形式的体系の定理証明支援系上での実現法

2006年10月31日

木下佳樹　高橋孝一　田辺良則　湯浅能史

産業技術総合研究所

システム検証研究センター

自動検証と対話型検証

- 自動検証
  - 検証すべき問題を，有限個の空間の探索問題に落とす.
  - 代表例: モデル検査
  - 適用できる範囲に限りはあるが，「ボタン一つで」答が出る.
- 対話型検証
  - 検証すべき問題を数学的に定式化.
  - 定理証明器による支援
  - 高い専門的知識が要求されるが，適用できる範囲は広い.
- 2つの混合

定理証明支援系によるプログラム検証

- 「プログラムの正しさ」を (数学の) 定理として表現し，定理を数学的に証明する.
  - このチュートリアルでは，「ホア論理」(Hoare logic) と呼ばれる体系を用いる.
- 証明は，計算機上で実行する.
  - 人間の誤りが混入し得ない.
- 証明を支援するシステム
  - 証明の入力を支援する.
  - 入力した証明の正しさを確認する.
  - このチュートリアルでは，Agdaというシステムを使用する.

本チュートリアルでは

- 以下を，例によって紹介する.
- ホア論理.
  - ホア論理とは何か.
  - どのようなことが示せるか.
- 定理証明支援系Agda.
  - Agda上での証明，とはどのようなものか.
  - 検証対象をAgda上で表現する手順はどのようになるか.

はじめに
ホア論理
定理証明支援系Agda
ホア論理のAgdaでの符号化
ホア論理の健全性
まとめ

はじめに
ホア論理
定理証明支援系Agda
ホア論理のAgdaでの符号化
ホア論理の健全性
まとめ

対象プログラム例

- 正の整数a,bを入力として,
  aをbで割った商qと余りrを出力するプログラム

- 仮定: a,b: 整数;　a,b > 0
- 結論: a = b*q + r; 0<= r < b

```
q := 0;
r := a;
while r >= b do
  r := r - b;
  q := q + 1
od
// returns (q,r)
```

ホア論理による証明例

$$\dfrac{\dfrac{\{a = bq + r \wedge 0 \le r \wedge b \le r \rightarrow a = b(q+1) + (r-b) \wedge 0 \le r - b\} \quad \{a = b(q+1) + (r-b) \wedge 0 \le r - b\}\ \texttt{r:=r-b}\ \{a = b(q+1) + r \wedge 0 \le r\}}{\{a = bq + r \wedge 0 \le r \wedge b \le r\}\ \texttt{r:=r-b}\ \{a = b(q+1) + (r-b) \wedge 0 \le r - b\}}}{\dfrac{\dfrac{\{a = b(q+1) + r \wedge 0 \le r\}\ \texttt{q:=q+1}\ \{a = bq + r \wedge 0 \le r\}}{\{a = bq + r \wedge 0 \le r \wedge b \le r\}\ \texttt{r:=r-b;\ q:=q+1}\ \{a = bq + r \wedge 0 \le r\}}}{\{a = bq + r \wedge 0 \le r\}\ \texttt{while (r>=b) do \{r:=r-b; q:=q+1;\} od}\ \{a = bq + r \wedge 0 \le r \wedge b < r\}}}$$

$$\dfrac{\dfrac{\{a > 0 \wedge b > 0 \rightarrow a = b \cdot 0 + a \wedge 0 \le a\} \quad \{a = b \cdot 0 + a \wedge 0 \le a\}\ \texttt{q:=0}\ \{a = bq + a \wedge 0 \le a\}}{\{a > 0 \wedge b > 0\}\ \texttt{q:=0}\ \{a = bq + a \wedge 0 \le a\}} \quad \{a = bq + a \wedge 0 \le a\}\ \texttt{r:=a}\ \{a = bq + r \wedge 0 \le r\}}{\{a > 0 \wedge b > 0\}\ \texttt{q:=0; r:=a; while (r>=b) do \{r:=r-b; q:=q+1;\} od}\ \{a = bq + r \wedge 0 \le r \wedge r < b\}}$$

対象: whileプログラム

- b∈B:「条件式」
- p∈P:「基本命令」
- c∈C: whileプログラム

```
c ::=  p
     |  skip
     |  abort
     |  c ; c
     |  if  b  then  c  else  c fi
     |  while b do c od
```

