UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Computabilità, Complessità e Logica

Prof. Adriano Peron

Computabilità: Macchine di Turing

Estensione del potere espressivo

- Gli **automi regolari** sono adeguati a modellare sistemi con capacità di memoria limitata.
- Gli **automi con pila** hanno una capacità di memoria illimitata ma le limitazioni sull'uso della memoria (gestione LIFO) limita la capacità espressiva
- Le **Macchine di Turing** propongono un modello che cattura la massima espressività legata alla nozione di computazione.
  - Proposta da Alan Turing nel 1936
  - Modello più accurato di un computer general pourpose
  - Può fare qualsiasi cosa un computer possa fare
  - Ci sono tuttavia problemi che non è in grado di risolvere
- …….. **Sono i problemi che vanno oltre i limiti teorici della computabilità**

Macchina di Turing

- Estende il modello della macchina stato-transizione per mettendo l'uso libero di un supporto di memoria sequenziale di capacità infinita.
- Supporto di memoria:
  - Nastro diviso in celle
  - Ogni cella può essere vuota (simbolo speciale blanck) o contenere un simbolo.
  - Ogni cella può essere letta e scritta
  - Il nastro è illimitato a destra e limitato a sinistra (c'è una cella iniziale)
  - Il nastro viene scorso sequenzialmente da sinistra a destra o da destra a sinistra

Macchina di Turing

- **Origine storica.**
- **Nel 1900 in un celebre intervento a un congresso il matematico David Hilbert elenca 23 problemi matematici che a suo parere costituiscono una sfida per la matematica del XX secolo.**
- **Il decimo problema riguarda la possibilità di trovare una procedura che risponda (test si/no) sull'esistenza di soluzioni intere per un generico polinomio a coefficienti interi.**
- "a process according to which it can be determined by a finite number of operations"
- **Dimostrare l'impossibilità di definire una tale procedura richiede la definizione precisa di un concetto di computazione e di computabilità o se preferiamo di ALGORITMO.**

Macchina di Turing

- **Origine storica.**
- **Nel 1936 vengono proposte due definizioni**
  - **Alonzo Church propone il λ-calculus**
  - **Turing propone la definizione di una Macchina**
- **E' stato dimostrato che entrambe le definizioni sono equivalenti**
- **La possibilità di collegare la nozione di qualitativa di algoritmo con le definizioni di Turing e Church è stabilita dalla Tesi di Church-Turing**
- **Una funzione sui numeri naturali può essere calcolata con un metodo effettivo se e solo è computabile da una macchina di Turing.**

Macchina di Turing

- **Estende il modello della macchina stato-transizione per mettendo l'uso libero di un supporto di memoria sequenziale di capacità infinita.**
- **Strumento di lettura/scrittura/scansione:**
  - Testina di lettura/scrittura
  - E' posizionata su una cella
  - Può leggere e riscrivere il contenuto della cella in un passo elementare
  - Può spostarsi di una cella a destra (se non si trova nella cella iniziale), rimanere nella stessa posizione, spostarsi di una cella a sinistra.

Macchina di Turing

- Passo elementare di computazione:
- In un fissato istante della computazione la macchina:
  - si trova in un determinato stato di controllo (stato corrente)
  - Ha la testina puntata ad una cella del nastro (posizione corrente)
  - Legge il contenuto della cella
  - In relazione allo stato e al simbolo letto determina:
    - Il simbolo da sovrascrivere sulla cella
    - La direzione di scansione del nastro (destra/sinistra/fermo)
    - Il prossimo stato di controllo

Macchina di Turing

- **Computazione:**
- Sequenza di passi elementari.
- **Terminazione della computazione**
  - L'insieme di stati elementari contiene due stati speciali
  - **Accept**: la computazione termina con successo raggiungendo lo stato Accept
  - **Reject**: la computazione termina con insuccesso raggiungendo lo stato Reject
  - La computazione può non terminare (**Loop**)

- Negli automi regolari o a pila la computazione termina alla lettura dell'ultimo simbolo della parola di input.

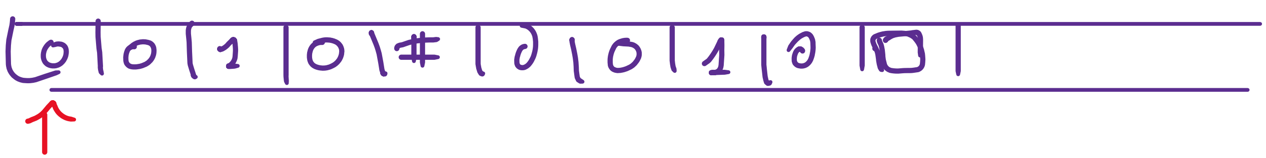
Macchina di Turing

- **Computazione:**
- Sequenza di passi elementari.
- **Terminazione della computazione**
  - L'insieme di stati elementari contiene due stati speciali
  - **Accept**: la computazione termina con successo raggiungendo lo stato Accept
  - **Reject**: la computazione termina con insuccesso raggiungendo lo stato Reject
  - La computazione può non terminare (**Loop**)

- Negli automi regolari o a pila la computazione termina alla lettura dell'ultimo simbolo della parola di input.

