Varianti delle macchine di Turing

Esistono molte definizioni alternative (*varianti*) di macchina di Turing.

Tra queste vedremo:

- macchine di Turing a più nastri;
- macchine di Turing non deterministiche.

Tutte le varianti "ragionevoli" della definizione originale hanno la stessa capacità computazionale: riconoscono la stessa classe di linguaggi.

Un primo esempio di variante: stayer

Definizione originale: la funzione di transizione

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

obbliga la testina a spostarsi a destra o a sinistra od ogni passo.

Variante "stayer": permettiamo alla testina anche la possibilità di restare ferma sulla stessa cella del nastro durante una transizione.

La funzione di transizione diventa:

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

dove S indica che la testina rimane ferma.

Potere computazionale di stayer

Domanda: permettendo alla macchina di Turing la possibilità di restare ferma, abbiamo aumentato la sua potenza di calcolo?
Risposta: No!

In generale: diciamo che il potere computazionale di due modelli di calcolo è lo stesso se essi riconoscono la stessa classe di linguaggi.

Possiamo provare che la versione originale della macchina di Turing, che obbliga ad ogni passo la testina a spostarsi, ha lo stesso potere computazionale di stayer.

Ovviamente:

Stayer può facilmente simulare una macchina di Turing convenzionale: basta non usare la possibilità di restare sulla stessa cella del nastro.

Potere computazionale

Resta da provare che una macchina di Turing convenzionale può simulare stayer: il suo potere computazionale è almeno pari al potere computazionale di stayer.

In altre parole: per ogni stayer esiste una macchina di Turing convenzionale che riconosce lo stesso linguaggio

Simuliamo stayer con una MdT convenzionale

Sia M una MdT stayer. Mostriamo che esiste una MdT convenzionale M' che simula M, cioè riconosce lo stesso linguaggio.

Definiamo M' come M, modificando solo la funzione di transizione.

Ogni transizione $\delta(q, a) = (q', a', S)$ può essere simulata da due transizioni in M':

1. $\delta_{M'}(q, a) = (\hat{q}, a', R)$: scrive il simbolo a', muove la testina a destra (R) e va nello stato aggiuntivo $\hat{q}$;
2. $\delta_{M'}(\hat{q}, x) = (q, x, L)$ per ogni $x \in \Gamma_{M'}$: lascia invariato il simbolo letto, muove la testina a sinistra (L) e va nello stato q.

Quindi M e M' riconoscono lo stesso linguaggio.

Simulazione in generale

Abbiamo dimostrato il seguente teorema.

Teorema

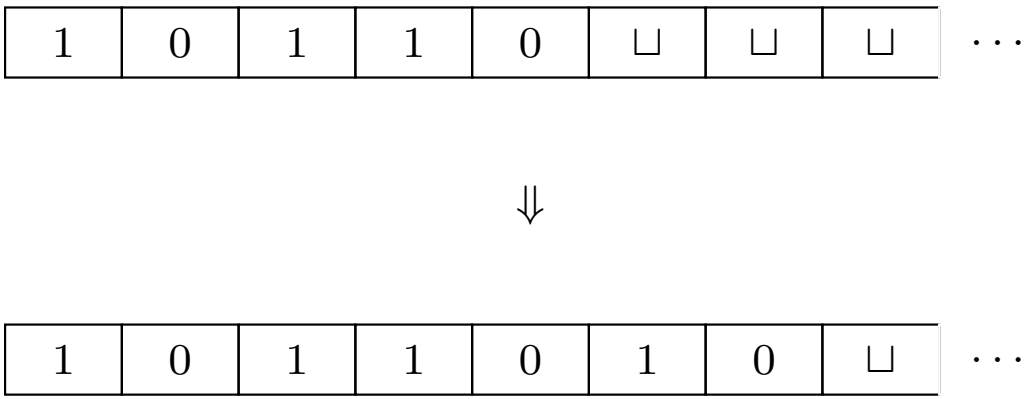
Per ogni macchina di Turing stayer esiste una macchina di Turing equivalente che obbliga la testina a spostarsi in ogni passo.

In generale, per provare che due varianti del modello di macchina di Turing hanno lo stesso potere computazionale si dimostra che è possibile simulare l'una con l'altra.

Implementazione della Memoria

In alcune occasioni avremo bisogno di memorizzare l'informazione letta sul nastro. Possiamo usare gli stati per memorizzare informazione.

Esempio: supponiamo che una macchina di Turing M debba leggere i primi 2 bit del nastro e scriverli alla fine dell'input (al posto dei 2 $\sqcup$ più a sinistra).

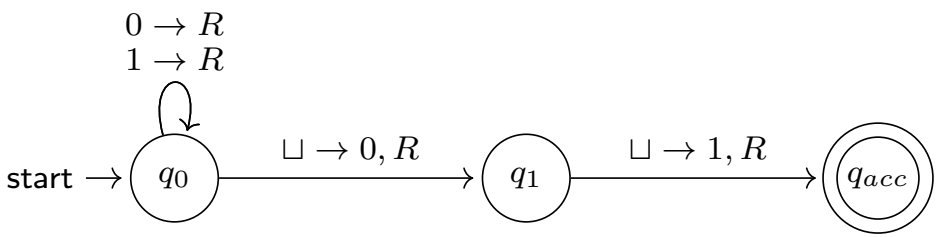


Avremo bisogno di definire una macchina di Turing che sia in grado di "ricordare" i primi due bit letti per poterli poi scrivere correttamente alla fine dell'input.

Costruzione di M

Idea:

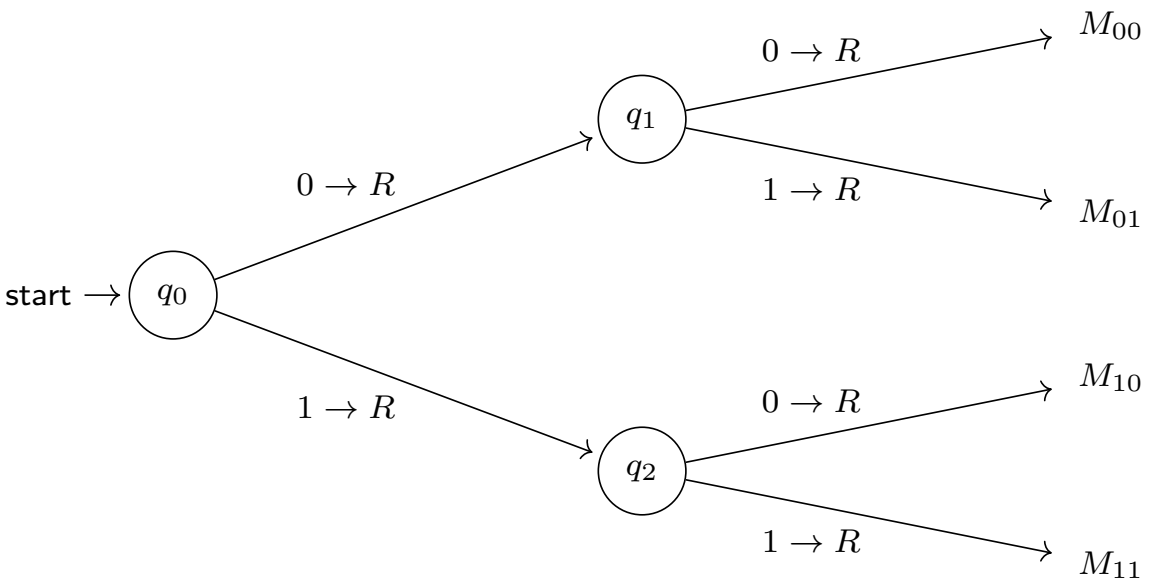
- Costruiamo prima le seguenti macchine di Turing:
  - M_{00} scrive 00 alla fine dell'input;
  - M_{01} scrive 01 alla fine dell'input;
  - M_{10} scrive 10 alla fine dell'input;
  - M_{11} scrive 11 alla fine dell'input.



Macchina di Turing M_{01}: scrive 01 alla fine dell'input.

Costruzione di M

- M legge i primi 2 bit dell'input e si sposta sulla macchina corrispondente ai 2 bit letti (che li scrive alla fine dell'input).



Macchina di Turing M che legge i primi due bit dell'input e li scrive alla fine.

Esempio: se i primi due bit sono 10, la macchina va da q_0 a q_2 e poi da q_2 allo stato iniziale di M_{10}. La computazione segue con M_{10} che si occuperà di scrivere i simboli letti (cioè 10) alla fine dell'input.

Implementazione della Memoria (idea generale)

Idea generale: macchina di Turing M che memorizza una sequenza di k simboli.

- Costruiamo le MdT
$$M_{0\ldots00},\ M_{0\ldots01},\ \ldots,\ M_{1\ldots11},$$
una per ogni possibile sequenza di k simboli.
- M legge i primi k simboli e si sposta sulla macchina che corrisponde alla sequenza letta.

Sono necessari $> |\Sigma|^k$ stati aggiuntivi: almeno uno per ciascuna delle $|\Sigma|^k$ sequenze di k simboli dell'alfabeto Σ, più tutti gli altri...

Macchina di Turing multinastro

Una macchina di Turing multinastro è una normale macchina di Turing che ha $k \geq 1$ nastri.

- Inizialmente l'input compare sul primo nastro e gli altri nastri sono vuoti.
- La funzione di transizione

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

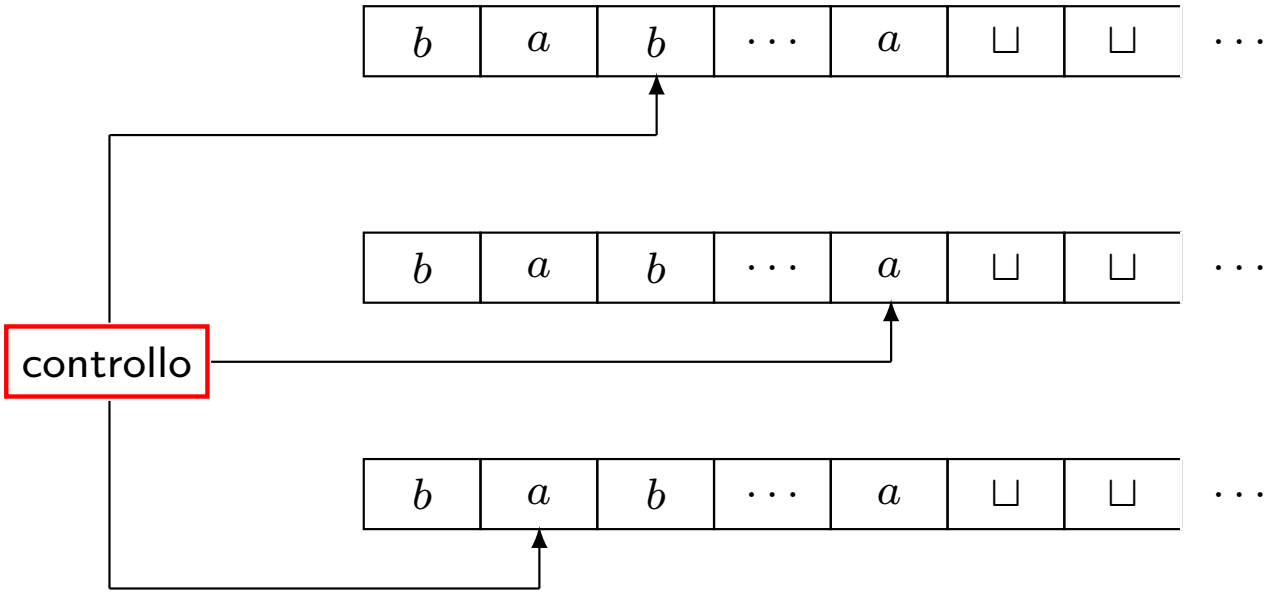
  consente alle k testine di muoversi in modo indipendente.

