実践離散数学と計算の理論
第7回　ホーア論理

大阪大学大学院　工学研究科　電気電子情報通信工学専攻
王　イントウ

モチベーション

ソフトウェアの正当性

- ソフトウェアとは、計算方法を示したものである。
- ソフトウェアが正しいとはつまり、ある制約条件が成り立つことである。
- 例：ソートのコードは、すべての入力に対して、出力がソートされるか？

バグ

- ソフトウェアの誤りをバグという。
- バグが原因で人が死ぬ、莫大な経済的損失などがおきる。
- バグが原因の脆弱性は、サイバー攻撃に利用される場合もある。

バグの由来

- 史上初のコンピューター・バグは1945年、『ハーバード・マーク2』のFパネルの70番リレーに虫が挟まった時にまでさかのぼる。乗算器と加算器のテスト中、異常に気づいた技術者が、この部分に蛾が挟まっているのを見つけたのだ。原因となった蛾は、「バグ(虫)が実際に見つかった最初のケース」との説明文とともに、業務日誌にテープで貼り付けられた。
- 出典：WIRED「史上最悪のソフトウェアバグ」ワースト10を紹介(上) https://www.wired.com/2005/11/historys-worst-software-bugs/

モチベーション

- バグを極力減らしたい
  →人命や金銭的な損失を防ぐ

- 数学、証明の力でソフトウェアに正当性を与えたい

導入

式

式	意味
skip	なにもしない
X := a	変数Xにaを破壊的代入
E1; E2	式E1のあとに式E2を実行
while C do E end	式Cの計算結果が真のあいだ式Eを実行（ループ）
if C then E1 else E2	もしCが真ならE1を、そうでないならE2を実行
A + B、A - B、A × B	加算、減算、乗算
A = B、A != B	AとBは等しい、AとBは等しくない
A < B、A > B、A <= B、A >= B	AとBの大小比較
A /\ B、A \/ B	AかつB、AまたはB
A ⇒ B	AならばB
¬A	Aでない
Q [X → a]	式Q中の変数Xをaに置換

破壊的代入

- 破壊的代入とは、同じ変数に再代入可能すること。PythonやCなど、多くのプログラミング言語は破壊的代入可能。
- ホーア論理では破壊的代入を許すようなプログラミングモデルを対象。
- 数学の式は破壊的代入不可で、多くの関数型言語は破壊的代入に制限がある
- Pythonの例：
  x = 10
  x = x + 1 #同じ変数に代入している

ホーアの3つ組

- { P } c { Q } で表される
- Pは事前条件（precondition）
- Qは事後条件（postcondition）
- cはコマンド（command）
  手続き的なコードと考えてもらって良い

ホーアの3つ組の例1

- { X = 0 } X := 1 { X = 1}
- =が等価記号で、:=が代入だとすると
- これは、非形式的には
  『X = 0が成り立つときに、Xに1を代入すると、X = 1が成り立つ』
  と読める

ホーアの3つ組の例2

- ∀m { X = m } X := m + 1 { X = m + 1 }
- =が等価記号で、:=が代入だとすると
- これは、非形式的には
  『X = mが成り立つときに、Xにm+1を代入すると、X = m+1が成り立つ』
  と読める

ホーアの3つ組の例3

- 正しい3つ組
  - { X = 2 } X := X + 1 { X = 3 }
  - { X = 0 /\ Y = 0 }
    while X < 10 do Y := Y + 2; X := X + 1 end
    { Y = 20 }
- 誤った3つ組
  - { X = 2 } X := 5 { X = 0 }
  - { X = 0 }
    while X != 0 do X := X + 1 end
    { X = 1 }

置換

- 式中の変数を置き換える操作
- E [X → Y]としたとき、E中の変数XをYに置き換える
- 例：
  X = Y + 1 [Y → Z]
  は、
  X = Z + 1
  となる

ホーア論理

代入文の規則

- 規則：{ Q [X → a] } X := a { Q } (or { Q [a/X] } X := a { Q }との書き方もあり、代入文の公理とも呼ばれる)