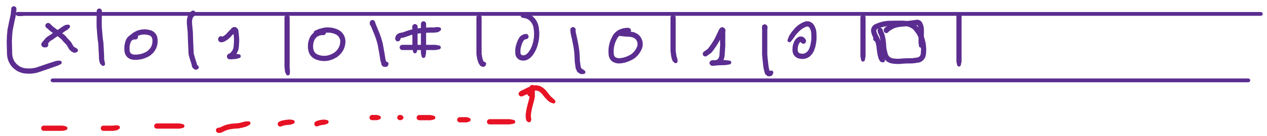
Macchina di Turing

- **MdT come accettore di linguaggi.**
- Il nastro nella condizione iniziale della macchina contiene la parola $w \in \Sigma$ da accettare accostata a destra (primo simbolo della parola nella prima cella).
- La parte del nastro non occupata dalla parola ha il simbolo blanck in ogni cella.
- La parola è inclusa nel linguaggio se la MdT termina la computazione nello stato Accept
- Esempio. Sia L={w#w: w$\epsilon\{0,1\}^*$}.

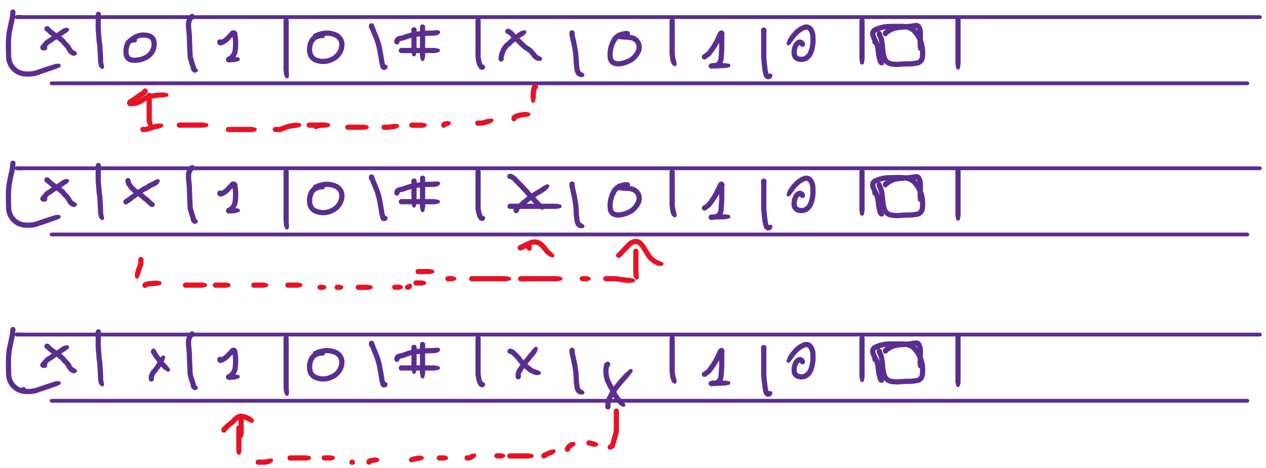


Macchina di Turing

- La macchina riconosce uno zero, marchia la casella come letta con una x e cerca uno 0 nella prima posizione dopo #

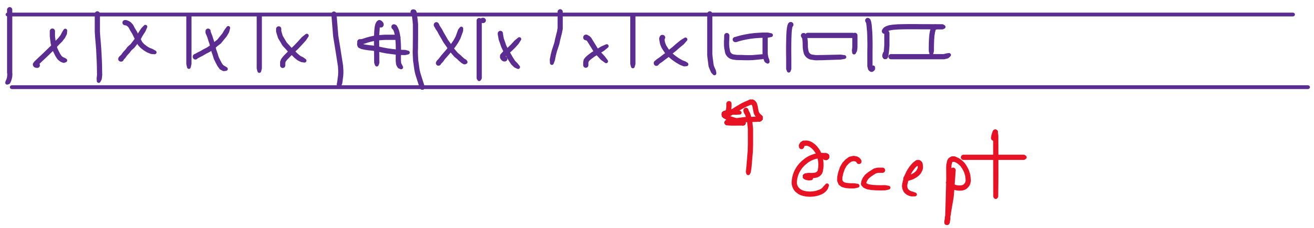


- La MdT trova la corrispondenza, la marchia con x come letta e ritorna al secondo simbolo da confrontare



Macchina di Turing

- **La computazione termina con accettazione quando tutte le corrisponenze sono accertate e la seconda stringa non è piu lunga della prima**



