TESI DI CHURCH-TURING → VALIDO SOLO PER I MODELLI ATTUALI

TUTTO CIO' CHE È CALCOLABILE LO È MEDIANTE UNA MACCHINA DI TURING

PASCAL MINIMO

MODELLO DI CALCOLO BASATO SU: → SIMULA GLI ODIERNI LINGUAGGI DI PROGRAMMAZIONE

i) ASSEGNAZIONE:

$a \leftarrow b$

ii) SELEZIONE:

IF (COND) THEN .... ELSE

iii) ITERAZIONE:

WHILE (COND) DO

iv) STRUTTURE DATI:

ARRAY → $A[i]$

TEOREMA RELAZIONE FRA PASCAL MINIMO E MACCHINA DI TURING

$\forall P$ PROGRAMMA SCRITTO IN PASCAL MINIMO

$\exists$ UN T TRASDUTTORE CHE:

$\forall$ INPUT RESTITUISCE LO STESSO OUTPUT DI P

DIM

SCRIVIAMO UN PROGRAMMA P CON 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
```

ELIMINIAMO GLI ARRAY DALLE CONDIZIONI DEI CICLI

```
i = 1
WHILE (i ≤ n) DO BEGIN:
    AUSIL <- A[i];
    IF (AUSIL > i) THEN m <- A[i];
    i <- i+1;
END;
RETURN m
```

LASCIAMO UN'ISTRUZIONE PER RIGA
E LE NUMERIAMO

