第3回の補足

情報工学科　篠埜 功

この資料では第 3 回の配布資料についていくつか補足を行う。

1　17ページの Hoare triple の導出

17 ページの Hoare triple **while** true **do** x:=1 の導出木を以下に示す。

$$\dfrac{\dfrac{\dfrac{\text{true} \Rightarrow \text{true} \wedge \text{true} \quad \overline{\{\text{true}\}\ \text{x:=1}\ \{\text{true}\}}\ (\text{assign})}{\{\text{true} \wedge \text{true}\}\ \text{x:=1}\ \{\text{true}\}}\ (\text{conseq})}{\{\text{true}\}\ \textbf{while}\ \text{true}\ \textbf{do}\ \text{x:=1}\ \{\text{true} \wedge \neg\ \text{true}\}}\ (\text{while}) \quad \text{true} \wedge \neg\text{true} \Rightarrow \text{false}}{\{\text{true}\}\ \textbf{while}\ \text{true}\ \textbf{do}\ \text{x:=1}\ \{\text{false}\}}\ (\text{conseq})$$

上記の導出木の {true} x:=1 {true} の部分は、

$$\text{true}[1/\text{x}] = \text{true}$$

であることにより、assignment axiom から成り立つ。

また、上記の導出木中で、assignment axiom を assign、conseqence rule を conseq、while rule を while と略記している。

2　置換の定義

assignment axiom で使われている、論理式の置換の定義を示す。まず、この講義の Hoare 論理で扱う論理式を以下のように定義する（配布資料ではスペースが足りないので論理式の定義を与えていない）。

$$\begin{array}{rcl}
P & := & \text{true} \mid \text{false} \\
 & \mid & P \wedge P \mid P \vee P \mid \neg P \mid P \Rightarrow P \\
 & \mid & E \leq E \mid E \geq E \mid E < E \mid E > E \mid E = E \\
E & := & N \\
 & \mid & V \\
 & \mid & E + E \\
 & \mid & E - E \\
N & := & \cdots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \cdots \\
V & := & \text{x} \mid \text{y} \mid \text{z} \mid \cdots
\end{array}$$

このような定義を帰納的定義 (inductive definition) という（これについては講義の範囲外)。上記で定義された論理式に対して置換を以下のように帰納的に定義する（これも講義の範囲外)。

$$
\begin{array}{rcll}
P[E/x] & = & \text{case } P \text{ of} & \\
 & & \text{true} & \rightarrow \quad \text{true} \\
 & & \text{false} & \rightarrow \quad \text{false} \\
 & & P_1 \wedge P_2 & \rightarrow \quad P_1[E/x] \wedge P_2[E/x] \\
 & & P_1 \vee P_2 & \rightarrow \quad P_1[E/x] \vee P_2[E/x] \\
 & & \neg P & \rightarrow \quad \neg P[E/x] \\
 & & P_1 \Rightarrow P_2 & \rightarrow \quad P_1[E/x] \Rightarrow P_2[E/x] \\
 & & E_1 \leq E_2 & \rightarrow \quad E_1[E/x] \leq E_2[E/x] \\
 & & E_1 \geq E_2 & \rightarrow \quad E_1[E/x] \geq E_2[E/x] \\
 & & E_1 < E_2 & \rightarrow \quad E_1[E/x] < E_2[E/x] \\
 & & E_1 > E_2 & \rightarrow \quad E_1[E/x] > E_2[E/x] \\
 & & E_1 = E_2 & \rightarrow \quad E_1[E/x] = E_2[E/x] \\
E[E_0/x] & = & \text{case } E \text{ of} & \\
 & & N & \rightarrow \quad N \\
 & & E_1 + E_2 & \rightarrow \quad E_1[E_0/x] + E_2[E_0/x] \\
 & & E_1 - E_2 & \rightarrow \quad E_1[E_0/x] - E_2[E_0/x] \\
 & & V & \rightarrow \quad \text{if } V = x \text{ then } E_0 \text{ else } V
\end{array}
$$