Macchina di Turing: sintassi

- **Una macchina di Turing è una struttura**
- $< \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{q_{accept}}, \boldsymbol{q_{reject}} >$
- $\boldsymbol{Q}$ **insieme finito degli stati di controllo**
- $\boldsymbol{\Sigma}$ **Alfabeto non contenente il simbolo blank** $\sqcup$
- $\boldsymbol{\Gamma}$ **Alfabeto per il nastro** $\boldsymbol{\Sigma} \subseteq \boldsymbol{\Gamma}, \sqcup \in \boldsymbol{\Gamma}$
- $\boldsymbol{q_0}$ **Lo stato iniziale**
- $\boldsymbol{q_{accept}}$ **Lo stato di accettazione**
- $\boldsymbol{q_{reject}}$ **Lo stato di rifiuto**
- **La funzione di transizione (macchina deterministica)**

$$\boldsymbol{\delta} : \boldsymbol{Q} \times \boldsymbol{\Gamma} \longrightarrow \boldsymbol{Q} \times \boldsymbol{\Gamma} \times \{\mathrm{L},\mathrm{R}\}$$

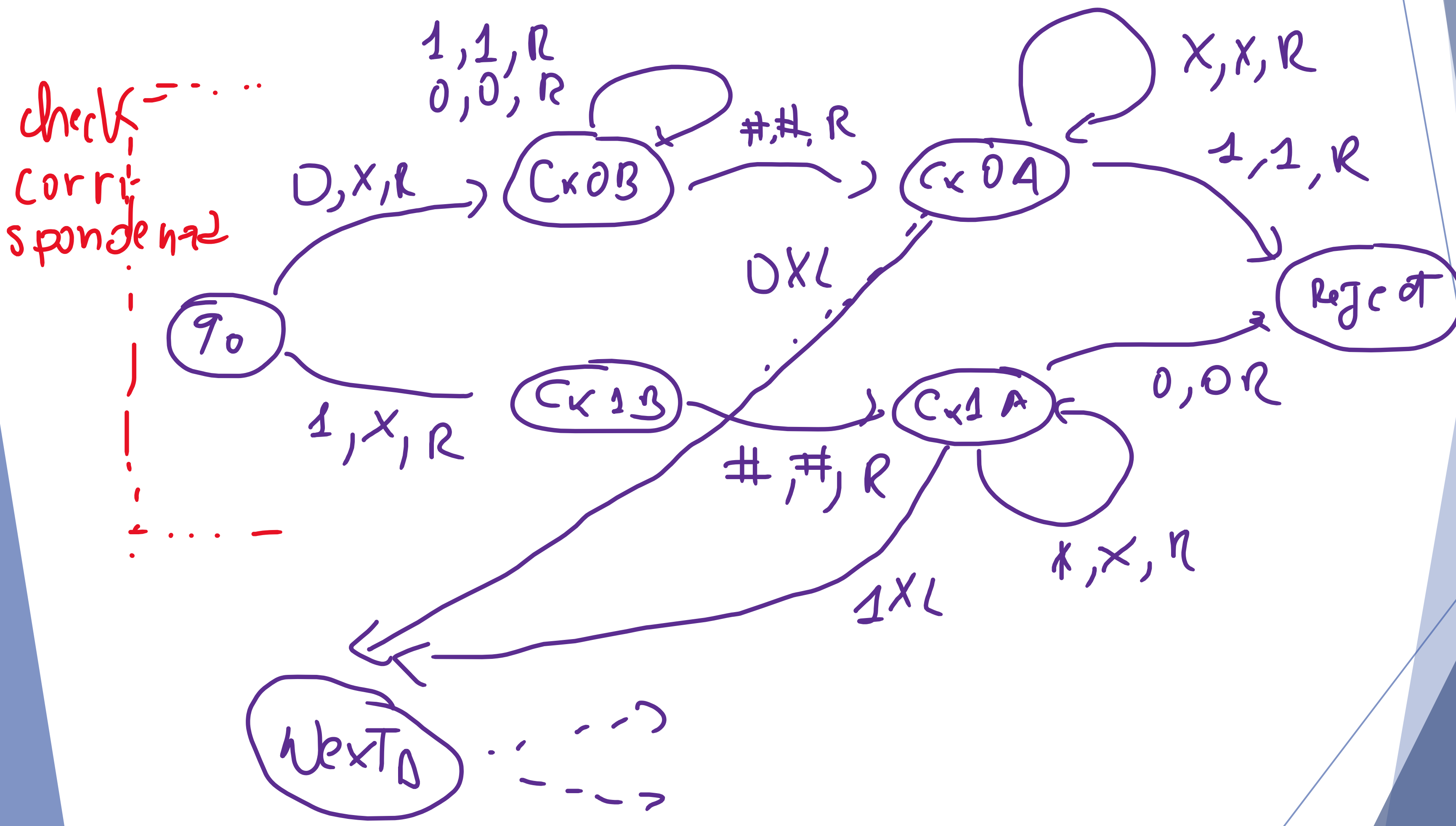
- **L,R indicano la direzione di spostamento della testina**