Una transizione, oltre agli stati di partenza e destinazione, dovrà specificare:

- k simboli letti;
- k simboli da scrivere;
- k movimenti delle k testine.

Macchina di Turing multinastro

Macchina di Turing con $k = 3$ nastri.



Una generica transizione assume la seguente forma:

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, D_1, \ldots, D_k),$$

dove $D_i \in \{L, R, S\}$ per $i = 1, \ldots, k$.

Macchina di Turing multinastro

Sia M una macchina di Turing a k nastri. La transizione

$$\delta(q_i, a_1, \ldots, a_k) = (q_i', b_1, \ldots, b_k, D_1, \ldots, D_k)$$

indica che se M

- si trova nello stato q_i
- la testina del j-esimo nastro legge il simbolo a_j, per $j = 1, \ldots, k$,

allora

- M va nello stato q_i'
- la testina del nastro j scrive b_j e si muove nella direzione $D_j \in \{L, R, S\}$, per $j = 1, \ldots, k$.

Esempio: MdT a 2 nastri per $\{0^n 1^n | \ w \in \{0,1\}^*\}$

Vediamo come progettare una macchina di Turing a due nastri per riconoscere il linguaggio $\{0^n 1^n | \ w \in \{0,1\}^*\}$.

Idea.

1. Scorriamo il primo nastro verso destra fino al primo `1`: per ogni `0`, scriviamo un `1` sul secondo nastro;
2. Scorriamo contemporaneamente il primo nastro verso destra e il secondo nastro verso sinistra: se ad un passo i simboli letti non sono uguali, terminiamo nello stato di rifiuto;
3. Se leggiamo $\sqcup$ su entrambi i nastri, terminiamo nello stato di accettazione.

stato	**simboli**	**stato**	**simboli**	**movimenti**
q0	$(0, \sqcup)$	q0	$(\sqcup, 1)$	(R, R)
q0	$(1, \sqcup)$	q1	$(1, \sqcup)$	(S, L)
q1	$(1, 1)$	q1	$(\sqcup, \sqcup)$	(R, L)
q1	$(\sqcup, \sqcup)$	q2	$(\sqcup, \sqcup)$	(S, S)

MdT mononastro e MdT multinastro

Le macchine di Turing multinastro sembrerebbero più potenti delle macchine di Turing a nastro singolo.

In realtà possiamo mostrare che le due varianti sono equivalenti.

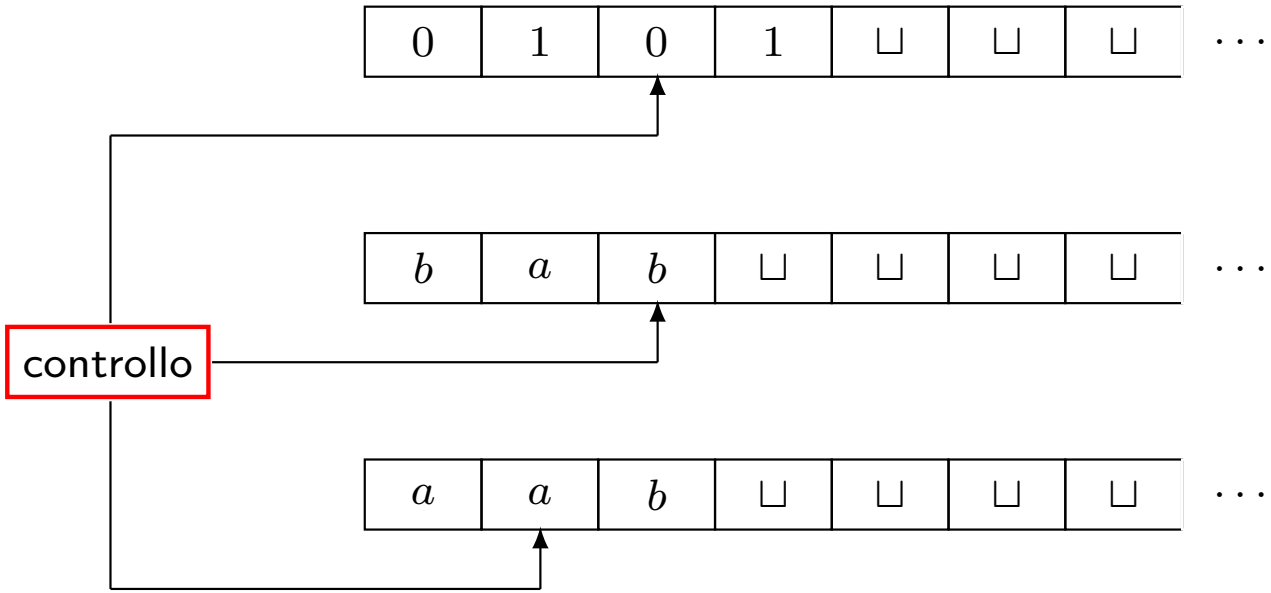
Teorema

Per ogni macchina di Turing multinastro esiste una macchina di Turing equivalente a nastro singolo.

Uno dei due versi è ovvio: una MdT multinastro è anche una MdT a nastro singolo. Tutto ciò che può fare una MdT mononastro lo può fare anche una MdT multinastro (basta usare un solo nastro).

MdT mononastro e MdT multinastro

Vale anche il verso opposto? Possiamo simulare una qualunque macchina di Turing multinastro M con una macchina di Turing S a nastro singolo?

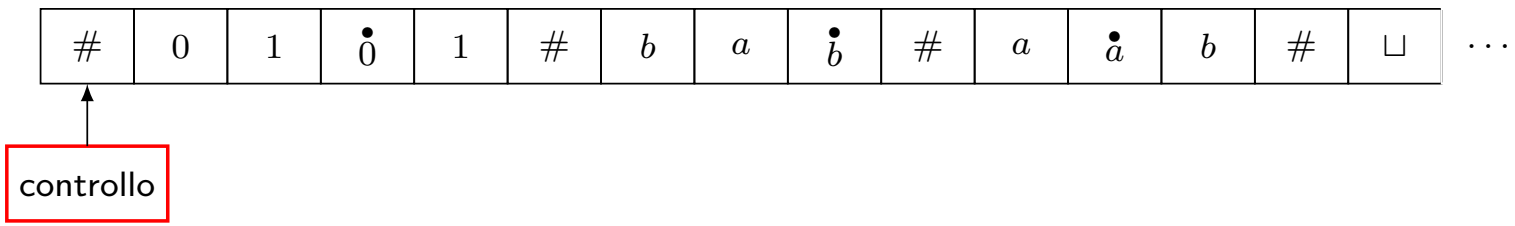


Dobbiamo costruire una macchina S che simuli l'effetto di k nastri memorizzando tutte le loro informazioni sul singolo nastro.

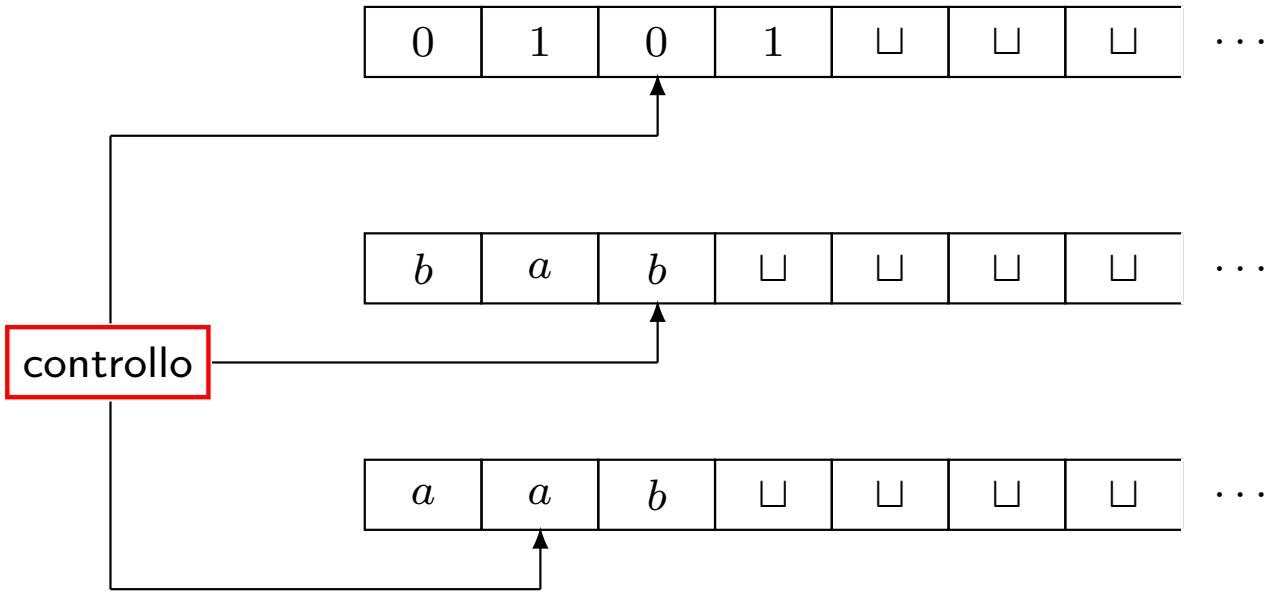
MdT mononastro e MdT multinastro

Tutte le informazioni contenute nei k nastri devono essere codificate su un singolo nastro.