条件式と基本命令

- 条件式の例:
  - a = 5
  - a != b
  - a <= b+7
  - a + b > c
- 基本命令の例
  - a := b
  - a := b - 3
  - b := (a + b) - 5

プログラム変数を使った算術式の(大小)比較

プログラム変数への算術式の代入

ホア三つ組

- ホア三つ組　{ b1 }　c　{ b2 }
  - b1, b2∈B ：条件式
  - c ∈ C : whileプログラム
- 直観的な意味: b1 が成立しているときに c を実行して，cが停止すれば b2 が成立する.
- 例1:　{ x = 5 }　x := x + 1　{ x > 0 }　正しい
- 例2:
  { x = 2 } if x = 1 then y := 3 else z := 3 fi { x < z }　正しい
- 例3:
  { a > 0 ∧ b > 0 } q:=0; r:=a; while r>=b do r:=r-b; q:=q+1 od { a = bq+r ∧ 0 <= r < b }　正しい?

ホア論理: 公理

- {b} skip {b}　　(b∈B)
- {b} abort {false}　　(b∈B)
- 基本命令については,

  {b} p {b'}　　(b,b'∈B, p∈P)

  の形の公理 の集合が与えられているものとする.

基本命令に関する公理の例

- 基本命令が算術式の代入の場合
  (変数v) := (算術式e)
- 条件式 b に対し，次を公理とする.
  $\{ b [v \leftarrow e] \}$ v := e $\{ b \}$

- 例:
  $\{ \underline{y+1} + y = 5 \}$ x := y+1 $\{ \underline{x} + y = 5 \}$

ホア論理: 推論規則

$$\frac{\{b_1 \to b_1'\} \quad \{b_1'\}\, c\, \{b_2'\} \quad \{b_2' \to b_2\}}{\{b_1\}\, c\, \{b_2\}}$$

「恒真な」条件式

$$\frac{\{b_1\}\, c_1\, \{b_2\} \quad \{b_2\}\, c_2\, \{b_3\}}{\{b_1\}\, c_1; c_2\, \{b_3\}}$$

$$\frac{\{b_1 \wedge b\}\, c_1\, \{b_2\} \quad \{b_1 \wedge \neg b\}\, c_2\, \{b_2\}}{\{b_1\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{fi}\ \{b_2\}}$$

$$\frac{\{b_1 \wedge b\}\, c\, \{b_1\}}{\{b_1\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \{b_1 \wedge \neg b\}}$$

推論規則の妥当性

$$\frac{\{b_1 \wedge b\}\ c_1\ \{b_2\} \qquad \{b_1 \wedge \neg b\}\ c_2\ \{b_2\}}{\{b_1\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{fi}\ \{b_2\}}$$

- 仮定
  - b1かつb が成り立つとき,
    c1 を実行して停止すれば b2 が成り立つ.
  - b1かつ "bでない" とき,
    c2 を実行して停止すれば b2 が成り立つ.
- 結論
  - b1 が成り立つとき,
    if b then c1 else c2 fi
    を実行して停止すれば b2 が成り立つ.

推論規則の妥当性

$$\frac{\{b_1 \wedge b\}\, c\, \{b_1\}}{\{b_1\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}\ \{b_1 \wedge \neg b\}}$$

- 仮定
  - b1かつb が成り立つとき,
    c を実行して停止すれば b1 が成り立つ.
    (b1 は 不変式 と呼ばれる)
- 結論
  - b1 が成り立つとき,
    while b do c od
    を実行して停止すれば b1かつ "bでない" が成り立つ.

健全性定理 (概略)

ホア3つ組 {b} c {b'} について,

- {b} c {b'} が証明可能であり,
- c の実行前に b が成立しており,
- c の実行が停止するとする.

このときcの実行後に b' が成立する.

- 「成立する」とは?
- 「実行」とは?