- 例：
  { **X > 30 [X** → **X + 20]** } X := X + 20 { X > 30}

  { **X > 10** } X := X + 20 { X > 30}

代入文の規則の解説

- { ??? } X := X + Y { X = 1 }を考える。
- Q：事後条件がX = 1のときに、事前条件???はどうなるか？
- A：XにX + Yを代入するとX = 1が成り立つので、X = 1のXにX + Yを代入したX + Y = 1が事前条件としてなり立つはず。
- つまり、{ Q [X → a] } X := a { Q }という定理が得られる。

問題

- 以下のホーアの3つ組中の?を求めよ
- { ? } X := 2 × X { X <= 10 }
- { ? } X := Z + X { 0 <= X /\ X <= 5 }

事前と事後条件に関する規則

- 規則（事前条件）：
$$\frac{\{\ P'\ \}\ c\ \{\ Q\ \} \qquad P \Rightarrow P'}{\{\ P\ \}\ c\ \{\ Q\ \}}$$

- 規則（事後条件）：
$$\frac{\{\ P\ \}\ c\ \{\ Q'\ \} \qquad Q' \Rightarrow Q}{\{\ P\ \}\ c\ \{\ Q\ \}}$$

事前条件に関する規則の例

- 例：

$$\frac{\{ X < 5 \}\ X := X + 10\ \{ X < 15 \} \qquad X < 3 \Rightarrow X < 5}{\{ X < 3 \}\ X := X + 10\ \{ X < 15 \}}$$

直感的には、より厳しい事前条件にしても成り立つということ

事後条件に関する規則の例

- 例：

$$\frac{\{ X < 5 \}\ X := X + 10\ \{ X < 15 \} \qquad X < 15 \Rightarrow X < 20}{\{ X < 5 \}\ X := X + 10\ \{ X < 20 \}}$$

帰結の規則

- 規則：

$$\frac{\{\ P'\ \}\ c\ \{\ Q'\ \} \qquad P \Rightarrow P' \qquad Q' \Rightarrow Q}{\{\ P\ \}\ c\ \{\ Q\ \}}$$

空文の規則

- 規則：{ P } skip { P }

複合文の規則

- 規則：

$$\frac{\{ P \}\ c1\ \{ Q \} \qquad \{ Q \}\ c2\ \{ R \}}{\{ P \}\ c1;\ c2\ \{ R \}}$$

複合文の規則の例

- 例：{ a = n } X := a; skip { X = n }

- これはつまり、
  { a = n } X := a { X = n }
  { X = n } skip { X = n }
  の複合文

- 証明図：

$$\dfrac{\{\ a = n\ \}\ X := a\ \{\ X = n\ \} \qquad \{\ X = n\ \}\ \mathrm{skip}\ \{\ X = n\ \}}{\{\ a = n\ \}\ X := a;\ \mathrm{skip}\ \{\ X = n\ \}}\ \text{複合文の規則}$$

例題：スワップ

- 以下のスワップに関するホーアの3つ組が正しいことを証明せよ
- XとYの値を入れ替えたとき、大小関係が反転することの証明
- $\{ X < Y \}\ Z := X;\ X := Y;\ Y := Z\ \{ Y < X \}$

例題：スワップの証明図

代入文の規則に則っている

代入文の規則

$$\dfrac{\dfrac{\{ X < Y \}\ Z := X\ \{ Z < Y \} \qquad \{ Z < Y \}\ X := Y\ \{ Z < X \}}{\{ X < Y \}\ Z := X\ ;\ X := Y\ \{ Z < X \}}\ \text{複合文の規則} \qquad \{ Z < X \}\ Y := Z\ \{ Y < X \}}{\{ X < Y \}\ Z := X;\ X := Y;\ Y := Z\ \{ Y < X \}}\ \text{複合文の規則}$$

条件文の規則

- 規則：

$$\frac{\{ P \wedge b \}\ c1\ \{ Q \} \qquad \{ P \wedge \neg b \}\ c2\ \{ Q \}}{\{ P \}\ \mathrm{if}\ b\ \mathrm{then}\ c1\ \mathrm{then}\ c2\ \{ Q \}}$$

