Modelli della concorrenza - Dispense

Alessandro Vasquez

November 11, 2016

Indice

Introduzione

Sia dato un programma P. Ci si vuole convincere del suo buon funzionamento. Vi sono tre diverse strade che si posso intraprendere:

- Sperimentazione
- Metodo assiomatico
- Model-checking

Questo corso tratterà le ultime due metodologie. Più precisamente, si dividerà in tre parti:

1. Correttezza dei programmi sequenziali
2. Modelli della concorrenza
3. Model-checking

Si riporta ora un esempio che introduce all'applicazione del metodo assiomatico.

Esempio 0.0.1. Sia data la seguente funzione:

```
int f (int n, int v[]){ //n = dimensione di v
   int x = v[0];
   int h = 1;
   while (h < n){
      if (x < v[h]){
         x = v[h];
         h = h+1;
      }
   }
   return x;
}
```

A una prima analisi la funzione sembra ordinare il vettore in modo decrescente. Più formalmente, si può assumere che i requisiti della funzione siano i seguenti:

$$\begin{aligned} \forall i \in \{0, .., n-1\} &\ :\ v[i] \leq x, \\ \exists i \in \{0, .., n-1\} &\ :\ x = v[i]. \end{aligned} \tag{1}$$

Si vuole dimostrare tali requisiti per induzione sulla dimensione del vettore v. Sia data quindi la seguente ipotesi induttiva:

$$Hp\ ind : \begin{cases} \forall i \in \{0, .., h-1\} \ : \ v[i] \leq x, \\ \exists i \in \{0, .., h-1\} \ : \ x = v[i]. \end{cases}$$

È necessario effettuare il passo induttivo e dimostrare tali proprietà per i casi base.

Formalizzando l'esempio 0.0.1 si ottiene la tripla:

$$\{n > 0\} \ P \ \{\alpha\}.$$

Dove $n > 1$ è chiamata precondizione, P è il programma o la funzione e α è la postcondizione che consiste dei requisiti esplicitati in (1).
Tale formalismo sarà ripreso nel capitolo 1.

Part I

Correttezza dei programmi sequenziali

Chapter 1

Logica di Hoare

1.1 Correttezza parziale

Riprendendo le strutture delle triple, si vuole ora costruire un sistema formale basato sulla deduzione naturale e sui seguenti assiomi.

1.1.1 Assiomi e regole di inferenza

Istruzione vuota Sia data una formula proposizionale α. Allora:

$$\frac{}{\{\alpha\}\ skip\ \{\alpha\}}$$

Assegnamento Sia data una formula proposizionale α, una variabile x e un'espressione E. Allora:

$$\frac{}{\{\alpha\left[\frac{E}{x}\right]\}\ x := E\ \{\alpha\}}$$

Conseguenza Siano date le formule proposizionali $p,\ q,\ p_1,\ q_1$ e il programma P. Allora:

$$\frac{p_1 \to p \qquad \{p\}\ P\ \{q\}}{\{p_1\}\ P\ \{q\}} \qquad\qquad \frac{\{p\}\ P\ \{q\} \qquad q \to q_1}{\{p\}\ P\ \{q_1\}}$$

Dalle ultime due deduzioni si ottiene induttivamente la seguente:

$$\frac{p_1 \to p \qquad \{p\}\ P\ \{q\} \qquad q \to q_1}{\{p_1\}\ P\ \{q_1\}}$$

Sequenza Siano date le formule proposizionali $p,\ q,\ r$ e le istruzioni $C_1,\ C_2$. Allora:

$$\frac{\{p\}\ C_1\ \{q\} \qquad \{q\}\ C_2\ \{r\}}{\{p\}\ C_1\, ;\, C_2\ \{r\}}$$

La regola di sequenza è utile nei cicli il cui corpo presenta istruzioni diverse che coinvolgono le stesse variabili.

Iterazione Siano dati un ciclo iterativo, la relativa condizione B, una sua invariante i, e il corpo del ciclo C. Allora:

$$\frac{\{i \wedge B\}\ C\ \{i\}}{\{i\}\ \texttt{while B do C od}\ \{\mathrm{i} \wedge \neg \mathrm{B}\}}$$

1.1.2 Regole di derivazione ottenibili induttivamente

Siano date le formule proposizionali $p,\ q,\ r$ e il programma P. Allora:

$$\frac{\{p\}\ P\ \{q\} \qquad \{r\}\ P\ \{q\}}{\{p \vee r\}\ P\ \{q\}}$$

Nel caso in cui $r = \neg p$, la precondizione nella conclusione della deduzione diventa una tautologia:

$$\frac{\{p\}\ P\ \{q\} \qquad \{\neg p\}\ P\ \{q\}}{\{p \vee \neg p\}\ P\ \{q\}}$$

Controllo di flusso Siano date le formule proposizionali $p,\ q$. Siano dati inoltre una condizione B, un blocco P (eseguito solo se B è verificata) e un blocco Q (eseguito altrimenti). Sia infine R il blocco che include l'istruzione di controllo su B e i blocchi P e Q. Vale la seguente deduzione:

$$\frac{\{p \wedge B\}\ P\ \{q\} \qquad \{p \wedge \neg B\}\ Q\ \{q\}}{\{p\}\ R\ \{q\}}$$