ホア論理による証明例

$$\dfrac{\dfrac{\{a = bq + r \wedge 0 \le r \wedge b \le r \rightarrow a = b(q+1) + (r-b) \wedge 0 \le r - b\} \quad \{a = b(q+1) + (r-b) \wedge 0 \le r - b\}\ \texttt{r:=r-b}\ \{a = b(q+1) + r \wedge 0 \le r\}}{\{a = bq + r \wedge 0 \le r \wedge b \le r\}\ \texttt{r:=r-b}\ \{a = b(q+1) + (r-b) \wedge 0 \le r - b\}}}{\dfrac{\dfrac{\{a = b(q+1) + r \wedge 0 \le r\}\ \texttt{q:=q+1}\ \{a = bq + r \wedge 0 \le r\}}{\{a = bq + r \wedge 0 \le r \wedge b \le r\}\ \texttt{r:=r-b;\ q:=q+1}\ \{a = bq + r \wedge 0 \le r\}}}{\{a = bq + r \wedge 0 \le r\}\ \texttt{while (r>=b) do \{r:=r-b; q:=q+1;\} od}\ \{a = bq + r \wedge 0 \le r \wedge b < r\}}}$$

$$\dfrac{\dfrac{\dfrac{\{a > 0 \wedge b > 0 \rightarrow a = b \cdot 0 + a \wedge 0 \le a\} \quad \{a = b \cdot 0 + a \wedge 0 \le a\}\ \texttt{q:=0}\ \{a = bq + a \wedge 0 \le a\}}{\{a > 0 \wedge b > 0\}\ \texttt{q:=0}\ \{a = bq + a \wedge 0 \le a\}} \quad \{a = bq + a \wedge 0 \le a\}\ \texttt{r:=a}\ \{a = bq + r \wedge 0 \le r\}}{\{a > 0 \wedge b > 0\}\ \texttt{q:=0; r:=a; while (r>=b) do \{r:=r-b; q:=q+1;\} od}\ \{a = bq + r \wedge 0 \le r \wedge r < b\}}$$

ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
 { a = bq + r ∧ 0 <= r ∧ b <= r }
 { a = b(q+1) + (r-b) ∧ 0 <= r-b }
 r := r - b;
 { a = b(q+1) + r ∧ 0 <= r }
 q := q + 1
 { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
 { a = bq + r ∧ 0 <= r ∧ b <= r }
 { a = b(q+1) + (r-b) ∧ 0 <= r-b }
 r := r - b;
 { a = b(q+1) + r ∧ 0 <= r }
 q := q + 1
 { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

{ b [v ← e] } v:=e {b}

ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
 { a = bq + r ∧ 0 <= r ∧ b <= r }
 { a = b(q+1) + (r-b) ∧ 0 <= r-b }
 r := r - b;
 { a = b(q+1) + r ∧ 0 <= r }
 q := q + 1
 { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

{ b [v ← e] } v:=e {b}

ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
 { a = bq + r ∧ 0 <= r ∧ b <= r }
 { a = b(q+1) + (r-b) ∧ 0 <= r-b }
 r := r - b;
 { a = b(q+1) + r ∧ 0 <= r }
 q := q + 1
 { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

$$\frac{\{b1\}c1\{b2\} \quad \{b2\}c2\{b3\}}{\{b1\}\ c1;c2\ \{b3\}}$$

ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
  { a = bq + r ∧ 0 <= r ∧ b <= r }
  { a = b(q+1) + (r-b) ∧ 0 <= r-b }
  r := r - b;
  { a = b(q+1) + r ∧ 0 <= r }
  q := q + 1
  { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

$$\frac{\{b1 \rightarrow b1'\} \quad \{b1'\}c\{b2\}}{\{b1\}\ c\ \{b2\}}$$

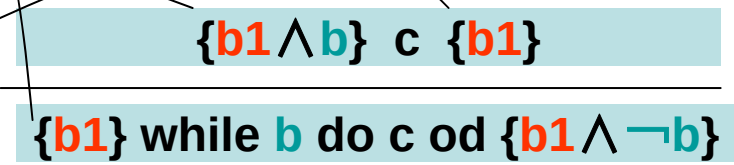
ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
 { a = bq + r ∧ 0 <= r ∧ b <= r }
 { a = b(q+1) + (r-b) ∧ 0 <= r-b }
 r := r - b;
 { a = b(q+1) + r ∧ 0 <= r }
 q := q + 1
 { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

$$\frac{\{b1 \wedge b\}\ c\ \{b1\}}{\{b1\}\ \text{while } b \text{ do } c \text{ od } \{b1 \wedge \neg b\}}$$

ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
 { a = bq + r ∧ 0 <= r ∧ b <= r }
 { a = b(q+1) + (r-b) ∧ 0 <= r-b }
 r := r - b;
 { a = b(q+1) + r ∧ 0 <= r }
 q := q + 1
 { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```



ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
  { a = bq + r ∧ 0 <= r ∧ b <= r }
  { a = b(q+1) + (r-b) ∧ 0 <= r-b }
  r := r - b;
  { a = b(q+1) + r ∧ 0 <= r }
  q := q + 1
  { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

$$\frac{\{b1 \wedge b\}\ c\ \{b1\}}{\{b1\}\ \text{while } b \text{ do } c \text{ od } \{b1 \wedge \neg b\}}$$

ホア論理による証明例

```
{ a > 0 ∧ b > 0 }
{ a = b*0 + a ∧ 0 <= a}
q := 0;
{ a = bq + a ∧ 0 <= a}
r := a;
{ a = bq + r ∧ 0 <= r}
while r >= b do
  { a = bq + r ∧ 0 <= r ∧ b <= r }
  { a = b(q+1) + (r-b) ∧ 0 <= r-b }
 r := r - b;
  { a = b(q+1) + r ∧ 0 <= r }
 q := q + 1
  { a = bq + r ∧ 0 <= r }
od
{ a = bq + r ∧ 0 <= r ∧ r < b }
```

非形式的

whileプログラム c

whileプログラム (定義)

検証したい性質 b

ホア論理による
cでbが成り立つことの証明 P

ホア論理 (定義)

日本語

符号化

形式的

whileプログラム c

whileプログラム (定義)

検証したい性質 b

cでbが成り立つことの証明 P

ホア論理 (定義)

Agda

はじめに
ホア論理
定理証明支援系Agda
ホア論理のAgdaでの符号化
ホア論理の健全性
おわりに

Agda

- 対話型証明支援ソフトウェア
- Martin-Löf型理論に基づく.
- プラグイン機構による他ツールとの連携
- 主にChalmers工科大学(スウェーデン)で開発.
  当センターも開発に協力.
- http://agda.sourceforge.net/　(整備中)

Agdaによる証明

- 命題を「その命題がなりたつことの証拠の集合」だと思う.

命題が成り立つ	集合が空でない.
A	A
AならばB (A→B)	関数 A→B
AかつB (A ∧ B)	直積 A × B
A またはB (A∨B)	直和 A ⊎ B

Agdaによる証明の例

命題1: $(A \to B \land B \to C) \to (A \to C)$

証明の考え方

$(A \to B \land B \to C) \to (A \to C)$

$\Downarrow$ $\Downarrow$

与えられた p に対して q を作る.