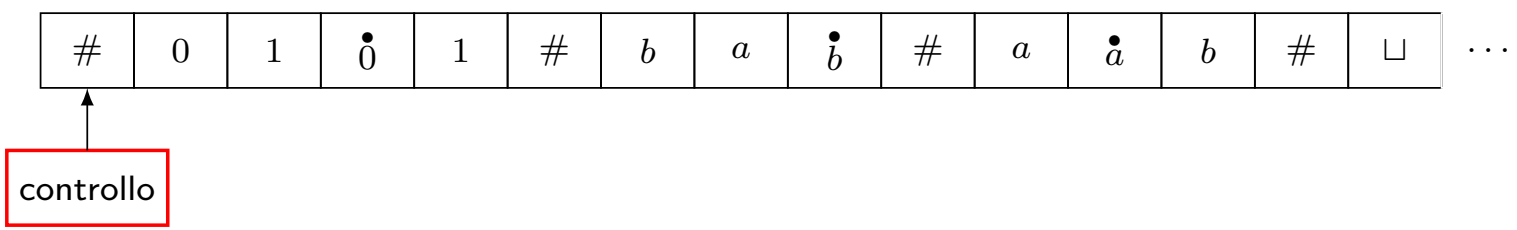
1. Utilizziamo il nuovo simbolo $\#$ per separare i contenuti dei diversi nastri.
2. Una configurazione su k nastri corrisponderà ad una configurazione di k blocchi delimitati dal simbolo $\#$: il contenuto dell'i-esimo nastro è rappresentato dall'i-esimo blocco.
3. Ogni blocco ha lunghezza variabile che dipende dal contenuto del nastro corrispondente.
4. Per indicare la posizione della testina su ciascuno dei nastri, includiamo un nuovo simbolo $\dot{\gamma}$ per ogni simbolo γ dell'alfabeto.
5. Se la testina dell'i-esimo nastro legge il simbolo a sulla j-esima cella, allora l'i-esimo blocco di S avrà il simbolo $\dot{a}$ sulla sua j-esima posizione.



MdT mononastro e MdT multinastro



Macchina M. Alfabeto del nastro: $\{0, 1, a, b, \sqcup\}$.



Macchina S. Alfabeto del nastro: $\{0, 1, a, b, \sqcup, \#, \dot{0}, \dot{1}, \dot{a}, \dot{b}, \dot{\sqcup}\}$.

Simulazione di M multinastro con S a nastro singolo (idea)

Supponiamo che a M venga assegnato l'input $w = w_1 \cdots w_n$.

1. S rappresenta la configurazione iniziale di M formattando il suo unico nastro nel modo seguente:
$$\# \dot{w}_1 \; w_2 \; \cdots \; w_n \; \# \dot{\sqcup} \; \# \dot{\sqcup} \; \# \; \cdots \; \#.$$
2. Per simulare una singola mossa di M, S scansiona il nastro dal primo $\#$ fino al $(k+1)$-esimo per leggere i k simboli puntati dalle k testine. Poi S fa tornare la testina al primo blocco e fa un secondo passaggio per aggiornare tutti i blocchi in accordo alla funzione di transizione di M.
3. Se in un determinato passo, M sposta la testina di un nastro sul primo $\sqcup$ a destra dell'ultimo simbolo dell'input, S scriverà un simbolo $\sqcup$ alla fine del blocco corrispondente, e sposterà, di una unità a destra, il contenuto dei successivi blocchi:
$$\cdots \# 0\,1\,0 \# 0\,1 \cdots \quad \Rightarrow \quad \cdots \# 0\,1\,0 \sqcup \# 0\,1 \cdots$$

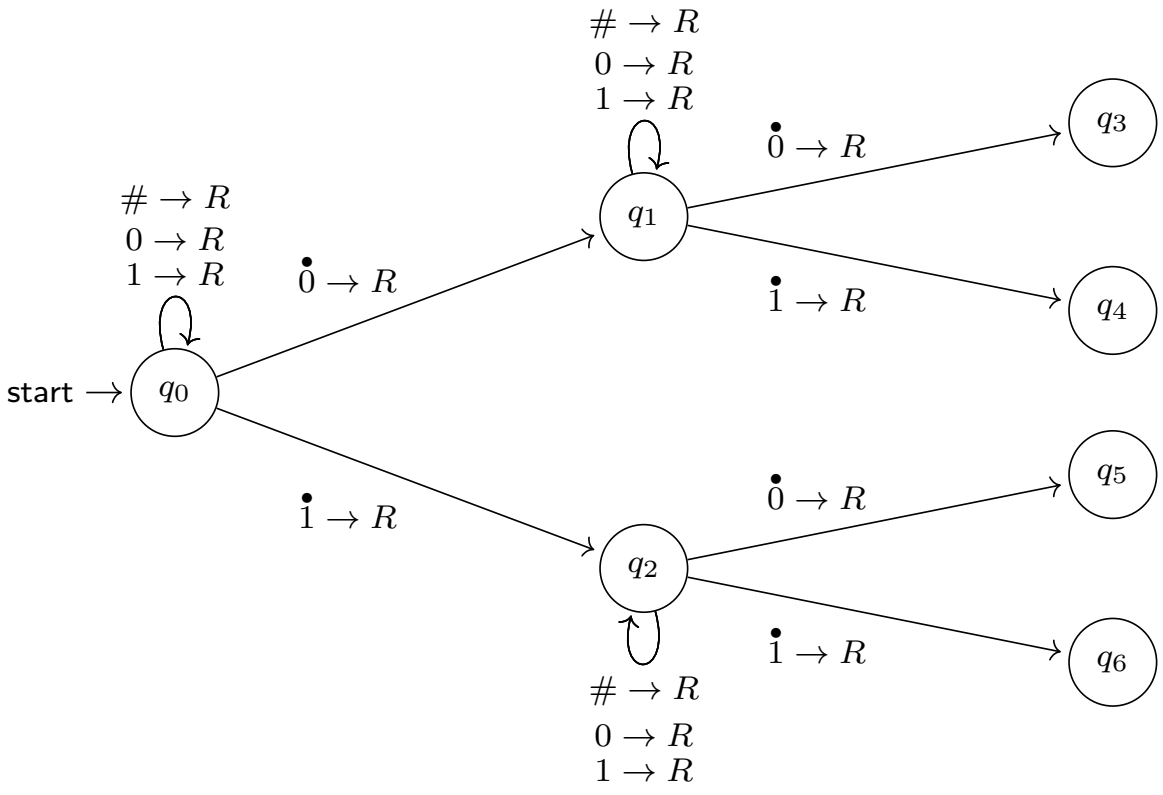
Simulazione di M multinastro con S a nastro singolo (idea)

Supponiamo di voler simulare le seguenti transizioni di una macchina a 2 nastri con una macchina mononastro.

stato	**simboli**	**stato**	**simboli**	**movimenti**
q	$(0,0)$	q'	$(0,0)$	(R,R)
q	$(0,1)$	q'	$(1,0)$	(R,L)
q	$(1,0)$	q'	$(0,0)$	(L,L)
q	$(1,1)$	q'	$(0,1)$	(L,R)

Simulazione di M multinastro con S a nastro singolo (idea)

Idea del primo passaggio: S "memorizza" il contenuto dei blocchi.



Se S arriva in q_3, vuol dire che M ha letto 0 sul primo nastro e 0 sul secondo.
Se S arriva in q_4, vuol dire che M ha letto 0 sul primo nastro e 1 sul secondo.
E così via.

Simulazione di M multinastro con S a nastro singolo (idea)

Lo stato q_4 corrisponde alla lettura di 0 sul primo nastro e 1 sul secondo. In tal caso, la macchina a due nastri ha la seguente transizione:

$$\delta(q, 0, 1) = (q', 1, 0, R, L).$$

Che indica il seguente comportamento.

Se leggi 0 sul primo nastro e 1 sul secondo:

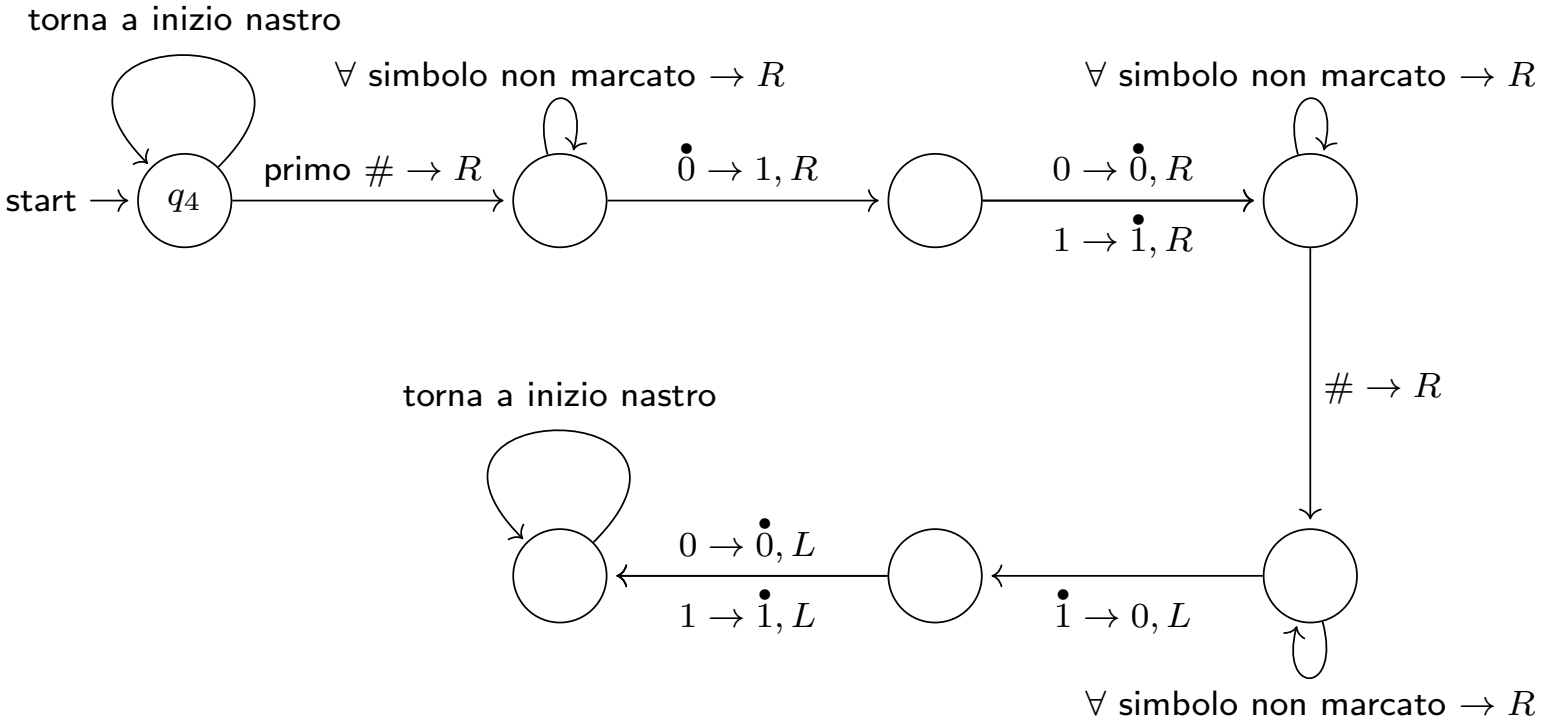
- Scrivi 1 sul primo nastro e sposta a destra la sua testina.
- Scrivi 0 sul secondo nastro e sposta a sinistra la sua testina.
- Passa allo stato q'.