Macchina di Turing: esempio

- **Macchina per il linguaggio** L={w#w: w$\epsilon\{0,1\}^*$}.
- $< \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{q_{accept}}, \boldsymbol{q_{reject}} >$
- $\boldsymbol{\Sigma} = \{\mathbf{0}, \mathbf{1}, \#\}$ e $\boldsymbol{\Gamma} = \{\mathbf{0}, \mathbf{1}, \#, \mathbf{X}\}$
- $\boldsymbol{Q}$ è l'unione dei seguenti insiemi
- $\{\boldsymbol{q_0}, \boldsymbol{q_{accept}}, \boldsymbol{q_{reject}}\}$
- {Ck0A,Ck0B} cerca corrispondenza di 0
- {Ck1A,Ck1B} cerca corrispondenza di 1
- {NextA,NextB} Ritorno a sinistra
- {CkE} Check finale

Macchina di Turing: esempio

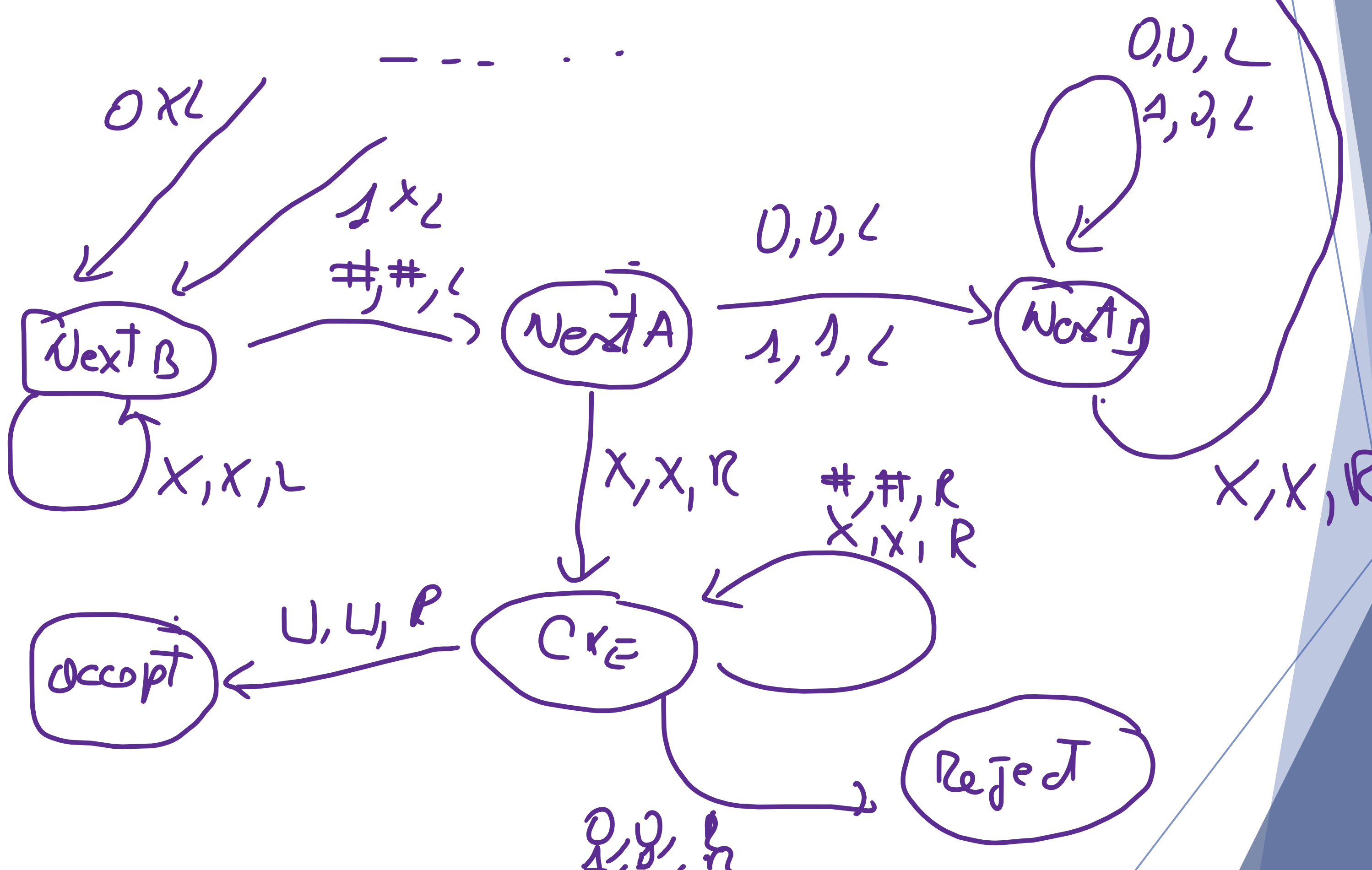
- **Macchina per il linguaggio** L={w#w: w$\in\{0,1\}^*$}.