条件文の規則の例

$$
\dfrac{
\dfrac{
\begin{array}{c}
\{\ X + 1 > 0 \ /\backslash\ b \ \}\ X := X + 1 \ \{\ X > 0 \ /\backslash\ b \ \} \\
\{\ X > 2 \ /\backslash\ b \ \} \Rightarrow \{\ X + 1 > 0 \ /\backslash\ b \ \} \\
\{\ X > 0 \ /\backslash\ b \ \} \Rightarrow \{\ X > 0 \ \}
\end{array}
}{
\{\ X > 2 \ /\backslash\ b \ \}\ X := X + 1 \ \{\ X > 0 \ \}
}
\qquad
\dfrac{
\begin{array}{c}
\{\ X + 2 > 0 \ /\backslash\ \neg b \ \}\ X := X + 2 \ \{\ X > 0 \ /\backslash\ \neg b \ \} \\
\{\ X > 2 \ /\backslash\ \neg b \ \} \Rightarrow \{\ X + 2 > 0 \ /\backslash\ \neg b \ \} \\
\{\ X > 0 \ /\backslash\ \neg b \ \} \Rightarrow \{\ X > 0 \ \}
\end{array}
}{
\{\ X > 2 \ /\backslash\ \neg b \ \}\ X := X + 2 \ \{\ X > 0 \ \}
}
}{
\{\ X > 2 \ \}\ \text{if } b \text{ then } X := X + 1 \text{ else } X := X + 2 \ \{\ X > 0 \ \}
}
$$

ループ文の規則

- 規則：

$$\frac{\{ P \wedge b \}\ c\ \{ P \}}{\{ P \}\ \mathrm{while}\ b\ \mathrm{do}\ c\ \mathrm{end}\ \{ P \wedge \neg b \}}$$

- Pはループ中に不変の条件であるため、ループ不変条件と呼ばれる

ループ文の規則の例

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \begin{array}{c}
        \{ X + 1 <= 3 \}\ X := X + 1\ \{ X <= 3 \} \\
        \{ X <= 2 \} \Rightarrow \{ X + 1 <= 3 \}
        \end{array}
      }{
        \{ X <= 2 \}\ X := X + 1\ \{ X <= 3 \}
      }
      \qquad
      \{ X <= 3 \land X <= 2 \} \Rightarrow \{ X <= 2 \}
    }{
      \{ X <= 3 \land X <= 2 \}\ X := X + 1\ \{ X <= 3 \}
    }
  }{
    \{ X <= 3 \}\ \mathrm{while}\ (X <= 2)\ \mathrm{do}\ X := X + 1\ \mathrm{end}\ \{ X <= 3 \land \neg(X <= 2) \}
  }
  \qquad E
}{
  \{ X <= 3 \}\ \mathrm{while}\ (X <= 2)\ \mathrm{do}\ X := X + 1\ \mathrm{end}\ \{ X = 3 \}
}
$$

ただし、$E = \{ X <= 3 \land \neg(X <= 2) \} \Rightarrow \{ X = 3 \}$

ホーア論理による
プログラムの検証

装飾表記

- プログラムの行毎に事前・事後条件を記述する方法
- ホーア論理による証明と対応している

事前・事後条件に関する表記

- { P }から{ Q }が導出可能な場合、{ P } -> { Q }と記述し、{ P }を{ Q }に変換する事を意味する
- つまり、帰結の規則の適用
- 例：{ True } -> { m = m }

装飾表記の例1：0までのデクリメント

```
  { True }
while ¬(X = 0) do
  X := X - 1
end
  { X = 0 }
```

証明

```
  { True }
while ¬(X = 0) do
    { True /\ ¬(X = 0) } -> { True }
  X := X - 1
    { True }
end
  { True /\ X = 0 } -> { X = 0 }
```

装飾表記の例2

```
   { True }
X := m;
Z := p;
while ¬(X = 0) do
   Z := Z - 1;
   X := X - 1
end
   { Z = p - m }
```

証明

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

例2の解剖（1/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

$$\frac{\{ P \wedge b \}\ c\ \{ P \}}{\{ P \}\ \text{while } b \text{ do } c \text{ end}\ \{ P \wedge \neg b \}}$$

P : Z - X = p - m

b : ¬(X = 0)

c : Z := Z - 1; X := X - 1

¬b : X = 0

例2の解剖（2/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

{ Q [X → a] } X := a { Q }

Q [X → a] : Z - (X - 1) = p - m

Q : Z - X = p - m

例2の解剖（3/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

{ Q [X → a] } X := a { Q }

Q [X → a] : (Z - 1) - (X - 1) = p - m

Q : Z - (X - 1) = p - m

例2の解剖（4/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
Z := Z - 1;
      { Z - (X - 1) = p - m }
X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

$$\frac{\{\ P\ \}\ c1\ \{\ Q\ \} \qquad \{\ Q\ \}\ c2\ \{\ R\ \}}{\{\ P\ \}\ c1;\ c2\ \{\ R\ \}}$$