3 Hoare tripleの表記法について

Hoare triple の書き方はさまざまなものが使われる。講義資料では $\{P_1\}\ S\ \{P_2\}$ の形の表記法を用いたが、Hoare の論文 [1] では $P_1\ \{S\}\ P_2$ のようにプログラムの方を中括弧で囲んでいる。

4 線形探索コードに対するHoare tripleの例

ここでは、参考書 [2] の 3.5 節の内容に沿って、線形探索コードに対する Hoare triple の例を提示する。授業で説明したように、命令型言語で制御フローが入口 1 箇所、出口 1 箇所になっている文は、その文の入口と出口で成り立つ条件（事前条件、事後条件）で意味を定めることができる。線形探索は番兵（sentinel）を使うと効率良く行うことができる。Pascal では、以下のようなコード断片を書けば良い。

```
A[0] := x;
i := n;
while x <> A[i] do i := i-1
```

これは、配列 A から、x と等しい値が入っている場所を線形探索で見つけるコード断片である。<>は左右の式の値が等しくないことを判定する演算子である。このプログラムを実行すると、実行直後において変数 i の値が以下のようになる。

- x が A[1] から A[n] の範囲内にある場合は、それらのうち一番右側の場所の添字。（A[1] ...A[n] が左から右へ並んでいる場合）
- 見つからなかったら、0。

実際にこうなることを（配列が扱えるように拡張した）Hoare 論理を使って証明したい。それには、以下の Hoare triple が証明できればよい。

```
{n≥1}
A[0] := x;
i := n;
while x <> A[i] do i := i-1
{(i=0 ∧ x∉A[1...n]) ∨ (0<i≤n ∧ x=A[i] ∧ x∉A[i+1...n])}
```

ただし、$x \notin a[i \ldots j]$ は、x が $a[i]$ から $a[j]$ の範囲にないことを表すものとする。$i > j$ の場合はこれは成り立つものとする。

以下を前提とする。

- x に探したい値が入っている。
- 配列 A は 0 から n の範囲の添字が使える。
- 配列 A の中で、添字 1 から n の範囲で探索する。A[0] は番兵のための場所である。
- 配列 A の要素は整数型であり、i, x, n は整数型の変数である。
- $\mathtt{n} \geq 1$

上記 Hoare triple は、$\{P\}$ `while x<>A[i] do i:=i-1` $\{Q\}$ が成り立てば成り立つ。ただし、P は

$$\mathtt{x} = \mathtt{A[0]} \wedge \mathtt{x} \notin \mathtt{A[i+1 \ldots n]} \wedge 0 \leq \mathtt{i} \leq \mathtt{n}$$

を表し、Q は

$$\mathtt{x} = \mathtt{A[i]} \wedge \mathtt{x} \notin \mathtt{A[i+1 \ldots n]} \wedge 0 \leq \mathtt{i} \leq \mathtt{n}$$

を表す。（このことを示すには配列の要素への代入に対応するように Hoare logic を拡張する必要があり、省略する。）Hoare triple $\{P\}$ `while x<>A[i] do i:=i-1` $\{Q\}$ の証明木は以下の通りである。

$$\dfrac{\dfrac{\dfrac{P \wedge \mathtt{x<>A[i]} \Rightarrow P[\mathtt{i-1/i}] \quad \overline{\{P[\mathtt{i-1/i}]\}\ \mathtt{i:=i-1}\ \{P\}}\ (\text{assign})}{\{P \wedge \mathtt{x<>A[i]}\}\ \mathtt{i:=i-1}\ \{P\}}\ (\text{conseq})}{\{P\}\ \mathtt{while\ x<>A[i]\ do\ i:=i-1}\ \{P \wedge \neg \mathtt{x<>A[i]}\}}\ (\text{while}) \qquad P \wedge \neg \mathtt{x<>A[i]} \Rightarrow Q}{\{P\}\ \mathtt{while\ x<>A[i]\ do\ i:=i-1}\ \{Q\}}\ (\text{conseq})$$

参考文献

[1] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 583, 1969.

[2] Ravi Sethi. *Programming Languages Concepts and Constructs, second edition.* Addison Wesley, 1996.