Macchina di Turing: esempio

q0

- **Macchina per il linguaggio** L={w#w: w$\epsilon\{0,1\}^*$}.



Macchina di Turing: semantica

- $< Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} >$
- **Configurazione:** rappresenta uno snapshot della macchina in un istante. Descrive:
  - Lo stato di controllo corrente
  - La posizione della testina nel nastro
  - Il contenuto (finito) del nastro (il prefisso iniziale diverso da blank)

$$C \in \Gamma^* \times Q \times \Gamma \times \Gamma^*$$

- Esempio:

$$\gamma_1\gamma_2\ \gamma_3\ \gamma_4\ \gamma_5(\mathrm{q}, \gamma_6)\ \gamma_7\ \gamma_8\ \gamma_9\ \gamma_{10}$$

  - Il nastro ha le prime 10 celle occupate
  - La testina è posizionata sulla sesta cella
  - Lo stato di controllo è q

Macchina di Turing: semantica

- $< Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} >$

$$C \in \Gamma^* \times Q \times \Gamma \times \Gamma^*$$

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L,R\}$$

- **Passo di computazione**

$$\gamma_1\gamma_2\ \gamma_3\ \gamma_4\ \gamma_5(q, \gamma)\ \gamma_7\ \gamma_8\ \gamma_9\ \gamma_{10}$$

$$\gamma_1\gamma_2\ \gamma_3\ \gamma_4\ \gamma_5\gamma'(q'\gamma_7)\ \gamma_8\ \gamma_9\ \gamma_{10}$$

Con $(q, \gamma, q', \gamma', R)\ \epsilon\delta$

(mossa a destra)

$$\gamma_1\gamma_2\ \gamma_3\ \gamma_4(q'\gamma_5)\gamma'\gamma_7\ \gamma_8\ \gamma_9\ \gamma_{10}$$

Con $(q, \gamma, q', \gamma', L)\ \epsilon\delta$

(mossa a sinistra)

  - Si osservi che un passo di computazione altera solo due posizioni del nastro se la direzione è stabilita

Macchina di Turing: semantica

- $< Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} >$

$$C \in \Gamma^* \times Q \times \Gamma \times \Gamma^*$$

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L,R\}$$

- Una computazione è una sequenza di configurazioni

$$C_1 C_2 C_3 \ldots\ldots C_n$$

- C_1 è una configurazione iniziale del tipo $C_1 \in \{q_0\} \times \Gamma \times \Gamma^*$

  (testina in prima posizione e stato di controllo iniziale)

- C_{i+1} è ottenuto da C_i mediante un passo di computazione per ogni $1 \leq i < n$
- Lo stato di controllo in C_i con $1 \leq i < n$, è diverso da q_{accept} e q_{reject}.
- La computazione è accettante se lo stato di controllo di C_n è q_{accept}.

Macchina di Turing: semantica

- **Osservazioni:**
- **Una computazione che raggiunge lo stato di controllo di rifiuto q_{reject} o lo stato di accettazione q_{accept} termina (non può essere estesa).**
- **Una computazione può:**
  - **terminare nello stato di accettazione q_{accept}**
  - **Terminare nello stato di rifiuto q_{reject}**
  - **Non terminare!.**

Macchina di Turing come accettore di linguaggi

- Una MdT $< Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} >$ **accetta** una parola $w \in \Sigma^*$ $(w = w_1 \dots w_n)$ se esiste una computazione accettante dalla configurazione

$$(q_0, w_1)\ w_2\ \dots\ w_n$$

- Nella configurazione iniziale il contenuto del nastro è la parola w e la testina si posiziona sul primo simbolo.
- Il linguaggio **accettato** da MdT è l'insieme delle parole **accettate** da MdT.
- **Una MdT accetta un linguaggio L se accetta tutte le parole di L.**
- Riguardo alle parole w che non fanno parte del linguaggio L accettato da MdT due diverse situazioni possono determinarsi:
  - Esiste una computazione **di rifiuto** per w
  - Esiste una computazione **non-terminante** per w!

Macchina di Turing come decisore di linguaggi

- E' possibile richiedere un criterio più stretto per la decisone di una MdT riguardo a un linguaggio L:
- Fissata una parola w
  - MdT termina in stato di accettazione se w appartiene a L
  - MdT termina in stato di rifiuto se w non appartiene a L

- Fissato $L \subseteq \Sigma^*$,Se esiste una MdT M che accetta ogni parola di L e rifiuta ogni parola di $\overline{L}$ si sice che M **decide** L.
- Se un linguaggio L puo' essere **deciso** da una MdT può anche essere **riconosciuto** da una MdT.
- **Vale il viceversa?????**

MdT accettore e decisore di linguaggi

- Di nuovo il decimo problema di Hilbert.
- **Prendiamo il seguente linguaggio D**

D={p : p è un polinomio con radici intere}

Esiste una MdT che accetta D?