P : (Z - 1) - (X - 1) = p - m

c1 : Z := Z - 1

Q : Z - (X - 1) = p - m

c2 : X := X - 1

R : Z - X = p - m

例2の解剖（5/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

{ Q [X → a] } X := a { Q }

Q [X → a] : X = m /\ p = p

Q : X = m /\ Z = p

例2の解剖（6/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

{ Q [X → a] } X := a { Q }

Q [X → a] : m = m

Q : X = m

例2の解剖（7/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

$$\frac{\{\ P\ \}\ c1\ \{\ Q\ \}\qquad \{\ Q\ \}\ c2\ \{\ R\ \}}{\{\ P\ \}\ c1;\ c2\ \{\ R\ \}}$$

P : X = m /\ p = p

c1 : Z := p

Q : Z - X = p - m

c2 : while ¬(X = 0) do 省略 end

R : Z - X = p - m /\ X = 0

例2の解剖（8/8）

```
   { True } -> { m = m }
X := m;
   { X = m } -> { X = m /\ p = p }
Z := p;
   { X = m /\ Z = p } -> { Z - X = p - m }
while ¬(X = 0) do
      { Z - X = p - m /\ ¬(X = 0) } ->
      { (Z - 1) - (X - 1) = p - m }
   Z := Z - 1;
      { Z - (X - 1) = p - m }
   X := X - 1
      { Z - X = p - m }
end
   { Z - X = p - m /\ X = 0 } -> { Z = p - m }
```

$$\frac{\{\ P\ \}\ c1\ \{\ Q\ \} \qquad \{\ Q\ \}\ c2\ \{\ R\ \}}{\{\ P\ \}\ c1;\ c2\ \{\ R\ \}}$$

P : m = m

c1 : X := m

Q : X = m /\ p = p

c2 : Z :=p ; while ¬(X = 0) do 省略 end

R : Z - X = p - m /\ X = 0

発展

ホーア論理の実際

- ホーア論理のみでは難しい
- ホーア論理を発展させ、分離論理（Separation Logic）に応用した研究成果もある
  - ヒープメモリの検証が可能
  - RustBelt、Viperなどで利用
  - RustBeltでは実際にRustのバグを発見している
- 事前条件、事後条件を抜き出してた契約プログラミングへも発展

契約プログラミング

- 英語だとProgramming by Contract
- ソースコード中に満たすべき述語を記述し、ソフトウェアの正当性を保証する手法
- ホーア論理が契約プログラミングの概念確立に大きな影響を与えた。今後10年で現実的に産業界に実用される可能性がある（C++ 0x23でも導入予定）

RustによるPrustiの例

```
#[pure] // 副作用無し
#[requires(!self.is_empty() ==> result > 0)] // 事前条件
#[ensures(result >= 0)] // 事後条件
fn len(&self) -> usize {    // こっちの関数で検証する
    match self {
        Link::Empty => 0,
        Link::More(box node) => 1 + node.next.len(),
    }
}
```

レポート

代入

- 以下の?を求めよ
  - { ? } X := 10 + X { Y = X }
  - { ? } X := X + 1; Y := Y + X { Y = X + Z }

m÷nの商と剰余

以下のプログラムはm÷nの商をYに、剰余をXに求めるプログラムである
この事前・事後条件が正しいことを証明せよ

```
  { True }
X := m;
Y := 0;
while n <= X do
  X := X - n;
  Y := Y + 1
end
  { n * Y + X = m /\ X < n }
```

二乗

以下のプログラムはmの2乗をZに求めるプログラムである
この事前・事後条件が正しいことを証明せよ

```
  { X = m }
Y := 0;
Z := 0;
while ¬(Y = X) do
  Z := Z + X;
  Y := Y + 1
end
  { Z = m × m }
```

問題

- 先の問題を証明し、レポートとして提出せよ
- 締め切り：8月16日 23時50分 (JST)