集合が空でない
= 関数が定義できる

$p \in A \to B \land B \to C$

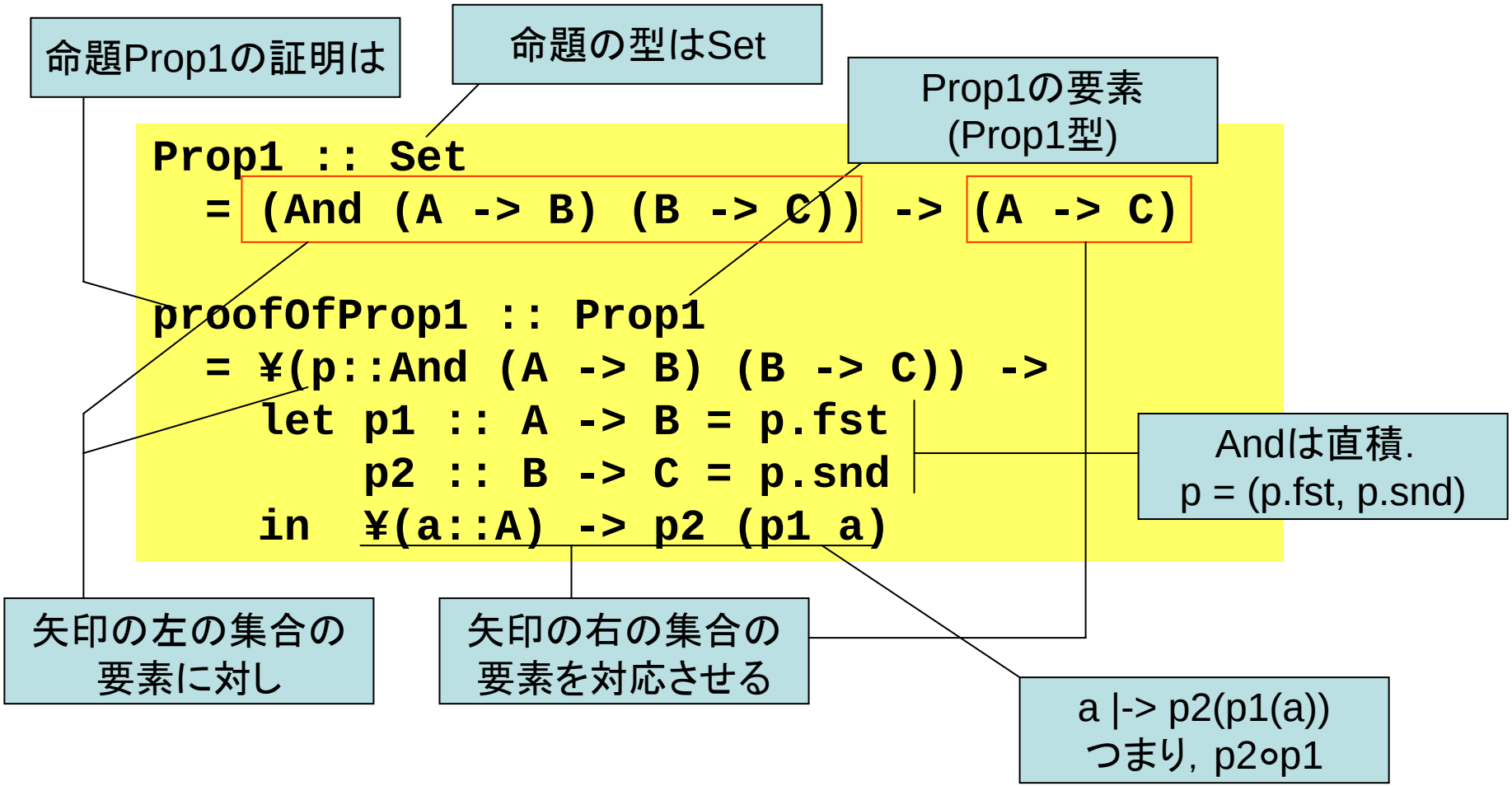
直積

$\Rightarrow$ $p = (p1, p2)$, $p1 \in A \to B$, $p2 \in B \to C$

$\Downarrow$

$q \in A \to C \Rightarrow q = p2 \circ p1$

Agdaによる証明の例



はじめに
ホア論理
定理証明支援系Agda
ホア論理のAgdaでの符号化
ホア論理の健全性
おわりに

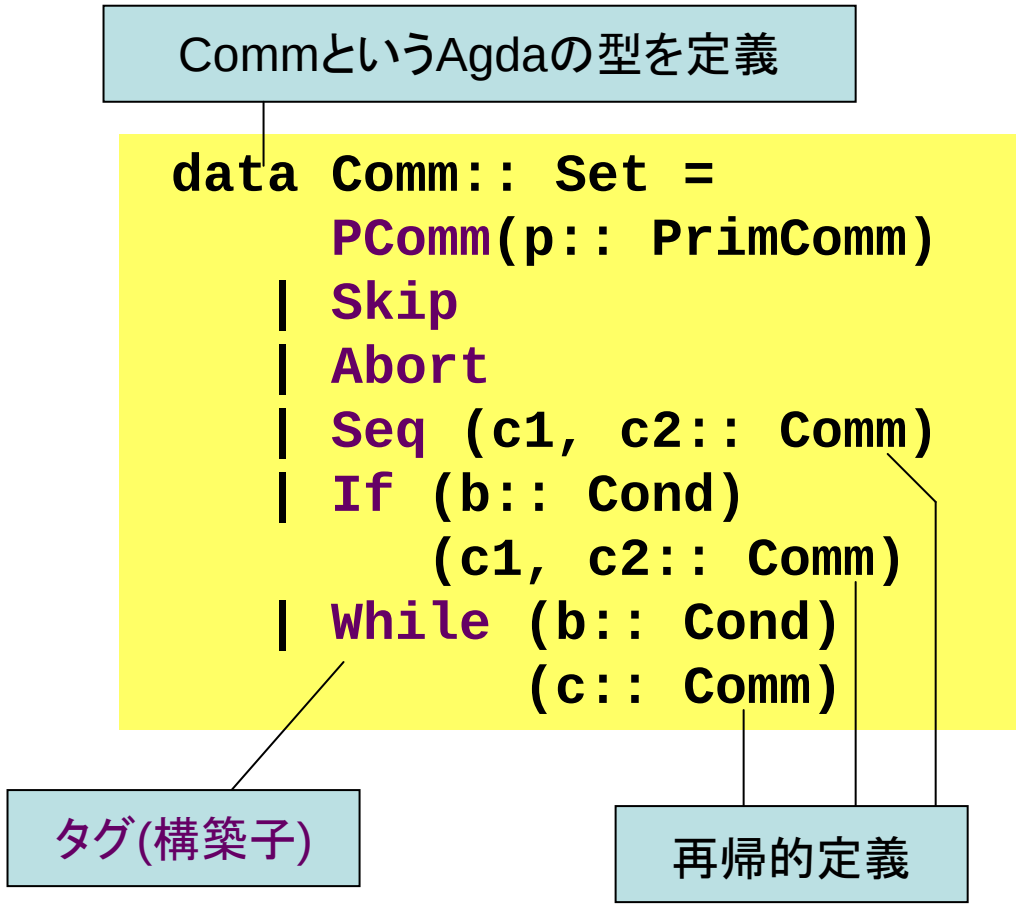
符号化

- 符号化：非形式的な世界の登場人物たちを，形式的体系の中で表現すること
- ここでは，Agdaに符号化する．符号化する対象は:
  - whileプログラム
  - ホア論理
    - ホア3つ組 / 公理 / 推論規則 / 証明図
- 対象に対応するAgdaの型 (集合) を定義していく．
- (非形式的な)ホア論理による(具体的な)証明が符号化できれば (定義した型を持てば)，証明に誤りがないことが保証されたことになる．

whileプログラムの符号化

whileプログラム

c ::= p
| skip
| abort
| c ; c
| if b then c
else c fi
| while b do c od



Cond : 条件式の集合

PrimComm : 基本命令の集合

符号化されたプログラムの例

次のプログラム Cmd1 の符号化:

if x = 1 then y := 3 else z := 3 fi

条件 x = 1, 基本命令 y:=3, z:=3 が, それぞれ

```
Cond1  :: Cond                -- x = 1