Per simulare tale comportamento su S, da q_4 la testina viene fatta ritornare all'inizio del nastro. Poi si prosegue con il secondo passaggio che mira ad aggiornare il contenuto del nastro:

- $\dot{0}$ del primo nastro va sostituito con 1,
- $\dot{1}$ sul secondo nastro va sostituito con 0 e poi vanno aggiornate le posizioni delle testine virtuali.

Simulazione di M multinastro con S a nastro singolo (idea)

Idea del secondo passaggio per la transizione $\delta(q, 0, 1) = (q', 1, 0, R, L)$.



$$\#\,0\,0\,1\,\dot{0}\,1\,0\,\#\,1\,1\,0\,\dot{1}\,1\,\#\,\sqcup \quad \Longrightarrow \quad \#\,0\,0\,1\,1\,\dot{1}\,0\,\#\,1\,1\,\dot{0}\,0\,1\,\#\,\sqcup$$

Simulazione di M multinastro con S a nastro singolo (idea)

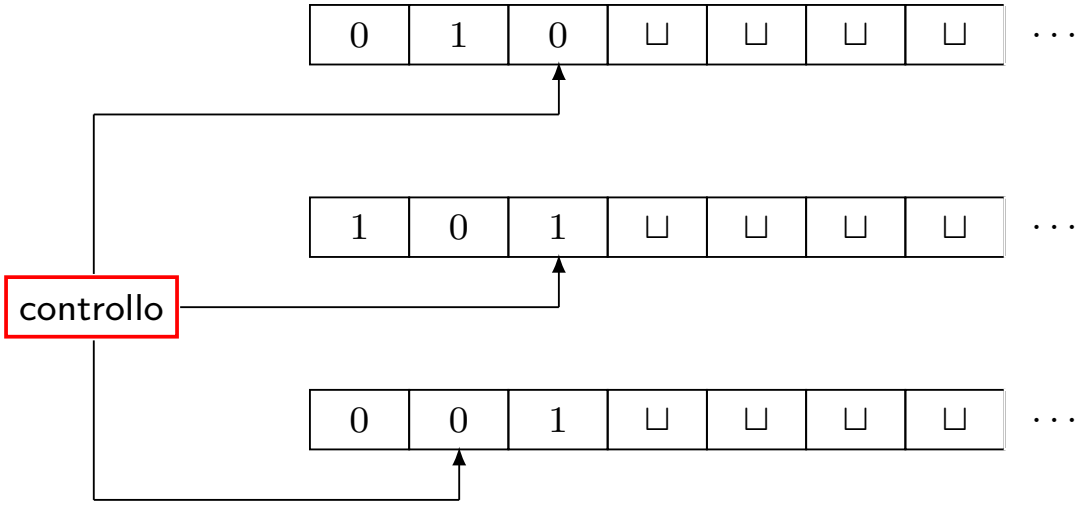
Ricapitolando.

Per ogni mossa del tipo $\delta(q, a_1, ..., a_k) = (s, b_1, ..., b_k, D_1, ..., D_k)$

- Primo passaggio: S scorre il nastro verso destra, fino al primo $\sqcup$ "memorizzando" nello stato in cui approda, i simboli marcati sui singoli blocchi.
- La testina si riposiziona all'inizio del nastro.
- Secondo passaggio: il nastro viene scandito di nuovo eseguendo su ogni blocco (nastro di M) le azioni di scrittura e spostamento che simulano quelle delle testine di M:
  1. Sovrascrivi tutti i simboli marcati.
  2. Sposta il marcatore alla nuova posizione corrispondente alla testina di M.

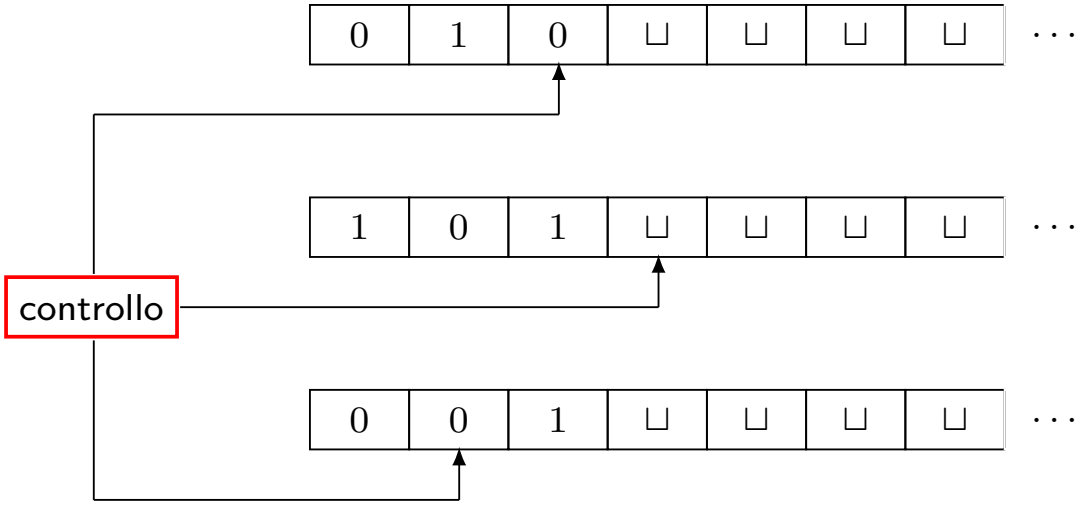
Simulazione di M multinastro con S a nastro singolo (idea)

Caso particolare: la testina del j-esimo nastro si sposta sul primo $\sqcup$ dopo la fine dell'input.



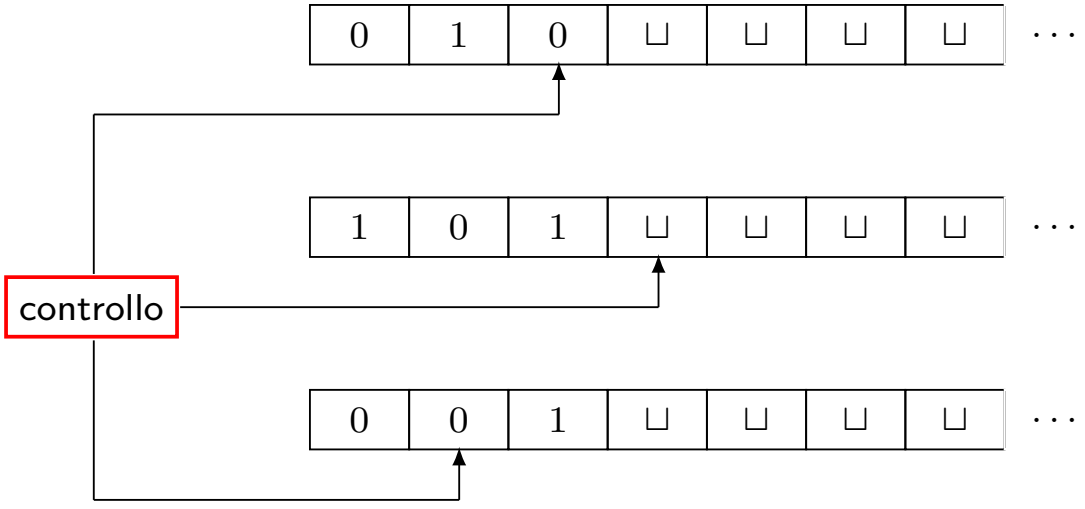
Simulazione di M multinastro con S a nastro singolo (idea)

Caso particolare: la testina del j-esimo nastro si sposta sul primo $\sqcup$ dopo la fine dell'input.



Simulazione di M multinastro con S a nastro singolo (idea)

Caso particolare: la testina del j-esimo nastro si sposta sul primo $\sqcup$ dopo la fine dell'input.

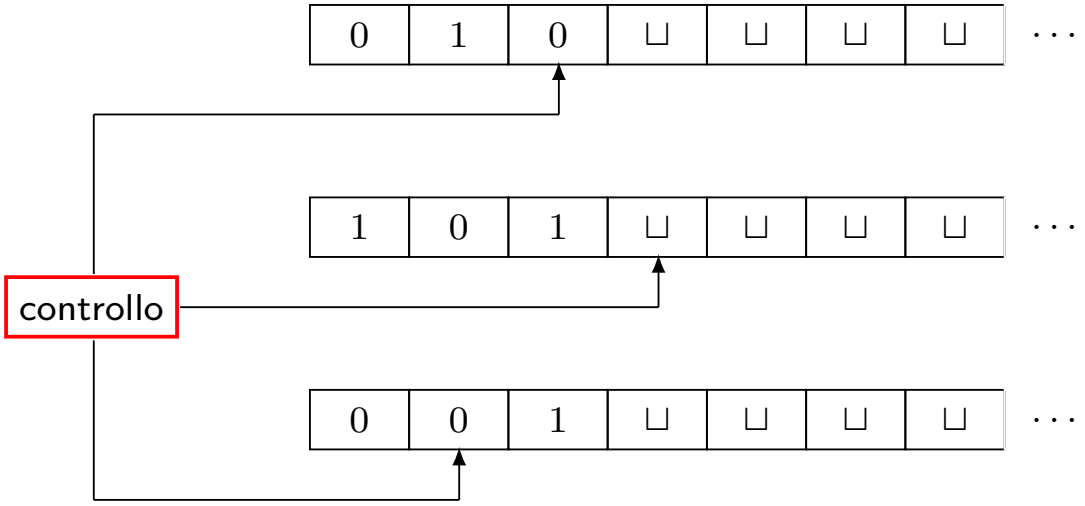


1. Il delimitatore destro del j-esimo blocco viene sostituito da $\sqcup$, e il contenuto dei successivi blocchi viene spostato di una cella a destra:

$$\#\,0\,1\,\dot{0}\,\#\,1\,0\,\dot{1}\,\#\,0\,\dot{0}\,1\,\#\,\sqcup \quad\Longrightarrow\quad \#\,0\,1\,\dot{0}\,\#\,1\,0\,\dot{1}\,\sqcup\,\#\,0\,\dot{0}\,1\,\#\,\sqcup$$

Simulazione di M multinastro con S a nastro singolo (idea)

Caso particolare: la testina del j-esimo nastro si sposta sul primo $\sqcup$ dopo la fine dell'input.