1.2 Correttezza totale

Si vuole verificare la terminazione di un programma dato. La non-terminazione può avvenire solo in presenza di cicli: ci si concentrerà sullo studio di programmi iterativi. Si supponga di avere un programma P che presenta un ciclo while W il cui corpo è denotato con C.
Si supponga di aver determinato un invariante i del ciclo. Dimostrare che il P termina significa determinare una espressione E e un invariante p, indipendente da E e da i, tali che:

1. $p \to E \geq 0$
2. $\vdash_p \{p \wedge B \wedge E = K\}\ C\ \{E < K\}$

1.2.1 Definizioni e notazioni

Definizione 1.2.1. Sia V l'insieme delle variabili di un programma P. Uno stato σ di P è una funzione

$$\sigma : V \to \mathbb{Z}.$$

Definizione 1.2.2. L'insieme di tutti gli stati di un programma è l'insieme:

$$\Sigma\{\sigma : V \to \mathbb{Z}\}.$$

Definizione 1.2.3. L'insieme di tutte le formule proposizionali costruite a partire da V di chiama Π.

Dati $\sigma \in \Sigma$ e $p \in \Pi$, si dirà che la formula p è valida nello stato σ usando la notazione

$$\sigma \models p.$$

Definizione 1.2.4. L'insieme di tutte le possibili asserzioni vere per lo stato σ è l'insieme

$$t(\sigma) = \{p \in \Pi : \sigma \models p\}.$$

Definizione 1.2.5. L'insieme degli stati che rendono vera la formula p è l'insieme

$$m(p) = \{\sigma \in \Sigma : \sigma \models p\}.$$

Tale insieme è definito come l'estensione della formula p.

Lemma 1.2.1. Siano $S \subseteq \Sigma$ e $F \subseteq \Pi$. Allora

$$t(S) = \{p \in \Pi : \forall s \in S : \sigma \models p\} = \bigcap_{\sigma \in S} t(\sigma),$$

$$m(F) = \{\sigma \in \Sigma : \forall p \in F : \sigma \models p\} = \bigcap_{p \in F} t(p).$$

Lemma 1.2.2. Siano $A, B \in \Sigma$ tali che $A \subseteq B$. Allora

$$t(A) \supseteq t(B).$$

Lemma 1.2.3. Siano $A, B \in \Pi$ tali che $A \subseteq B$. Allora

$$m(A) \supseteq m(B).$$

Lemma 1.2.4. Siano p, q formule proposizionali. Grazie alla definizione ricorsiva di formula proposizionale, valgono le seguenti proprietà:

- $m(\neg p) = \Sigma \setminus m(p)$
- $m(p \vee q) = m(p) \cup m(q)$
- $m(p \wedge q) = m(p) \cap m(q)$
- $(m \rightarrow q) = (\Sigma \setminus m(p)) \cup m(q)$

in cui l'ultima uguaglianza è data dal fatto che $p \rightarrow q \equiv \neg p \vee q$[1].

1.2.2 Determinare la precondizione

Si supponga di avere una tripla di Hoare priva della precondizione: $P\{q\}$. Si rende necessario trovare un metodo per determinarne la precondizione. Tramite l'assioma dell'assegnamento è possibile ottenere la precondizione più debole possibile[2].

Esempio 1.2.1. Si supponga di avere la seguente tupla:

$$C_1; C_2\{q\}.$$

Tramite l'assioma dell'assegnamento applicato a $C_2\{q\}$ posso ottenere la tripla

$$\vdash \{wp(C_2, q)\}\ C_2\ \{q\}^3.$$

Applicando poi l'assioma di assegnamento e l'assioma di sequenza a C_1, ottengo la tripla

$$\{wp(C_1, wp(C_2, q))\}\ C_1; C_2\ \{q\},$$

nella quale compare la più debole precondizione della tupla iniziale.

Esempio 1.2.2. Sia dato un generico programma iterativo $P\{q\}$:

```
while B do
C
od
```

Si vuole determinare la precondizione più debole. Vi possono essere due possibili condizioni: B è vera o no.

[1] Si veda la definizione di β-formula.
[2] La precondizione che pone meno vincoli allo stato iniziale del programma
[3] wp = weakest precondition

1. Se B è falsa prima del ciclo allora il while viene saltato e P equivale a un'istruzione vuota. La precondizione in questo caso deve coincidere con la postcondizione:
$$\{\neg B \wedge q\}.$$
2. Se B è vera prima del ciclo allora posso pensare di provare ad eseguire una volta il corpo del ciclo subito prima del ciclo stesso:

```
C
while B do
C
od
```

Posso quindi estrarre la precondizione più debole di C:

$$wp(C, q) = (B \wedge wp(C; P, q)).$$

Mettendo insieme le due precondizioni ottengo:

$$wp(P, q) = \{(\neg B \wedge q) \vee (B \wedge wp(C; P, q))\}.$$

Il risultato dell'esempio 1.2.2 potrebbe sembrare insoddisfacente: si è giunti a una formula più complessa di quella iniziale.
In effetti il calcolo di wp per un programma iterativo non porta meccanicamente a un risultato: tale calcolo non definisce un algoritmo, la soluzione può essere determinata grazie a metodologie euristiche informalmente giustificate. Tuttavia, nei programmi non iterativi il calcolo di wp può essere usato meccanicamente per determinare una soluzione.