```
1  i = 1
2  WHILE (i ≤ n) DO BEGIN:
3      AUSIL <- A[i];
4      IF (AUSIL > i) THEN
5          m <- A[i];
6      i <- i+1; END;
7  RETURN m
```

TRADUCIAMO ORA IL PROGRAMMA P NELLA MACCHINA T

i) NASTRI:

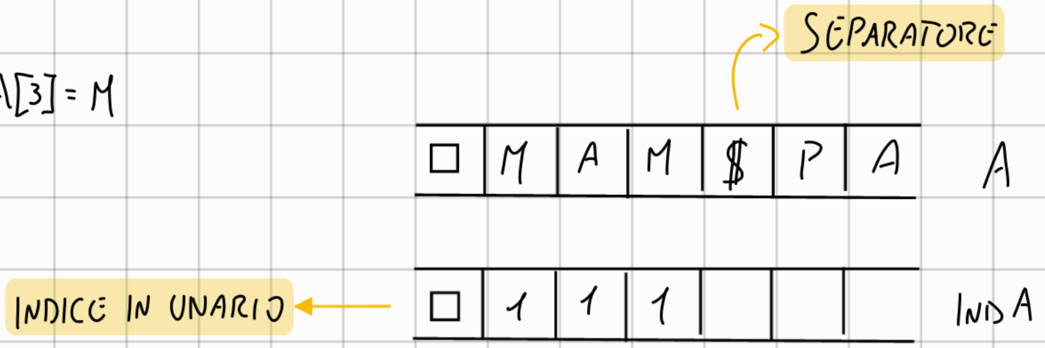
i.1) ∀ VARIABILE ASSOCIAMO UN NASTRO

i.2) ∀ ARRAY ASSOCIAMO 2 NASTRI:

UNO CON GLI ELEMENTI → PER SEMPLIFICARE OGNI CELLA HA UNA PAROLA

UNO CON L'INDICE ESAMINATO

A[3] = M



ii) STATI:

∀ RIGA CORRISPONDE UNO STATO

1) $i = 1$ → q_1

2) WHILE $(i \leq n)$ DO BEGIN: → q_2

3) AUSIL <— A[i]; → q_3

⋮ → ⋮

iii) SELEZIONE:

J) IF $(a > b)$ THEN BEGIN → $< q_J, (\text{SE } N_a > N_b), q_{J+1}, F >$

J+1) $b <- a$; → $< q_{J+1}, (\text{COPIA } N_a \text{ SU } N_b), q_{J+2}, F >$

J+2) $a <- 0$; END; → $< q_{J+2}, (\text{SCRIVI } 0 \text{ SU } N_a), q_{J+3}, F >$

J+3) → $< q_J, (\text{SE } N_b > N_a), q_{J+3}, F >$

DESCRIZIONE AD ALTO LIVELLO DI UN INSIEME DI QUINTUPLE

iV) ASSEGNAZIONE:

J) $a \leftarrow b;$ $\longrightarrow$ $< q_J,$ (COPIA IL CONTENUTO DI N_a SU N_b)$, q_{J+1}, F >$

$J+1$) ⋮

V) ITERAZIONE:

J) WHILE $(a<b)$ THEN BEGIN: $\longrightarrow$ $< q_J,$ (SE $N_a > N_b$)$, q_{J+1}, F >$

$J+1$) $b \leftarrow a;$ $\longrightarrow$ $< q_{J+1},$ (COPIA N_b SU N_a)$, q_{J+2}, F >$

$J+2$) $a \leftarrow 0;$ END; $\longrightarrow$ $< q_{J+2},$ (SCRIVI 0 SU N_a)$, q_J, F >$

$J+3$) ⋮ $\longrightarrow$ $< q_J,$ (SE $N_a > N_b$)$, q_{J+3}, F >$

TORNO INDIETRO

VI) ARRAY:

J) $A[i] \leftarrow b;$

$J+1$) ⋮

α	β	γ	∂	β	μ	$\square$	N_A

1	1	1	1	$\square$			$N_{IND\,A}$

$< q_J,$ (SCRIVI N_i SU $N_{IND\,A}$ IN UNARIO)$, q'_J, F >$

$< q'_J,$ (SCORRI N_i E $N_{IND\,A}$ VERSO DX FINO A $\square$, POI TORNO A SX)$, q^2_J, F >$

$< q^2_J,$ (COPIA N_b SU N_a)$, q_{J+1}, F >$

SIAMO RIUSCITI A SCRIVERE OGNI OPERAZIONE DEL PASCAL MINIMO COME QUINTUPLE DI UNA T

TEOREMA RELAZIONE FRA MACCHINA DI TURING E PASCAL MINIMO

$\forall\, T_U$ MACCHINA UNIVERSALE DI TURING
$\exists$ UN $\mathcal{P}_U$ PROGRAMMA IN PASCAL MINIMO CHE:
$\forall$ INPUT RESTITUISCE LO STESSO OUTPUT DI T_U

DIM

DEFINIAMO INNANZITUTTO LA MACCHINA T_U COME:

$$T_U = \langle \Sigma, Q, P, q_0, \{q_A, q_R\} \rangle \text{ AD 1 NASTRO}$$

ABBIAMO DIMOSTRATO CHE CON UN NASTRO POSSIAMO FARE QUALUNQUE COSA

DOVE P È DEL TIPO:

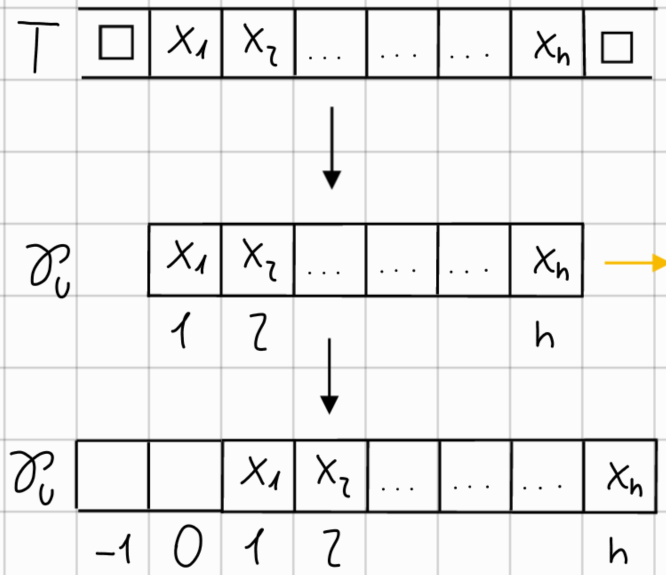
$$\begin{aligned} P = \{ &\langle q_{11}, S_{11}, S_{21}, q_{21}, m_1 \rangle, \\ &\langle q_{12}, S_{12}, S_{22}, q_{22}, m_2 \rangle, \\ &\langle q_{13}, S_{13}, S_{23}, q_{23}, m_3 \rangle, \\ &\vdots \\ &\langle q_{1k}, S_{1k}, S_{2k}, q_{2k}, m_k \rangle \} \end{aligned}$$

AVENDO P STRUTTURATO IN QUESTO MODO POSSIAMO GESTIRE GLI ARRAY DI $\mathcal{P}_U$ IN MODO CHE:

ARRAY:

- $Q_1[1:k] \rightarrow Q_1[3] = q_{13}$
- $S_1[1:k]$
- $S_2[1:k]$
- $Q_2[1:k]$
- $M[1:k] \rightarrow M[3] = -1, 0, 1 \iff m_3 = Sx, F, Dx$
- $N[1:n] \rightarrow$ PAROLA IN INPUT

DOBBIAMO GESTIRE LA RAPPRENTAZIONE DEGLI ARRAY



POICHÉ I NASTRI SONO INFINITI E GLI ARRAY NO ABBIAMO BISOGNO DI RENDERLI DINAMICI

IMPLEMENTIAMO GLI INDICI NEGATIVI PER SIMULARE IL NASTRO INFINITO

SCRIVIAMO ORA L'ALGORITMO DI SIMULAZIONE $\mathcal{M}_U$

INPUT $\mathcal{M}_U$: $q_0, q_A, q_R, Q[1{:}k], S_1[1,k], S_2[1,k], Q_2[1,k], M[1,k], N[1,n]$