1. Il delimitatore destro del j-esimo blocco viene sostituito da $\sqcup$, e il contenuto dei successivi blocchi viene spostato di una cella a destra:

$$\#\,0\,1\,\dot{0}\,\#\,1\,0\,\dot{1}\,\#\,0\,\dot{0}\,1\,\#\,\sqcup \quad\Longrightarrow\quad \#\,0\,1\,\dot{0}\,\#\,1\,0\,\dot{1}\,\sqcup\,\#\,0\,\dot{0}\,1\,\#\,\sqcup$$

2. Il marcatore viene tolto dalla sua posizione e posto sul $\sqcup$:

$$\#\,0\,1\,\dot{0}\,\#\,1\,0\,\dot{1}\,\sqcup\,\#\,0\,\dot{0}\,1\,\#\,\sqcup \quad\Longrightarrow\quad \#\,0\,1\,\dot{0}\,\#\,1\,0\,1\,\dot{\sqcup}\,\#\,0\,\dot{0}\,1\,\#\,\sqcup$$

Simulazione di M multinastro con S a nastro singolo (idea)

Per ogni istruzione della MdT multinastro M, la MdT mononastro S opera come segue.

1. Scorre i k blocchi e "raccoglie" informazioni sui k simboli letti (essi corrispondono ai k simboli che la multinastro legge sui rispettivi k nastri).
2. Riporta la testina all'inizio del nastro.
3. Simula la transizione di M, scorrendo nuovamente i k blocchi e applicando su ciascuno scrittura e spostamento come indicato dalla funzione di transizione di M.
4. Entra nello stato che "ricorda" il nuovo stato di M e riposiziona la testina all'inizio del nastro.

S usa molti più stati e mosse di M, ma il suo potere computazionale è lo stesso.

Macchina di Turing non deterministica

Una macchina di Turing **non deterministica** (NMdT) è una macchina di Turing avente funzione di transizione:

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\}).$$

invece di

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

Come computa una NMdT?

Analogamente a quanto visto per gli NFA:

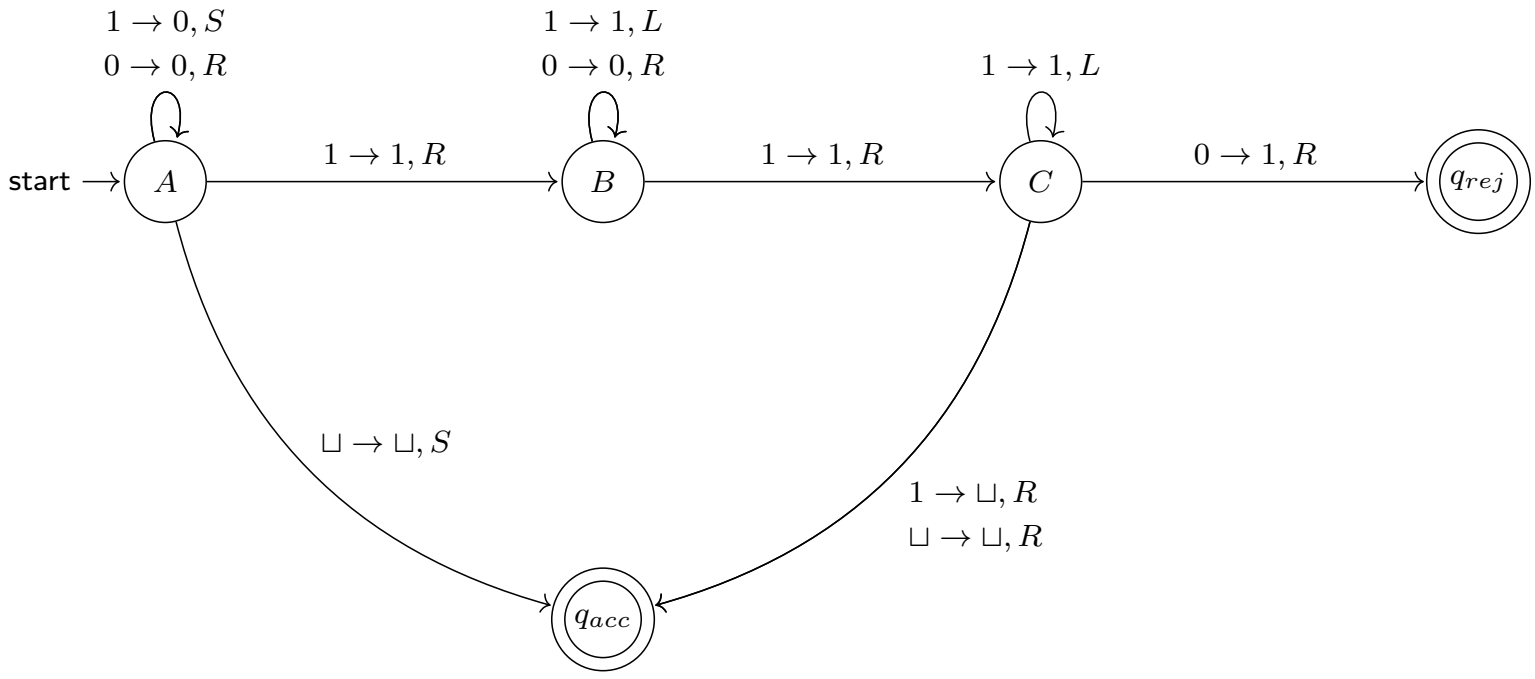
Se una NMdT è in uno stato e, leggendo un simbolo a dell'alfabeto, si trova davanti a più scelte:

- si divide in più copie di se stessa: una copia per ciascuna scelta;
- ogni copia segue la computazione in parallelo indipendentemente dalle altre.

- Come nel caso di NFA, la computazione di una NMdT si rappresenta con un albero i cui nodi sono configurazioni.
- Ogni ramo è una sequenza di configurazioni che corrisponde ad una delle possibili scelte della macchina.
- Se qualche ramo della computazione porta ad una configurazione di accettazione allora la macchina accetta l'input.

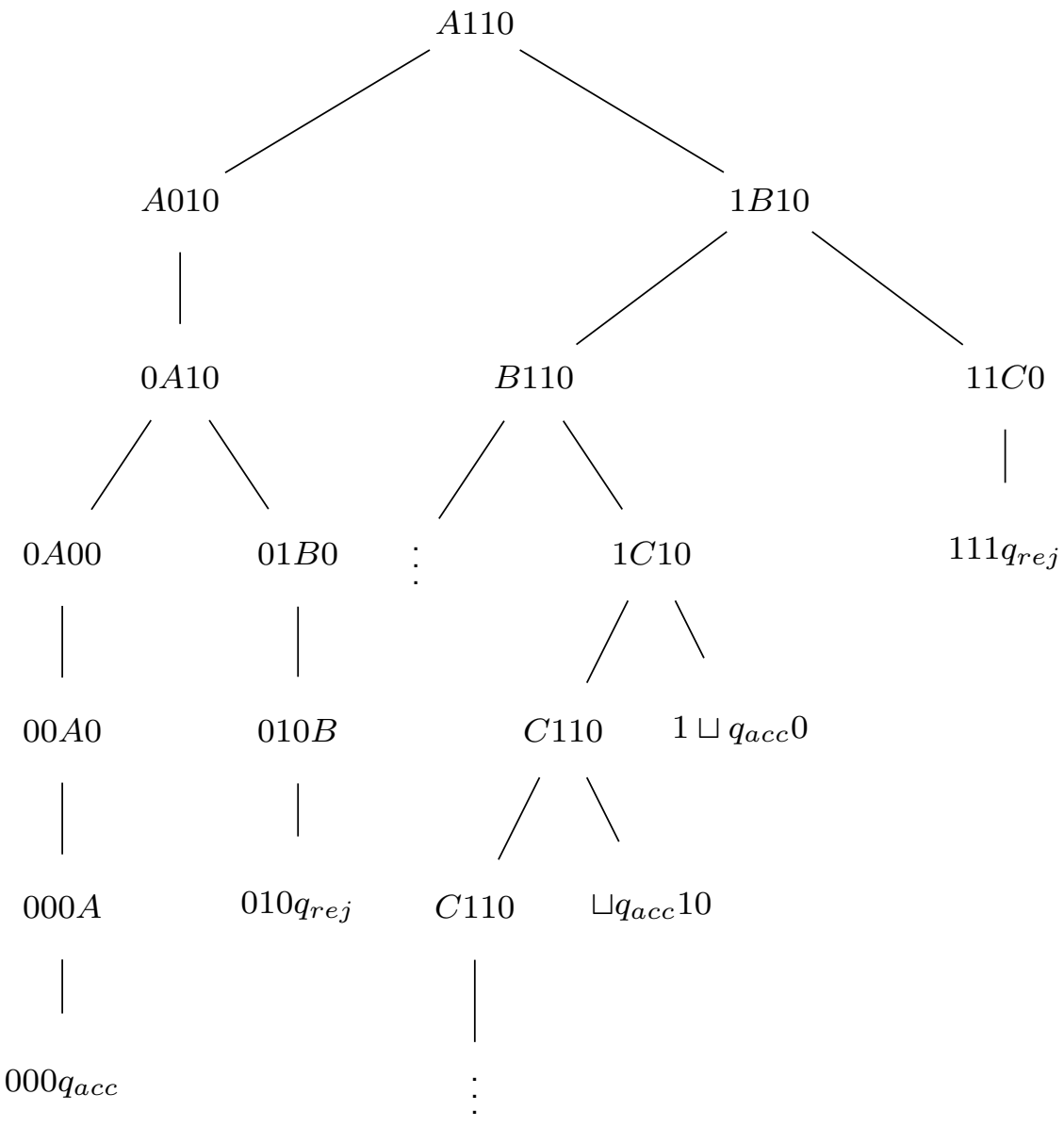
Esempio

Si consideri la seguente MdT non deterministica. Le transizioni mancanti conducono nello stato di rifiuto, lasciando inalterato l'input e la posizione della testina.



Determiniamo l'albero della computazione sull'input 110.

Esempio


$A110$
$A010$
$1B10$
$0A10$
$B110$
$11C0$
$0A00$
$01B0$
$1C10$
$111q_{rej}$
$00A0$
$010B$
$C110$
$1 \sqcup q_{acc}0$
$000A$
$010q_{rej}$
$C110$
$\sqcup q_{acc}10$
$000q_{acc}$

Equivalenza tra determinismo e non determinismo

Le macchine di Turing non deterministiche sembrerebbero più potenti, ma si può dimostrare che il non determinismo non influisce sulla potenza di calcolo.

