=-a **fi**

AP_1

$\{x \neq 0 \vee y \neq 0\}\ P\ \{a = gcd(x, y)\}$

例　P : a:=x; b:=y;
while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
if a<0 **then** a:=-a **fi**

AP_1　　AP_2

$\{x \neq 0 \vee y \neq 0\}\ P\ \{a = \gcd(x, y)\}$

例　P : a:=x; b:=y;
while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
if a<0 **then** a:=-a **fi**

AP_1　　AP_2　　AP_3

$\{x \neq 0 \vee y \neq 0\}\ P\ \{a = \gcd(x, y)\}$

例　P : a:=x; b:=y;
while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
if a<0 **then** a:=-a **fi**

AP_1　　AP_2　　AP_3　　AP_4

$\{x \neq 0 \vee y \neq 0\}\ P\ \{a = \gcd(x, y)\}$

例　P : a:=x; b:=y;
while b≠0 **do begin** a:=a **mod** b; c:=a; a:=b; b:=c **end** ;
if a<0 **then** a:=-a **fi**

AP_1　　AP_2　　AP_3　　AP_4

$$\{x\neq 0 \vee y\neq 0\}\ P\ \{a=gcd(x, y)\}$$

但し　AP_1　　$\{x\neq 0 \vee y\neq 0\}$ a:=x$\{(x\neq 0 \vee y\neq 0) \wedge a=x\}$

AP_2　　$\{(x\neq 0 \vee y\neq 0) \wedge a=x\}$ b:=y$\{(x\neq 0 \vee y\neq 0) \wedge a=x \wedge b=y\}$

$\{(x\neq 0 \vee y\neq 0) \wedge a=x \wedge b=y\}$

AP_3　　**while** b≠0 do **begin** a:=a **mod** b; c:=a; a:=b; b:=c **end**
$\{|a|=gcd(x, y)\}$

AP_4　　$\{|a|=gcd(x, y)\}$ **if** a<0 **then** a:=-a **fi** $\{a=gcd(x, y)\}$

割当公理(代入文の公理)

(assignment axiom)

{A[t/a]}a:=t{A}

　　ただし A[t/a]はA中の変数aをtで置き換えて得られる命題

実行前のaの値をa_0, 実行後をa_1とする

$A[t[a_0/a]/a]$かつ$a_1=t[a_0/a]$ならば$A[a_1/a]$

したがって $\{A[t/a]\}a:=t\{A\}$

例

$\{5=5\}a:=5\{a=5\}$
$\{a+3=10\}a:=a+3\{a=10\}$

実行前のaの値をa_0, 実行後をa_1とする

$A[t[a_0/a]/a]$かつ$a_1=t[a_0/a]$ならば$A[a_1/a]$

したがって $\{A[t/a]\}a:=t\{A\}$

例

$\{5=5\}a:=5\{a=5\}$
$\{a+3=10\}a:=a+3\{a=10\}$

$a_0+3=10 \wedge a_1=a_0+3 \supset a_1=10$

(参考)割当公理(assignment axiom) forward substitution

$$\{A\}\ a:=t\ \{\exists y(A[y/a] \wedge a=t[y/a])\ \}$$

帰結規則(consequence rule)

$$\frac{A \supset B \quad \{B\}P\{C\} \quad C \supset D}{\{A\}P\{D\}}$$

例

$$\frac{\{-x=x_0\}\ x:=-x\{x=x_0\}}{\{|x|=x_0 \wedge x<0\}\ x:=-x\{x=x_0\}}$$

(∵ $|x|=x_0$ かつ $x<0$ ならば $-x=x_0$)

演習

次の表明付きプログラムを証明せよ

{b≠0∧gcd(a, b)=gcd(x, y)}

begin

a:=a **mod** b;

c:=a; a:=b; b:=c

end

{gcd(a, b)=gcd(x, y)}