Idea.

- Inizialmente nel nastro si ha la codifica del polinomio.
- Codifico in una MdT un algoritmo che prova a sostituire iterativamente i valori delle variabili 0, 1, -1, 2, -2, ….
- Se la sostituzione trova una radice va in stato di accettazione altrimenti continua con l'iterazione.
- Se esiste una sostituzione intera delle variabili che fornisce una radice la macchina termina
- Se non esiste una radice la macchina non termina.

MdT accettore e decisore di linguaggi

- **D={p : p è un polinomio con radici intere}**

Esiste una MdT che decide D?

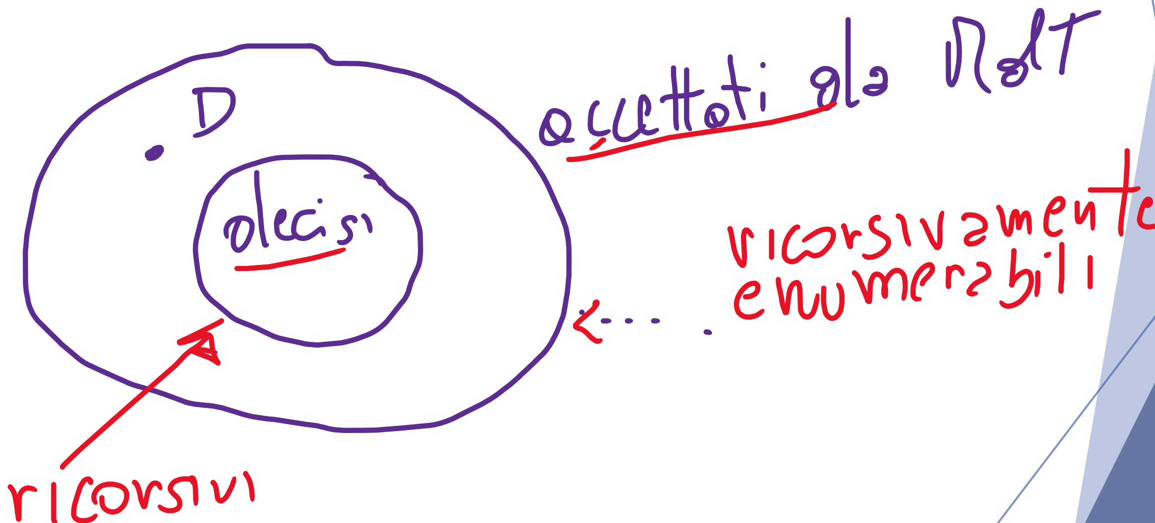
Si consideri un problem più semplice: polinomi su una sola variabile

DX={p : p è un polinomio a una sola variabile con radici intere}

- In questo caso è possibile esprimere intervallo di esistenza delle radici $\pm k \frac{c_{\max}}{c_1}$
  - K numero di termini
  - $c_{\max}$ coefficiente massimo del polinomio
  - c_1 coefficiente del termine di grado massimo
- E possibile limitare l'iterazione ai due estremi e rifiutare se gli estremi sono superati nel processo iterativo
- DX può essere sia accettato sia deciso!

MdT accettore e decisore di linguaggi

- **D={p : p è un polinomio con radici intere}**
- A differenze del caso dei polinomi a una variabile nel caso generale non è possibile limitare l'intervallo di esistenza delle radici.
- E' stato provato nel 1970 che D non può essere deciso (e può essere dunque solo accettato da una MdT)



Estensioni del modello

- E’ possibile considerare varianti della definizioni di una MdT:
  - MdT con più di un nastro
  - MdT non-deterministiche
- I modelli estesi danno vantaggi descrittivi ma non estendono l’espressività del modello di base.

MdT con nastro multiplo.