Teorema

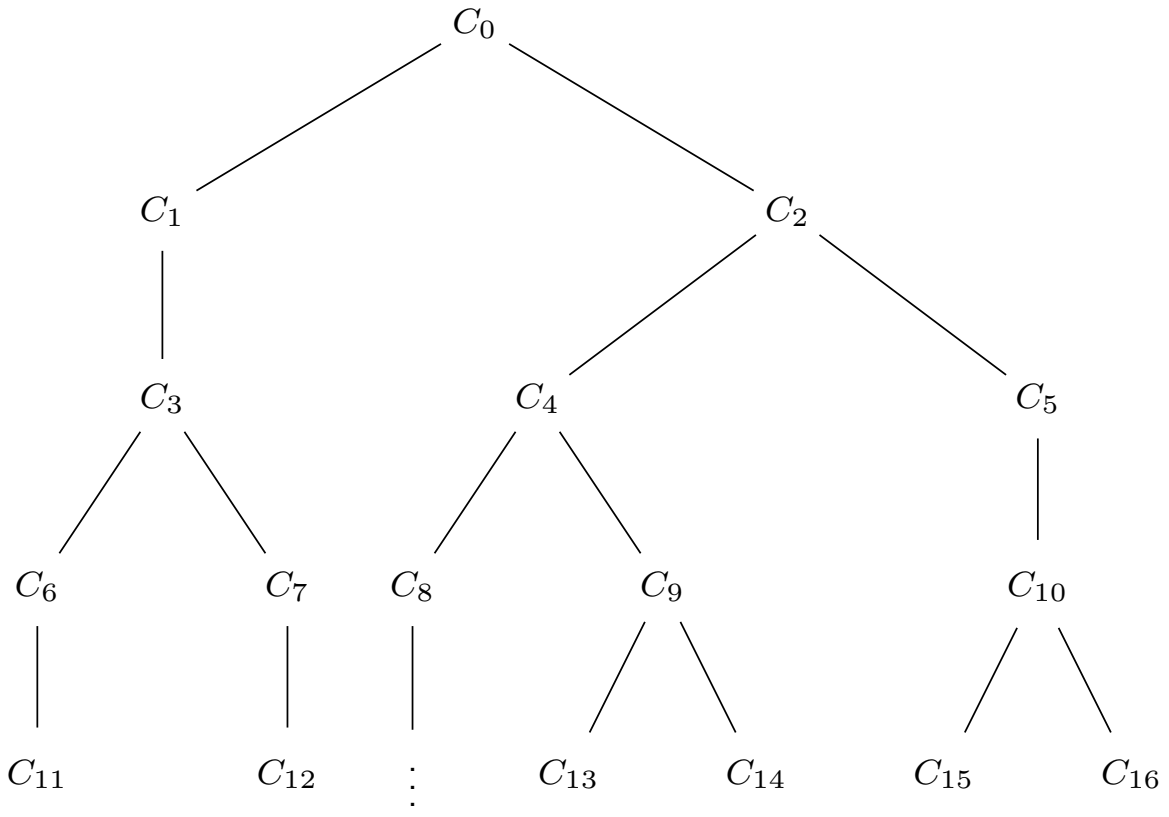
Per ogni macchina di Turing non deterministica, esiste una equivalente macchina di Turing deterministica.

Uno dei due versi dell'asserto è ovvio: una macchina di Turing deterministica è anche (caso particolare) una macchina di Turing non deterministica.

Altro verso: possiamo anche simulare una macchina di Turing non deterministica N con una deterministica D?

Equivalenza tra determinismo e non determinismo

Abbiamo visto che la computazione di N su un input w è rappresentabile con un albero avente per radice la configurazione iniziale C_0.



Ogni nodo dell'albero è una configurazione di N raggiungibile da C_0. Per simulare N, la macchina deterministica D dovrà esplorare l'albero alla ricerca di una configurazione di accettazione.

Equivalenza tra determinismo e non determinismo

Come esploriamo l'albero della computazione?

Bisogna stare attenti perchè alcune diramazioni dell'albero della computazione hanno lunghezza infinita. La macchina non deterministica può accettare una stringa anche se una delle diramazioni è di lunghezza infinita (loop). L'importante è che **almeno una** di esse termini nello stato di accettazione.

Se la macchina deterministica esegue la simulazione e segue una diramazione infinita, va in loop e non si potrà mai accorgere di una eventuale altra diramazione che raggiungerebbe lo stato di accettazione.

Equivalenza tra determinismo e non determinismo

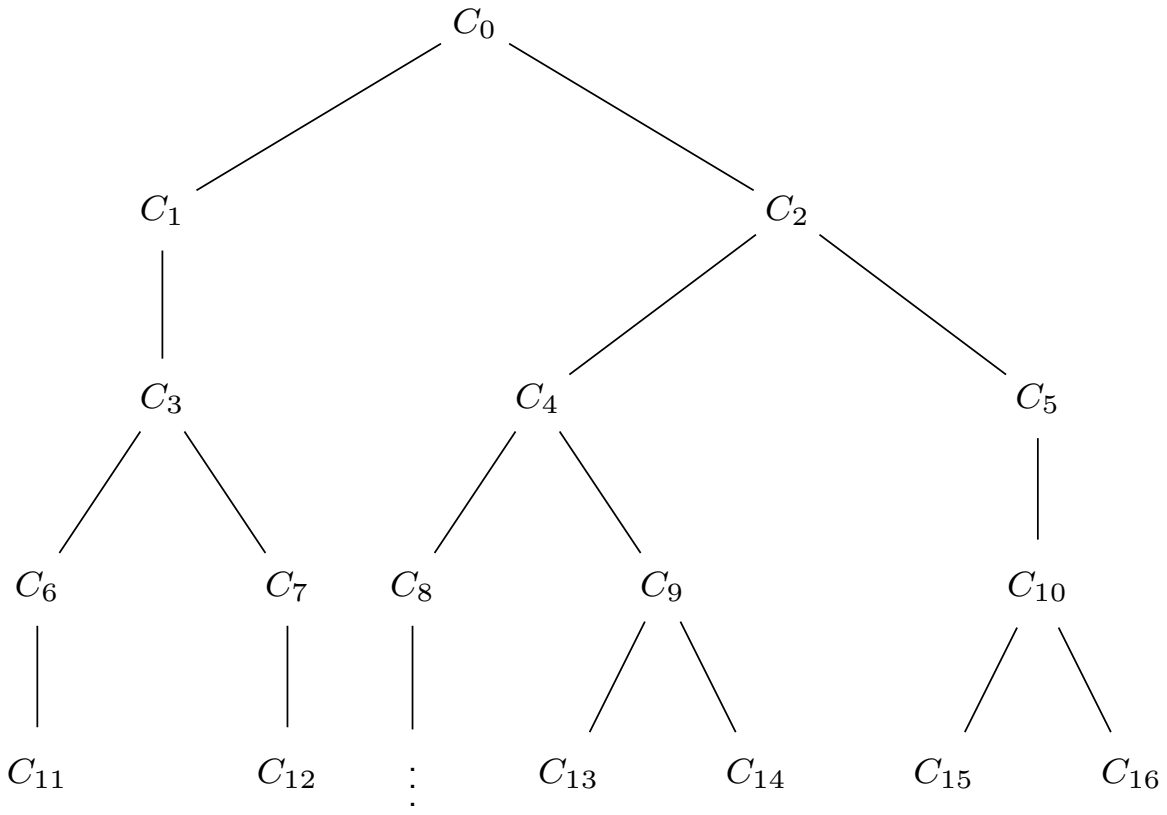
Come esploriamo l'albero della computazione?

Bisogna stare attenti perchè alcune diramazioni dell'albero della computazione hanno lunghezza infinita. La macchina non deterministica può accettare una stringa anche se una delle diramazioni è di lunghezza infinita (loop). L'importante è che **almeno una** di esse termini nello stato di accettazione.

Se la macchina deterministica esegue la simulazione e segue una diramazione infinita, va in loop e non si potrà mai accorgere di una eventuale altra diramazione che raggiungerebbe lo stato di accettazione.

Idea: eseguiamo la simulazione usando una visita BFS dell'albero delle computazioni.

Da NMdT a MdT



1. Esegui il primo passo di ogni computazione. Se almeno una accetta allora accetta.
2. Esegui il secondo passo di ogni computazione. Se almeno una accetta allora accetta.

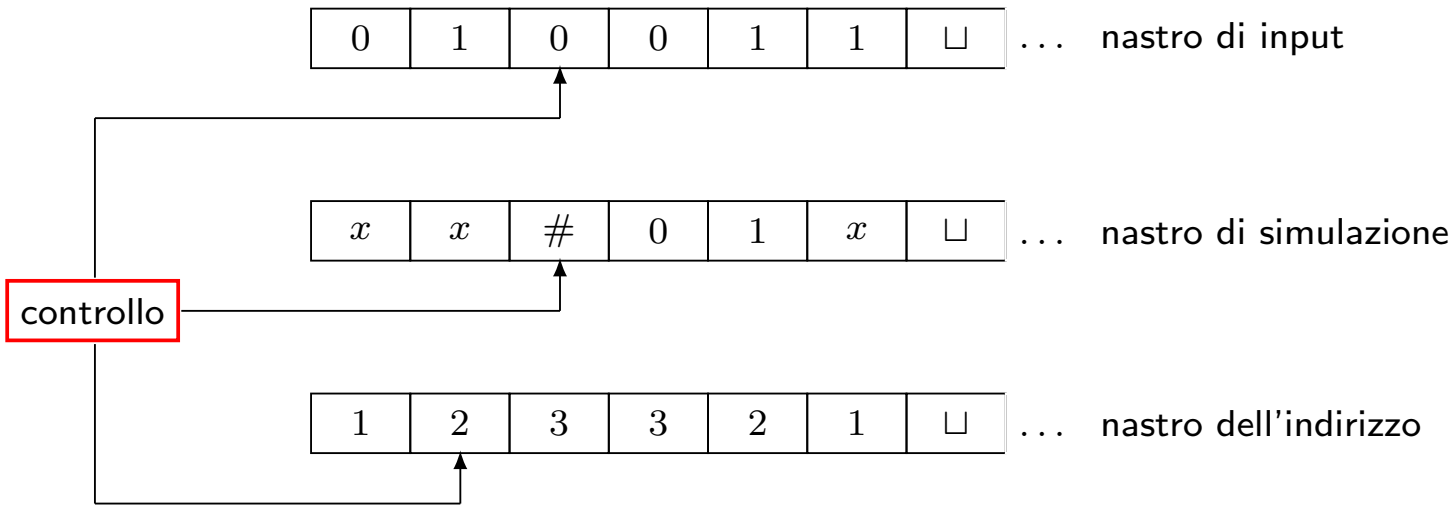
   ...

i. Esegui il passo i-mo di ogni computazione. Se almeno una accetta allora accetta.