Cmd_y3 :: PrimComm            -- y := 3
Cmd_z3 :: PrimComm            -- z := 3
```

と符号化済みだとする. Cmd1を符号化すると次のようになる.

```
Cmd1 :: Comm
    = If Cond1 (PComm Cmd_y3) (PComm Cmd_z3)
```

ホア論理の符号化

- ホア3つ組　{b1} c {b2}　の符号化

```
data HT :: Set
    = ht (b1:: Cond) (c:: Comm) (b2:: Cond)
```

ホア論理の符号化

- 証明図を符号化する. (公理と推論規則は証明図の定義の中に含まれる.)
- 証明図は, 次のいずれかである.
  - (PrimProof, c) は公理cの証明図である.
  - ...(略)...
  - pr1 が {b1}c1{b2} の証明図で,
    pr2 が {b2}c2{b3} の証明図ならば,
    (SeqProof, pr1, pr2) は, {b1}c1;c2{b3}の証明図である.
  - ...(略)...
  - pr1が {b1∧b} c {b1} の証明図ならば,
    (WhileProof, pr1)は,
    {b1} while b do c od {b1∧¬b} の証明図である.

ホア論理の符号化

- data による再帰的定義では，うまく表現できない．

```
data HTproof :: Set =

    PrimProof (ax :: Axiom)
  | ...
  | SeqProof (ht::HT)(pr1,pr2::HTproof)
  | ...
  | WhileProof (ht::HT)(pr)
```

付帯条件が必要

ホア論理の符号化

idata: タグごとにパラメタを変えられる帰納的定義.