```
q <- q0;                 → STATO INIZIALE
Pc <- 1; Uc <- n;        → Pc = PRIMA CASELLA, Uc = ULTIMA CASELLA
t <- 1;
i <- 1;
WHILE (q ≠ qA ∧ q ≠ qR) DO BEGIN
    J <- 1;
    TROVATA <- FALSE;
    WHILE (J ≤ K ∧ TROVATA = FALSE) DO BEGIN
        IF (q = Q1[J] ∧ N[t] = S1[J]) THEN
            TROVATA <- TRUE; END;
        J <- J + 1;
    END;
    IF (TROVATA = FALSE) THEN
        IF (q ≠ qA) THEN q <- qR; END;
    END;
```

```
ELSE (% TROVATA = TRUE) BEGIN
    N[E] <- S[J-1];
    q <- Q_2[J-1];
    t <- t + M[J-1];
    IF (t > U_c) THEN
        U_c <- U_c + 1;
        N[t] = □; END;   → INIZIALIZZO IL CARATTERE □
    ELIF (t < P_c) THEN
        P_c <- P_c - 1;
        N[t] = □; END;   → INIZIALIZZO IL CARATTERE □
END;
END;
RETURN q;
```

PROGRAMMI PER MACCHINE NON DETERMINISTICHE → NON RICHIESTO PER L'ESAME

```
                                                    LUNGHEZZA DELLA COMPUTAZIONE
                                                          ↗
Q SIMULA RICORSIVO (Q q, INT t, Σ[ ]N; INT i);  --> TUTTI COSTANTI TRANNE N

INPUT: q0, qA, qR, Q1[1,k], S1[1,k], S2[1,k], Q2[1,k], M[1,k], N[1....n]
   IF (i = 0) THEN
       P <- q_i
   ELSE BEGIN
       J <- 1;
       RIGETTO <- VERO
       WHILE (J ≤ k ∧ q ≠ qA) DO BEGIN
          WHILE (J ≤ k ∧ [q ≠ Q1[J] ∧ N[t] ≠ S1[J]) DO
              J <- J+1; END;
          IF (J = k+1) THEN P <- qR;
          ELSE (q = Q1[J] ∧ N[t] = S1[J]) BEGIN
              N[t] <= S2[J];
              IF (t+M[J] > Vc) THEN BEGIN Vc <- Vc+1; N[t] <= □; END;
              ELIF (t+M[J] < Pc) THEN BEGIN Pc <- Pc-1; N[t] = □; END.
              P <- SIMULA RICORSIVO (Q2[J], t+M[J], N, i-1);
                                 └─> PUÒ ESSERE MODIFICATO SOLO QUI ALTRIMENTI
                                     NON VERIFICO TUTTE LE QUINTUPLE
              IF (P = qA) THEN q = qA;
              ELSE IF (P ≠ qR) THEN RIGETTO <- FALSO;
              N[t] <- S1[J]
              J <- J+1
          END;
   END;
                                 VA BENE ∀q∈Q, BASTA CHE P SIA DIVERSO DA qA E qR
                                          ↗
   IF (P ≠ qA ∧ RIGETTO = FALSE) THEN P <- q0
   RETURN P;
```