26/03/2024

LEZ 7

SIAMO:

- $F = \Sigma_1^* \to \Sigma_2^*$ / $L_F = \{(x,y) \in \Sigma_1^* \times \Sigma_2^* : y = F(x)\}$

↳ LINGUAGGIO DI F

TEOREMA 3.14 (dispensa 3)

$\boxed{\text{SE } L_F \text{ È DECIDIBILE} \Rightarrow F \text{ CALCOLABILE}}$

DIM

COSTRUZIONE m.d.t. TRASDUTTORE, T_F, CON:

- INPUT → $x \in \Sigma^*$
- OUTPUT → $y = f(x)$

- SIA T MACCHINA CHE DECIDE L_F

$L_F \Rightarrow \forall (x,y) \in \Sigma_1^* \times \Sigma_2^* \; O_T(x,y) \begin{cases} q_A \text{ IF } (x,y) \in L \\ q_R \text{ IF } (x,y) \notin L \end{cases}$

X DIM CHE f CALCOLABILE, DOBBIAMO VERIFICARE CHE

$\forall y \in \Sigma_2^* \, [T(x,y) = q_A ?]$

FUNZIONAMENTO T_F → 3 NASTRI

T_F: – INPUT $x \in \Sigma_1^*$ (SU N_1)

– WRITE 0 SU N_2.

① SCRIVI SU N_3 TUTTE LE PAROLE $\in \Sigma_2^*$ LUNGHE QUANTO IL VALORE SCRITTO SU N_1, SEPARATI DA *.

② EXTRACT y SU N_3, CONCATENA CON x, SIMULA $T(x,y)$.

$\not\exists$ TOTAL ⇓ PER x INDEFINI. LA MACCHINA GO TO ∞

- IF $O_T(x,y) = q_A \Rightarrow$ WRITE y SU N_4 (OUTPUT) E TERMINA.
- IF $O_T(x,y) = q_R \wedge N_3 \neq \emptyset \Rightarrow$ RETURN TO ②
- ELSE ++ VAL. IN N_2 e RETURN TO ①

M.d.C : CHURC-TURING TESI

è stato dimostrato che ogni modello di calcolo inventato non è più potente del modello della macchina di turing e viceversa

RIFERITO AI MODELLI CHE USANO IL CONCETTO DI ISTR. ELEMENTARE DEL MODELLO DI TURING.

↳ UNICHE MACCHINE "REALI" CHE CONOSCIAMO FINO AD ORA.

TESI DI CHURCH-TURING

TUTTO CIÒ CHE È CALCOLABILE È CALCOLABILE MEDIANTE UNA MACCHINA DI TURING

DIM → usando un nuovo modello di calcolo: il lunguaggio PascalMinimo

SET ISTRUZIONI PASCAL:

- $a \leftarrow b;$ / $A[i]$
- IF (COND) THEN (ELSE)
- WHILE (cond) do

QUINDI ∀ PROGRAMMA IN PASCAL ∃ MACCHINA T TRASDUTTORE, t.c. ∀ INPUT $\rightarrow F_T(x) = F_P(x)$

CONSID. UN PROGRAMMA P

```
INPUT : n, m, A[ ]
I=1;
WHILE (i≤n) do BEGIN
      IF (A[i] >i) tHEN  m ← A[i];
      i ← i+1;
END
RETURN m
```

1 TOLGO A[i] IN CONDIZIONE

modifico il testo, ogni riga = istruzione 'elementare'

INPUT : n, m, A[ ]

```
i=1;
WHILE (i≤n) do BEGIN
    SUCA ← A[i]
    IF (suca >i) tHEN   m ← A[i];
    i ← i+1;
END
RETURN m
```

USO VAR x CONFRONTO.

2 SU OGNI RIGA CI DEVE STARE 1 ISTRUZIONE

INPUT : n, m, A[ ]

```
i=1;
WHILE (i≤n) do BEGIN
    SUCA ← A[i]
    IF (suca >i) tHEN
        m ← A[i];
    i ← i+1;
END
RETURN m
```

CONSIDERO ORA LA MACCHINA SIMULATRICE T

- x ∀ variabile la memorizzo in un suo nastro