Lezione 13 20/11/2023

prima parte su carta

```
int f (int n, const int v[]) {

    int x = v[0];

    int h = 1;
    x è il max in v[0..h-1]
    while (h < n) {
          x è il max in v[0..h-1], h<n
          if (x < v[h])
              x = v[h];
          x è il max in v[0..h], h<n
          h = h + 1;
          x è il max in v[0..h-1], h ≤ n
    }

    return x;
}
```

```
int f (int n, const int v[]) {
    n > 0
    int x = v[0];

    int h = 1;

    while (h < n) {

          if (x < v[h])
              x = v[h];

          h = h + 1;

    }
    x è il max in v[0..h-1], h=n
    return x;
}
```

Logica proposizionale — sintassi

$PA = \{ p_1, p_2, \dots, p_i, \dots \}$ proposizioni atomiche

$\perp, \top$ costanti logiche

$\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ connettivi

$($, $)$ delimitatori

Definiamo induttivamente F_P insieme delle formule ben formate

1. $\perp, \top \in F_P$
2. Per ogni $p_i \in PA$, $p_i \in F_P$
3. Se $A, B \in F_P$, allora $(\neg A), (A \vee B), (A \wedge B), \dots \in F_P$

Esempio $(x < 0 \wedge (\neg (Y \geq Z)))$

Logica proposizionale — semantica

Funzione di valutazione $v : PA \rightarrow \{0, 1\}$ (0 FALSO, 1 VERO)

Estendiamo induttivamente v a F_P

$I_v : F_P \rightarrow \{0, 1\}$ interpretazione

1. $I_v(\perp) = 0 \quad I_v(\top) = 1$
2. Per ogni $p_i \in PA \quad I_v(p_i) = v(p_i)$
3. $I_v(\neg A) = 1 - I_v(A)$

   $I_v(A \vee B) = 1$ se, e solo se, $I_v(A) = 1$ o $I_v(B) = 1$

   ...

   $I(A \rightarrow B) = 1$ se, e solo se, $I_v(B) = 1$ o $I_v(A) = 0$

Logica proposizionale – semantica

A è SODDISFATTA da I_V se $I_V(A) = 1$

A è SODDISFACIBILE se esiste I_V tale che $I_V(A) = 1$

A è una TAUTOLOGIA se $I_V(A) = 1$ per ogni I_V

A è CONTRADDITTORIA se $I_V(A) = 0$ per ogni I_V

Esempi di tautologie $A \to (B \to A)$ $\quad \bot \to A$

EQUIVALENZA LOGICA $\quad A \equiv B$ se, e solo se $I_V(A) = I_V(B)$ per ogni I_V

$$A \wedge (A \vee B) \equiv A \qquad \neg\neg A \equiv A \qquad \neg(A \wedge B) \equiv \neg A \vee \neg B$$

$$A \vee \neg A \equiv \top \qquad A \to B \equiv \neg B \to \neg A \qquad A \wedge \neg A \equiv \bot$$

Modello

$$M \subseteq PA \qquad I_M : F_P \to \{0,1\} \qquad I_M(p_i) = 1 \text{ sse } p_i \in M$$

$$M \vDash A$$ M MODELLA A, A è SODDISFATTA in M

Logica proposizionale – apparato deduttivo

Regola di inferenza $\quad \dfrac{A_1, \dots, A_N}{B}$ $\quad A_i \in F_P$ premesse, $B \in F_P$ conclusione

Calcolo della deduzione naturale

$$\frac{A_1 \quad A_2}{A_1 \wedge A_2} \qquad \frac{A_1 \wedge A_2}{A_1} \qquad \frac{A_1 \wedge A_2}{A_2}$$

$$\frac{A \quad A \to B}{B} \text{ MODUS PONENS} \qquad \frac{A \to B \quad \neg B}{\neg A} \text{ MODUS TOLLENS}$$

...

Dimostrazione Catena di applicazioni di regole $\quad A_1, \dots, A_N \vdash B$

Teorema Se $A_1, \dots, A_n \vdash B$ allora $A_1, \dots, A_n \vDash B$ validità, correttezza

Teorema Se $A_1, \dots, A_n \vDash B$ allora $A_1, \dots, A_n \vdash B$ completezza

Una logica per ragionare sui programmi

Primo livello

Proposizioni atomiche: $x \leq 5 \quad x > y \quad z = 2^y \quad \dots$

Stato della memoria

V insieme delle variabili del programma

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

$x \leq 5$ è vera in σ se $\sigma(x) \leq 5$

$\sigma \models \alpha$ "α è vera in σ"

Una logica per ragionare sui programmi

Secondo livello

$$\underbrace{\{\alpha\}}_{\text{precondizione}} \quad \overbrace{C}^{\text{"programma" (comando)}} \quad \underbrace{\{\beta\}}_{\text{postcondizione}}$$

tripla di Hoare

$\boxed{\text{Se}}$ il comando C viene eseguito a partire da uno stato della memoria nel quale α è vera, $\boxed{\text{allora}}$ l'esecuzione termina, e nello stato finale β è vera