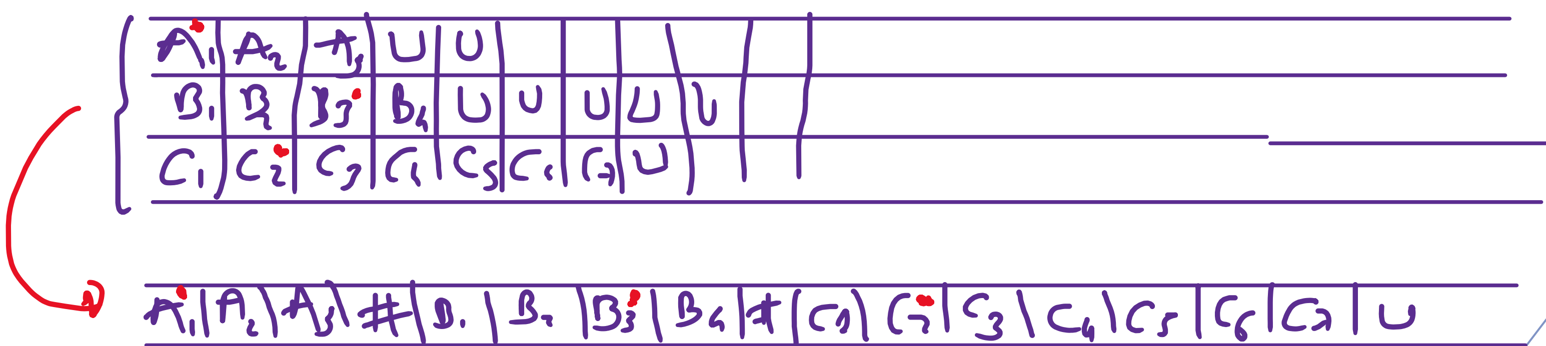
- La macchina può lavorare su k nastri infiniti simultaneamente.
- Ha una testina per ogni nastro
- In ogni passo:
  - si leggono k simboli (ciascuno su un nastro),
  - si riscrive il contenuto delle celle lette (separatamente in ciascun nastro)
  - si altera lo stato di controllo della macchina
  - Ci si sposta in ciascun nastro (lo spostamento in ciascun nastro è indipendente).
- Nella definizione della MdT con k nastri cambia solo la funzione di transizione

$$< \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{q_{accept}}, \boldsymbol{q_{reject}} >$$

$$\boldsymbol{\delta}: \boldsymbol{Q} \times \boldsymbol{\Gamma}^{\boldsymbol{k}} \longrightarrow \boldsymbol{Q} \times (\boldsymbol{\Gamma} \times \{\boldsymbol{L}, \boldsymbol{R}\})^{\boldsymbol{k}}$$

MdT con nastro multiplo.

- **Teorema.** **Per ogni MdT M a nastro multiplo esiste una MdT M' a unico nastro che riconosce il medesimo linguaggio.**
- **Idea della prova.**
- **La macchina M' simula il comportamento della macchina M su un unico nastro.**
- **Il contenuto dei k nastri è riportato sull'unico nastro separato da un carattere speciale.**



MdT con nastro multiplo.

- Le posizioni correnti dei k nastri sono marcate con simboli speciali.
- Simulazione di uno step.
  - M’ scandisce il nastro per determinare il contenuto di tutte le k posizioni correnti (una scansione)
  - M’ usa la funzione di transizione di M per determinare il nuovo contenuto delle k celle e le mosse delle k testine
  - M’ riscrive 2k celle per aggiornare le posizioni delle testine (k celle) e il contenuto delle celle correnti precedenti (k celle) (una scansione)
  - Se nella simulazione serve aggiungere una cella ad un nastro codificato serve spostare a destra di una posizione tutto il contenuto (una scansione)

Inserimento $\square$

shift di 1 posizione

MdT non-deterministica.

- La macchina può avere transizioni non deterministiche: dalla configurazione corrente possono essere derivate più configurazioni correnti
- Nella definizione della MdT non-deterministica cambia solo la funzione di transizione

$$< Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} >$$

$$\delta: Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L,R\}}$$

- Ad uno stato di controllo ed a un simbolo del nastro viene associato un insieme di triple stato-simbolo-direzione tra le quali scegliere.
- **Una MdT non-deterministica** $< Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} >$ **accetta una parola** $w \in \Sigma^*$ $(w = w_1 \ldots w_n)$ **se esiste una computazione accettante dalla configurazione**

$$(q_0, w_1)\ w_2 \ldots w_n$$

Determinizzazione di MdT.

