

# 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;
}
return x;

```

## Logica proposizionale – sintassi

$PA = \{ p_1, p_2, \dots, p_i, \dots \}$	proposizioni atomiche
$\perp, T$	costanti logiche
$\neg, \vee, \wedge, \rightarrow, \leftrightarrow$	connettivi
$(, )$	delimitatori

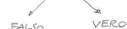
Definiamo induttivamente  $F_p$  insieme delle formule ben formate

1.  $\perp, T \in F_p$
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  $v: PA \rightarrow \{ 0, 1 \}$



Estendiamo induttivamente  $v$  a  $F_p$

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

1.  $I_v(\perp) = 0 \quad I_v(T) = 1$
2. Per ogni  $p_i \in PA \quad I_v(p_i) = v(p_i)$
3.  $I_v(\neg A) = 1 - I_v(A)$   
 $I_v(A \vee B) = 1 \quad \text{se, e solo se, } I_v(A) = 1 \circ I_v(B) = 1$   
 $\dots$   
 $I_v(A \rightarrow B) = 1 \quad \text{se, e solo se, } I_v(B) = 1 \circ I_v(A) = 0$

## 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  $A \rightarrow (B \rightarrow A)$      $\perp \rightarrow A$

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

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

$$A \vee \neg A \equiv T \quad A \rightarrow B \equiv \neg B \rightarrow \neg A \quad A \wedge \neg A \equiv \perp$$

### Modello

$$M \subseteq PA \quad I_M: F_p \rightarrow \{0,1\} \quad I_M(p_i) = 1 \text{ sse } p_i \in M$$

$M \models A$      $M$  MODELLO DI  $A$ ,  $A$  è SODDISFATTA in  $M$

## Logica proposizionale – apparato deduttivo

Regola di inferenza  $\frac{A_1, \dots, A_N}{B}$      $A_i \in F_p$  premesse     $B \in F_p$  conclusione

### Calcolo della deduzione naturale

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

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

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

$$\frac{A \quad A \rightarrow B}{B} \quad \text{MODUS PONENS}$$

$$\frac{A \rightarrow B \quad \neg B}{\neg A} \quad \text{MODUS TOLLENS}$$

...

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

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

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

## Una logica per ragionare sui programmi

### Primo livello

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

### Stato della memoria

$V$  insieme delle variabili del programma

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

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

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

## Una logica per ragionare sui programmi

### Secondo livello

→ "programma" (comando)

$$\{\alpha\} C \{\beta\}$$

↙  
precondizione

↘  
postcondizione

tripla di Hoare

Se il comando  $C$  viene eseguito a partire da uno stato della memoria nel quale  $\alpha$  è vera, allora l'esecuzione termina, e nello stato finale  $\beta$  è vera