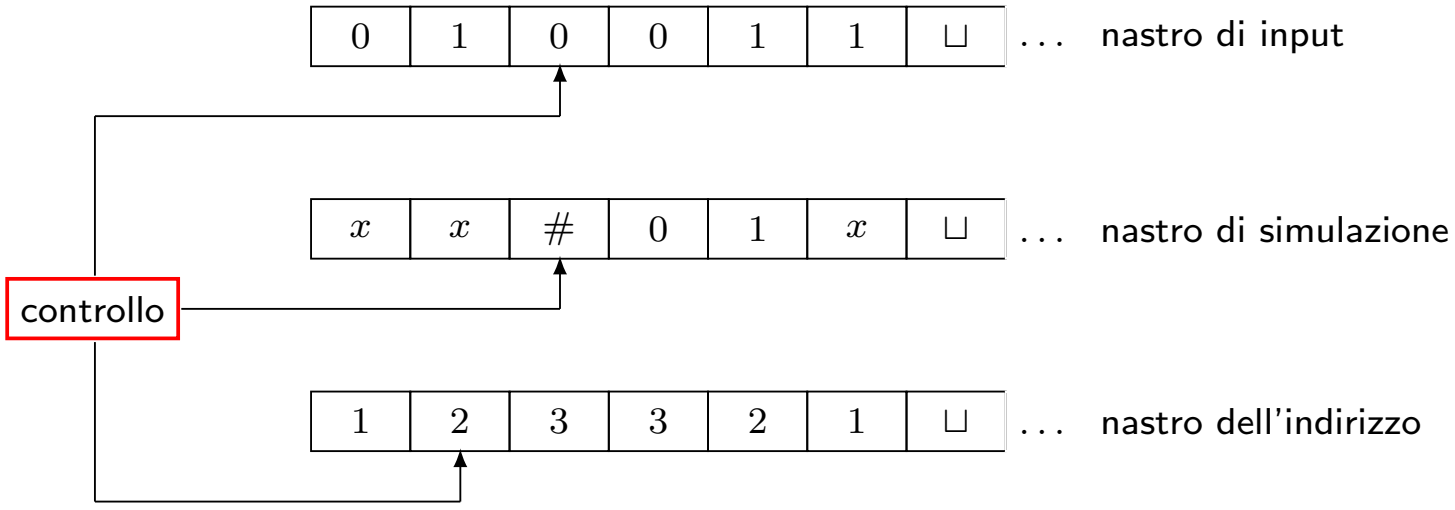
   ...

Da NMdT a MdT

Possiamo simulare la macchina non deterministica N con una macchina deterministica D a tre nastri (equivalente ad una mononastro).



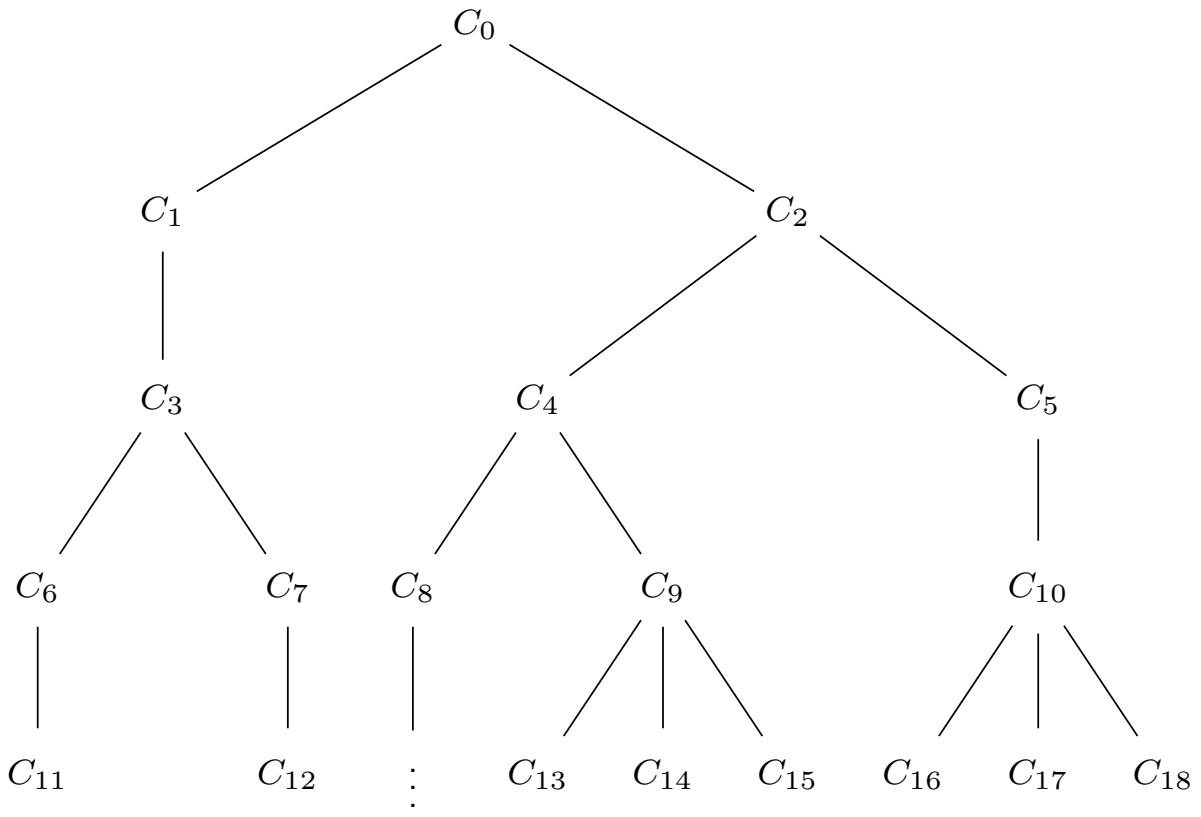
Da NMdT a MdT



- Il nastro 1 contiene sempre la stringa input e non viene mai modificato.
- Il nastro 2 contiene una copia del contenuto del nastro di N corrispondente ad un ramo della sua computazione non deterministica.
- Il nastro 3 serve a tener traccia della posizione di D nell'albero della computazione non deterministica di N.

Da NMdT a MdT

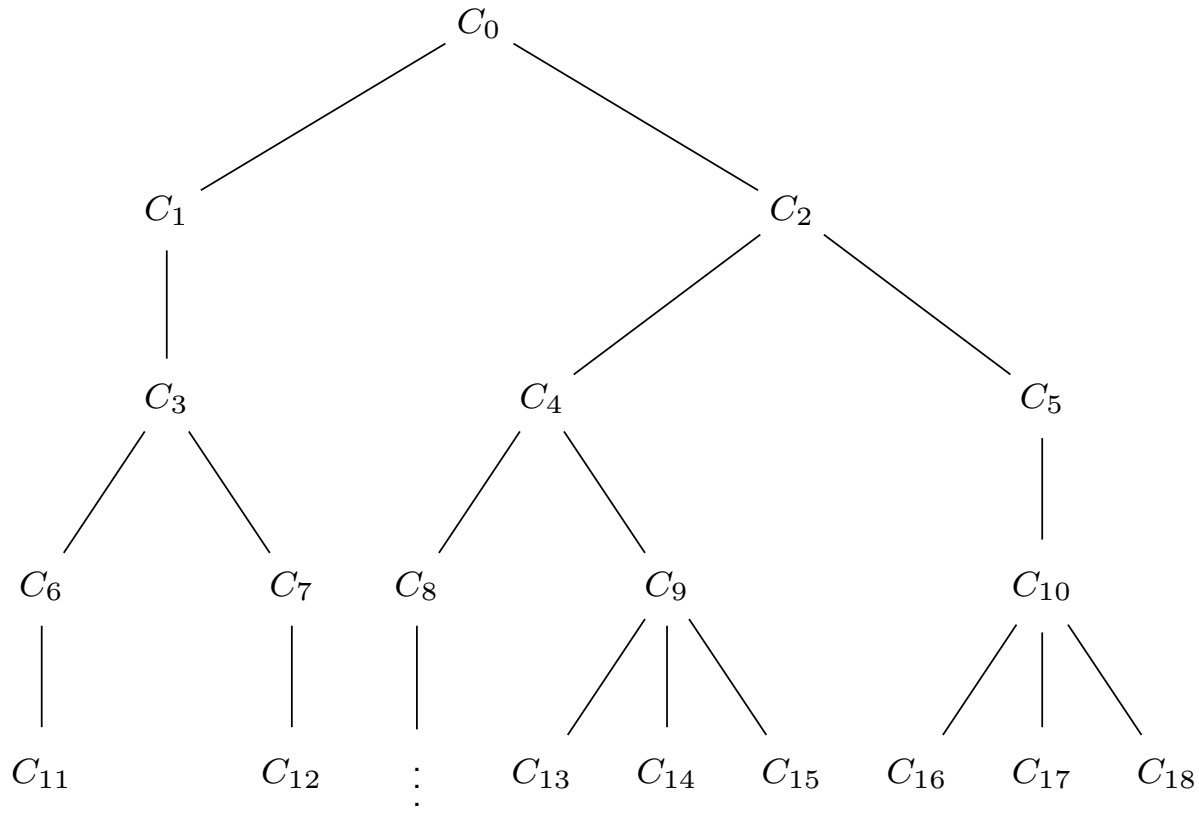
Partiamo dalla rappresentazione dei dati sul nastro 3.



Ogni nodo dell'albero può avere al massimo b figli, dove b è il massimo numero di scelte non deterministiche di N (si vede dalla funzione di transizione di N). Nel nostro esempio $b = 3$.

Da NMdT a MdT

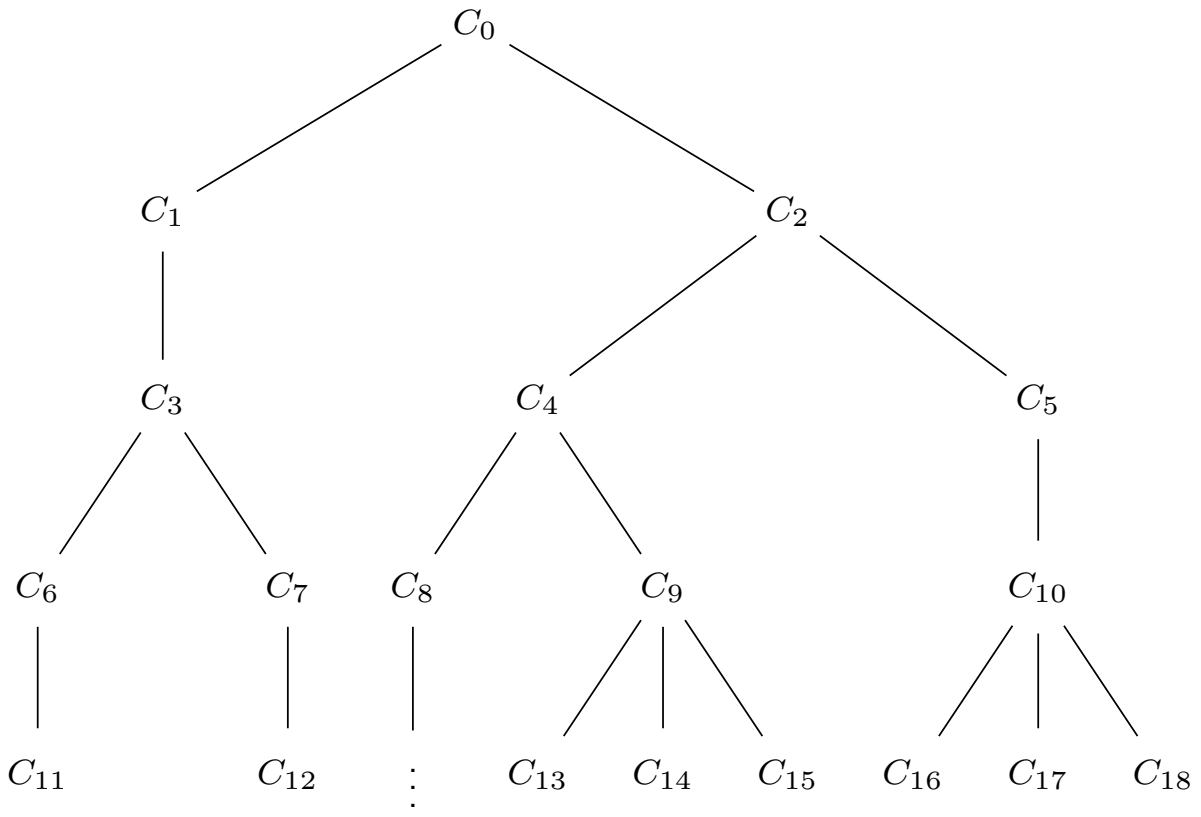
Partiamo dalla rappresentazione dei dati sul nastro 3.