HTproof b1 c b2
: ホア三つ組 {b1} c {b2} の証明の型

```
idata HTproof :: Cond -> Comm -> Cond -> Set where
  PrimProof (bPre::Cond)(pcm::PrimComm)(bPost::Cond)
        (pr::Axiom bPre pcm bPost)
        :: HTproof bPre (PComm pcm) bPost
  ... (略) ...

  SeqProof(bPre::Cond)(cm1::Comm)(bMid::Cond)
        (cm2::Comm)(bPost::Cond)
        (pr1:: HTproof bPre cm1 bMid)
        (pr2:: HTproof bMid cm2 bPost)
        :: HTproof bPre (Seq cm1 cm2) bPost
  ... (略) ...
```

**** の条件を満たすとき PrimProof bPre pcm bPost prは,
{bPre} pcm {bPost} の証明である.

**** の条件を満たすとき,
SeqProof bPre cm1 bMid... は,

{bPre} cm1;cm2 {bPost} の証明である.

符号化された証明の例

{ x = 2 } if x = 1 then y := 3 else z := 3 fi { x < z }

(コード参照)

cによる遷移系 T

健全性定理の証明 (Pがあれば Tでbが成り立つという保証)

健全性定理

whileプログラム c

whileプログラム (定義)

ホア論理による cでbが成り立つことの証明 P

ホア論理 (定義)

検証したい性質 b

非形式的

日本語

符号化

cによる遷移系 T

健全性定理の証明

whileプログラム c

whileプログラム (定義)

cでbが成り立つことの証明 P

ホア論理 (定義)

検証したい性質 b

形式的

Agda

はじめに
ホア論理
定理証明支援系Agda
ホア論理のAgdaでの符号化
ホア論理の健全性
おわりに

遷移系

- 遷移系 T = (S, R) :
  - S : 状態の集合
  - R : 集合 S 上の関係　　$R \subseteq S \times S$
- 状態の集合 S を固定する.
- 各基本命令 $p \in P$ に対して，遷移系 Tp = (S,Rp) が与えられている.
  - Tp : pの「ふるまい」の指定
- 各条件式 $b \in B$ に対して，$f(b) \subseteq S$ が与えられている.
  - f(b): bが「成り立つ」状態の集合

whileプログラムの意味

- whileコマンド $c \in C$ に，遷移系 $T_c = (S, R_c)$を対応させる.
  - c = skipのとき，$R_c = \{ (s,s) \mid s \in S \}$
  - c = abortのとき，$R_c = \{\}$
  - c = c1;c2のとき，$R_c = R_{c1} \circ R_{c2}$
  - c = if b then c1 else c2 fi のとき，
    $R_c = R_{c1} \circ \Delta_{f(b)} \cup R_{c2} \circ \Delta_{S-f(b)}$
    ここに，$A \subseteq S$ に対して，$\Delta_A = \{ (s,s) \mid s \in A \}$
  - c = while b do c1 od のとき，
    $R_c = \cup_{n \in N} R'_n$
    ここに $R'_0 = \Delta_{S-f(b)}$
    $R'_{n+1} = R'_n \circ R_{c1} \circ \Delta_{f(b)}$

ホア三つ組の意味

- 遷移系 Tc = (S,Rc) が ホア三つ組 {b1} c {b2} を満足するとは,

  $s1 \in f(b1), (s1,s2) \in R_c$ ならば $s2 \in f(b2)$

  となること.

  – 直観的に述べた
  「b1が成り立っているときに c を実行して停止すれば, b2 が成り立つ」
  の正確な意味

健全性定理

- 公理の集合「が与えられており,
- 「の要素は, {b1} p {b2}　(b1,b2∈B, p∈P)　の形であり,
- p∈P に対して与えられている遷移系 Tp は, {b1}p{b2}を満足しているとする.
- whileコマンドcに対応する遷移系を Tc として,
- ホア三つ組 {b1}c{b2}が「から証明可能ならば,

Tcは{b1}c{b2}を満足する.