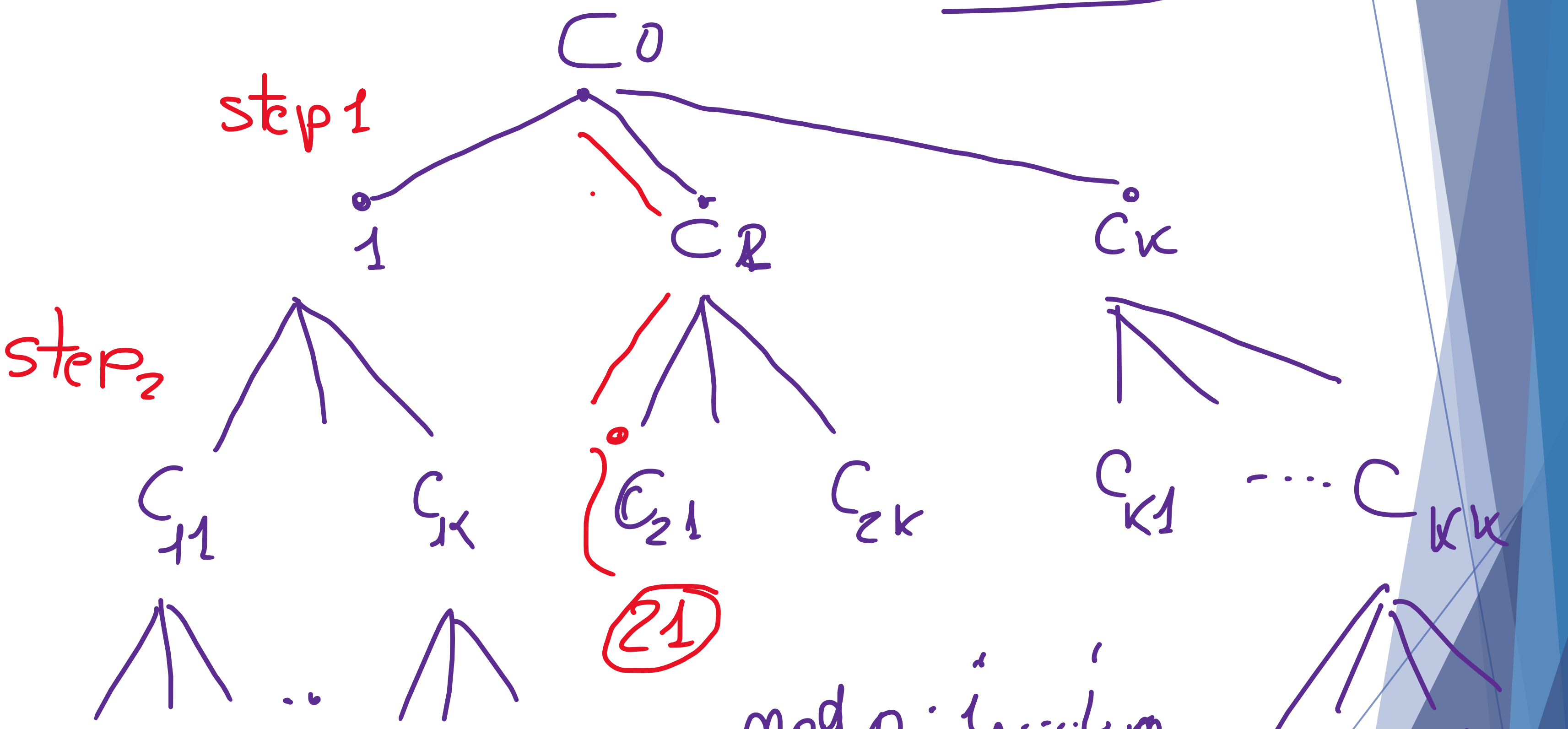
- **Teorema.** **Per ogni MdT M non-deterministica esiste una MdT M' deterministicache riconosce il medesimo linguaggio.**
- **Idea della prova.**
- **Ricordiamo che nel caso non-deterministico, fissata la configurazione iniziale (il contenuto iniziale del nastro) le possibili computazioni posso essere rappresentate da una struttura ad albero:**
  - **I nodi sono le configurazioni**
  - **Gli archi sono i passi di computazione**
  - **I figli di un nodo sono i possibili passi dovuti a scelte non deterministiche**
  - **Esiste una limitazione superiore al numero di scelte non-deterministiche derivabile dalla funzione di transizione.**

Determinizzazione di MdT.

nodi $\{1, \dots k\}^+$

ε - radice

Albero delle computazioni.



nodo: $i_1 \dots i_m$
cammino dalla radice
al nodo

Determinizzazione di MdT.

- Idea della prova.
- Una stringa $\{1,..,k\}^*$ codifica le scelte ad ogni passo di computasione.
- La MdT M' usa tre nastri
- Il primo nastro non viene modificato e contiene la parola iniziale
- Il secondo nastro contiene il codice di un nodo dell'albero delle computazioni
  - Se contiene il nodo $i_1..i_k$ significa che si sta simulando k passi della MdT non-deterministica con le scelte $i_1..i_k$
- Il terzo nastro viene usato per la simulazione enumerata nel secondo nastro.
  - Si parte dalla configurazione iniziale e si fa il numero di passi e le scelte codificate nel secondo nastro

Determinizzazione di MdT.

- La simulazione delle computazioni segue l'ordine lessicografico della codifica dei nodi dell'albero delle computazioni:
  - Corrisponde a una visita in ampiezza dell'albero
  - Tutte le computazioni vengono fatte avanzare a turno di un passo
  - Evita il problema di percorrere completamente computazioni infinite
- Ciclo iterativo
- Si scrive nel secondo nastro la codifica del nodo successivo
- Si simula il comportamento della MdT
  - A partire dal contenuto iniziale nel primo nastro
  - Operando le scelte codificate nel secondo nastro
  - Per un numero di passi massimo pari alla lunghezza della codifica
- Se nella simulazione si arriva a uno stato di accettazione il ciclo termina e si riporta l'accettazione.
- Se tutte le simulazioni lunghe al più k passi portano a rifiuto il ciclo termina e si riporta rifiuto.