Ad ogni nodo assegniamo un indirizzo che è una stringa sull'alfabeto $\Gamma_b = \{1, 2, \ldots, b\}$.

Da NMdT a MdT

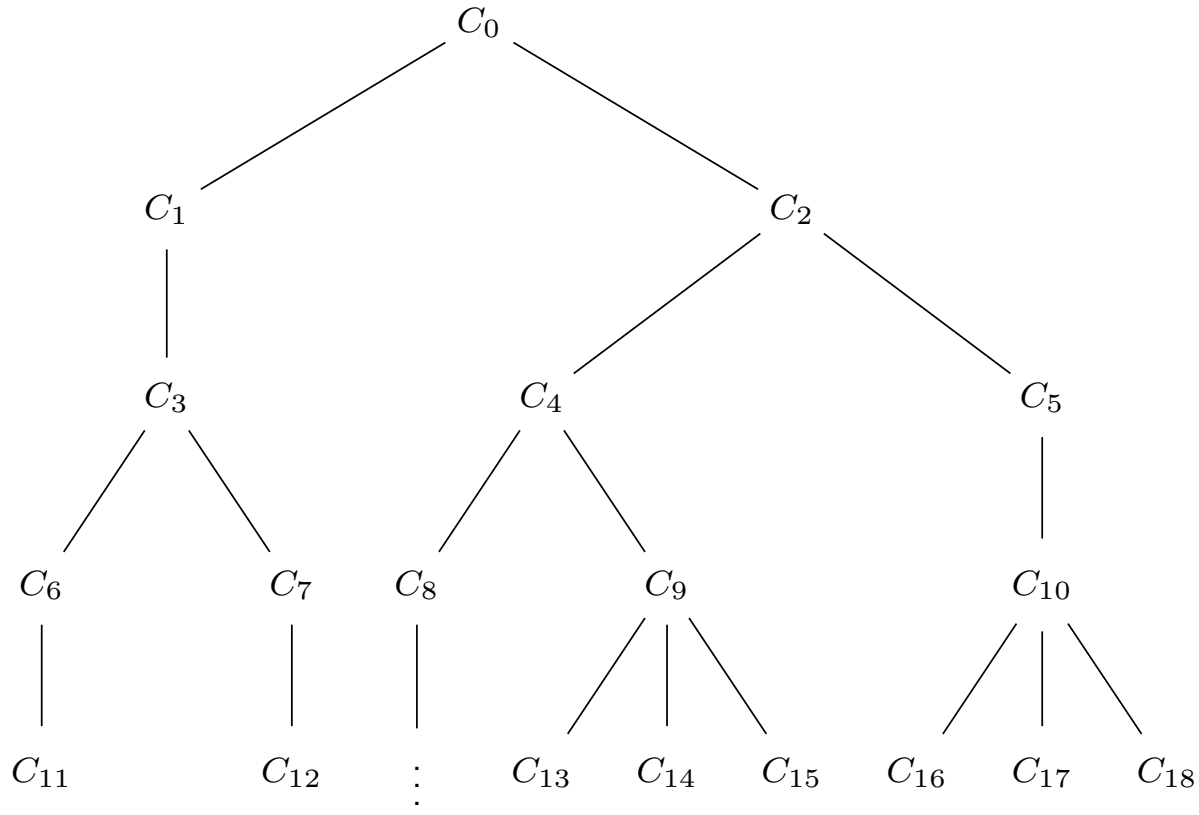
Partiamo dalla rappresentazione dei dati sul nastro 3.



Se il nastro 3 contiene la stringa 212 significa che D sta simulando la configurazione C_9 di N. La configurazione successiva (nella visita BFS) è C_{10} e sarà rappresentata dalla stringa 221.

Da NMdT a MdT

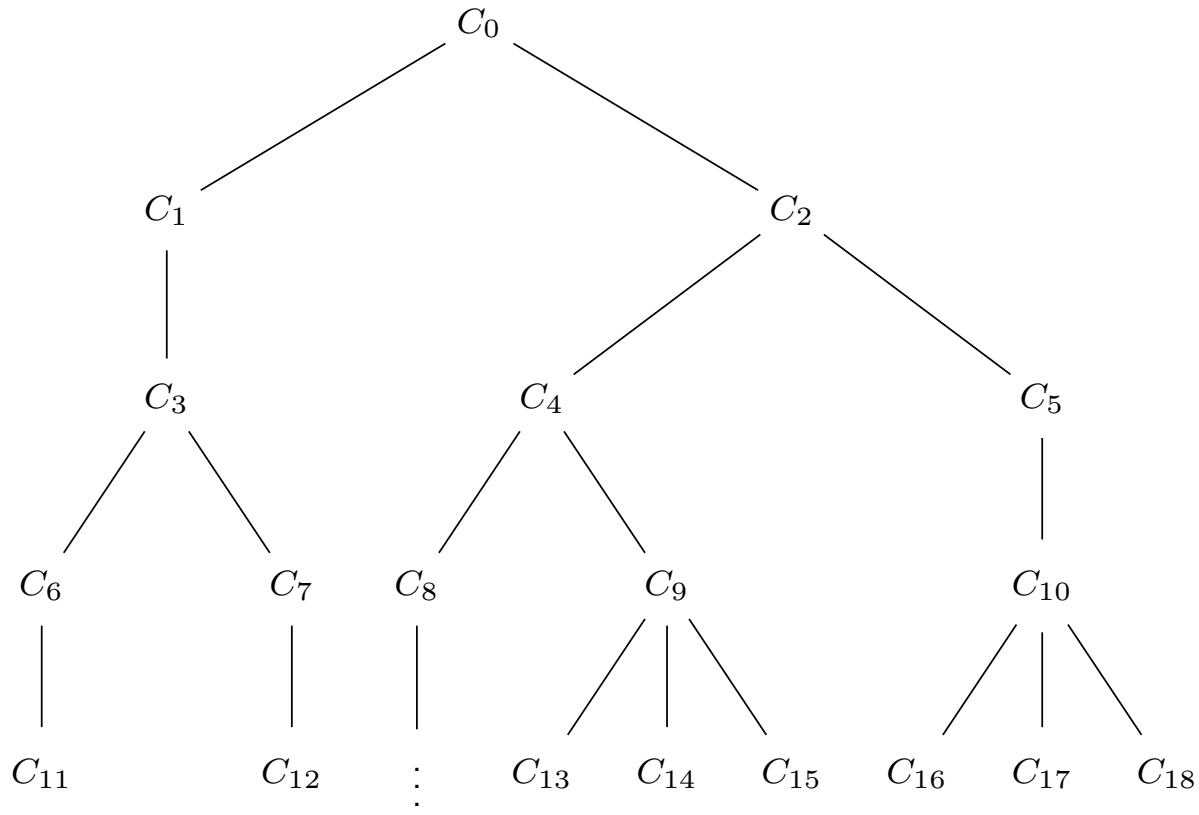
Partiamo dalla rappresentazione dei dati sul nastro 3.



Sul nastro 3 vengono generate tutte le stringhe corrispondenti a numeri crescenti dell'alfabeto b-ario: $1, 2, \ldots, b, 11, 12, \ldots, 1b, 21, 22, \ldots 2b, 31, 32, \ldots$

Da NMdT a MdT

Partiamo dalla rappresentazione dei dati sul nastro 3.



Nel nostro caso, per $b = 3$, la successione sarà:

$$1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, \ldots$$

Le stringhe in rosso non corrispondono a una computazione di N: D le ignora.

Da NMdT a MdT

Esempio: Supponiamo che sul nastro 3 si sia arrivati alla stringa **112**. Cosa fa la macchina D?

1. Copia il contenuto del nastro 1 (che contiene l'input) sul nastro 2. Da questo momento tutta la simulazione (scrittura) avviene sul nastro 2.
2. A partire dalla configurazione iniziale, D esegue la **prima** transizione nella lista delle transizioni possibili.
3. A partire dalla configurazione raggiunta al passo precedente, esegue la **prima** transizione nella lista delle transizioni possibili.
4. A partire dalla configurazione raggiunta al passo precedente, esegue la **seconda** transizione nella lista delle transizioni possibili.

Nota: se sul nastro 3 appare una stringa che non corrisponde ad una configurazione sull'albero, D se ne accorge guardando la funzione di transizione di N e la ignora.

Da NMdT a MdT

Simulazione di N con D:

1. Scrivi 1 sul nastro 3.
2. Copia l'input dal nastro 1 sul nastro 2.
3. Se il nastro 3 contiene la codifica di un prefisso di una computazione di N, allora esegui la transizione corrispondente sul nastro 2.
4. Se sul nastro 2 si raggiunge una configurazione accettante, allora accetta.
5. Incrementa di 1 il numero b-ario sul nastro 3.
6. Vai al passo 2.

Tesi di Church–Turing

Formalizzazioni della nozione di **Computabilità**:

- Alonzo Church creò un metodo per definire le funzioni computabili, λ*–calcolo*.
- Alan Turing creò la *Macchina di Turing*.
- Church, Kleene e Rosser diedero la definizione formale di una classe di funzioni il cui valore può essere ottenuto mediante *ricorsione*.

Si è dimostrato che sono formalizzazioni equivalenti.

Tesi di Church–Turing

Formalizzazioni della nozione di **Computabilità**:

- Alonzo Church creò un metodo per definire le funzioni computabili, λ*–calcolo*.
- Alan Turing creò la *Macchina di Turing*.
- Church, Kleene e Rosser diedero la definizione formale di una classe di funzioni il cui valore può essere ottenuto mediante *ricorsione*.

Si è dimostrato che sono formalizzazioni equivalenti.

Tesi di Church–Turing: *"Se esiste un algoritmo per eseguire un calcolo allora questo calcolo può essere eseguito da una MdT (o variante equivalente)"*