遷移系の符号化

- 集合と関係
  - 集合A上の述語:
  - 集合A上の関係:
  - 関係 R1 と R2の合成
  - etc

```
Pred(A::Set) :: A -> Set
```

```
Rel(S::Set) :: S -> S -> Set
```

```
comp(S::Set)(R1,R2::Rel S) :: Rel S
  = ¥(s1::S) -> ¥(s2::S) ->
    Exist (¥(s::S) -> (And (R1 s1 s) (R2 s s2)))
```

- 自然数

```
data Nat = zero | succ (m :: Nat)
```

- 遷移系
  - 台集合

```
State :: Set
```

  - 遷移関係

```
TransRel(State::Set) :: Set = Rel State
```

健全性定理の符号化

- B, P, S, f, 「 などをパラメタとするパッケージ.

```
package Hoare
  (Cond :: Set)
  (PrimComm :: Set)
  (State :: Set)
  (SemCond :: Cond -> Pred State)
  (PrimSemComm :: PrimComm -> Rel State)
  (Axiom :: Cond -> PrimComm -> Cond -> Set)
  ... 略 ...
  where
  ......
```

条件式の集合B

基本命令の集合P

状態の集合S

条件式の意味 f

基本命令の意味 T_p

公理の集合 「

健全性定理の符号化

- プログラムcに対する遷移関係 R_c

```
SemComm(!c::Comm) :: Rel State
  = case c of
  (Skip       )-> deltaGlob
  (Abort      )-> emptyRel
  (PComm pc   )-> PrimSemComm pc
  (Seq c1 c2 )-> comp (SemComm c1) (SemComm c2)
  (If b c1 c2)->
    union (comp (delta (SemCond b)) (SemComm c1))
          (comp (delta (NotP (SemCond b))) (SemComm c2))
  (While b c1)->
    unionInf (¥(n::Nat)->
              comp (repeat n (comp (delta (SemCond b)) (SemComm c1)))
                   (delta (NotP (SemCond b))))
```

$$R_c = (R_{c1} \circ \Delta_{f(b)}) \cup (R_{c2} \circ \Delta_{S\text{-}f(b)})$$

健全性定理の符号化

- 「Tc が {b1}c{b2} を満足する」ことの定義

```
Satisfies (!b1::Cond)(!c::Comm)(!b2::Cond) :: Set
  = (s1,s2::State) ->
    SemCond b1 s1 -> SemComm c s1 s2 -> SemCond b2 s2
```

任意の s1,s2∈S について,
s1∈f(s1)かつ(s1,s2)∈Tc ならば
s2∈f(s2)である.

健全性定理の符号化

- 健全性定理の言明

```
Soundness (!b1::Cond)(!c::Comm)(!b2::Cond)
  :: HTproof b1 c b2 -> Satisfies b1 c b2
```

- 健全性定理の証明
  - (コード例を参照)

はじめに
ホア論理
定理証明支援系Agda
ホア論理のAgdaでの符号化
ホア論理の健全性
おわりに

まとめ

- ホア論理
  - プログラムの正当性を，定理証明の手法で検証するための枠組み
- Agda
  - Martin-Löf型理論に基づいた定理証明支援系
- Agdaへの符号化
  - 形式的体系や，数学的対象を「符号化」することで，Agda上の表現とすることができる.
  - 体系の性質をAgdaの定理として表現し，証明を与えることで，性質が成立することを検証することができる.

参考書

- ホア論理
  - 林 晋, プログラム検証論, 情報数学講座 8, 共立出版
- Agda
  - An Agda Tutorial (http://agda.sourceforge.net/tutorial/ 内のDocumentセクション) ... 文法が一部現在のものとは変わっている. 修正作業中.
  - Nordström, Petersson, and Smith: Programming in Martin-Löf's Type Theory (http://www.cs.chalmers.se/Cs/Research/Logic/book/)

- このスライドおよび関連するAgda1.0.0で記述したコードは，次のURLからダウンロードできます．
  `http://staff.aist.go.jp/tanabe.yoshinori/06/10/31/`
- Agda1.0.0は次のURLからダウンロードできます．
  `http://sourceforge.net/project/showfiles.php?group_id=123257`
  (2006年10月31日現在) Windows用インストーラは "File Releases" にある **`agda-1.0.0rc4-i386-windows.exe`** です．